Smart Cube Predictions for Online Analytic Query Processing in Data Warehouses

Prepared by Andrei Belcin
Co-Supervised by Dr. Herna Viktor and Dr. Eric Paquet

School of Electrical Engineering and Computer Science
University of Ottawa

Acknowledgements

None of this would have been possible without the enduring support of my family in the most real of terms, and my mentors for guidance in maneuvering the machine learning world in academia as much as taking my first steps in the data science profession.

A special thanks to Dr. Herna Viktor and Dr. Eric Paquet, for your invaluable guidance during my Masters program, our detailed debates over data analytics, and the frank advice that placed me back on track to stay in scope when I often strayed. Dr. Viktor, I have much appreciation for taking me on since the honours project in my undergraduate years, through my first publication, and into an NSERC collaboration project which allowed me to get my footing to enter the data science profession with confidence.

To my mom, Cristina, for continual advice that has kept me cool headed and focused on the opportunities ahead rather than getting stuck on mishaps or when being dealt the momentary bad hand of cards in life. The value of work and compromise will stay with me for all my life as I witnessed you exemplify them day in and out. Happy to say I caught the data analytics bug from you as well.

To my dad, Niculae, for steering me towards informatics since early childhood. By sharing your experiences, I saw the complexities and intrigue that comes with the domain. Rest assured the attitude of a lifelong learner has been instilled in me as you have done throughout yours.

To my stepdad, John, for the earliest lesson on working among professionals instilled in me an on-the-job work attitude of technical rigour and cooperation with my peers with the highest of stakes in my first 22 years. Alongside your support I've taken

your example to approach challenges head on with hard work to ensure a resolution, sitting idly by is never your approach. I owe my exposure to the world of audio and music production to you, and it has become one of the biggest joys to explore further.

Finally, to all my friends who have been apart of my getting thus far, my many thanks.

Abstract

A data warehouse (DW) is a transformation of many sources of transactional data integrated into a single collection that is non-volatile and time-variant that can provide decision support to managerial roles within an organization. For this application, the database server needs to process multiple users' queries by joining various datasets and loading the result in main memory to begin calculations. In current systems, this process is reactionary to users' input and can be undesirably slow. In previous studies, it was shown that a personalization scheme of a single user's query patterns and loading the smaller subset into main memory the query response time significantly shortened the query response time. The Lattice Prediction under Change Detection and Adaption (LP-CDA) framework developed in this research handles multiple users' query demands, and the query patterns are subject to change (so-called concept drift) and noise. To this end, the LP-CDA framework detects changes in user behaviour and dynamically adapts the personalized smart cube definition for the group of users.

Numerous data mart (DM)s, as components of the DW, are subject to intense aggregations to assist analytics at the request of automated systems and human users' queries. Subsequently, there is a growing need to properly manage the supply of data into main memory that is in closest proximity to the CPU that computes the query in order to reduce the response time from the moment a query arrives at the DW server. As a result, this thesis proposes an end-to-end adaptive learning ensemble for resource allocation of cuboids within a DM to achieve a relevant and timely constructed smart cube before the time in need, as a way of adopting the just-in-time inventory management strategy applied in other real-world scenarios.

The algorithms comprising the ensemble involve predictive methodologies from Bayesian statistics, data mining, and machine learning, that reflect the changes in the data-

generating process using a number of change detection algorithms. Therefore, given different operational constraints and data-specific considerations, the ensemble can, to an effective degree, determine the cuboids in the lattice of a DM to pre-construct into a smart cube ahead of users submitting their queries, thereby benefiting from a quicker response than static schema views or no action at all.

Acronyms

ANN A Neural Network. 48, 63, 64, 67

BBB Bayesian prediction on the Beta-Binomial. 48, 49, 55, 56, 66, 76, 120

CD critical difference. 105–110

CDA change detection and adaption. 44, 46, 48, 78, 113

CPU central processing unit. 3, 4

CUSUM cumulative sum. 37, 38, 68, 70–72

DBMS database management system. 2–5, 8, 9, 12, 18, 20–23, 41, 42, 46, 68, 74, 87, 122

DDM Drift Detection Method. 68, 72–76

DM data mart. iv, 6, 14–16, 23, 34, 62, 71, 81–88, 97–99

DW data warehouse. 2, 5–7, 11, 14, 15, 18–22, 44, 82, 88, 97, 99, 105

E-R entity relationship modelling. 12–14

EDDM Early Drift Detection Method. 68, 72, 74, 75, 77

Gap-BIDE Gap-BIDE. 66, 76

GBR Gradient Boosting Regressor. 115–120, 137

GPU graphics processing unit. 3

LP lattice prediction. 21, 27, 35, 44, 46, 48, 53, 55, 58, 59, 67, 78

LP-CDA Lattice Prediction under Change Detection and Adaption. iv, 9, 41, 43, 45, 47, 48, 55, 59, 60, 63, 67, 74, 81–83, 87, 91, 100, 103, 122

LSTM Long-Short Term Memory. 66, 67, 76, 120, 122

ML Machine Learning. 3, 4, 11, 25, 26, 63, 71, 101, 102, 106, 113, 116, 121, 122

NLP natural language processing. 63

OLAP Online Analytical Processing. 13, 15, 20, 22, 41

PCP Percentage Correct Prediction. 101–103

PHI Personal Health Information. 4

RAM random access memory. 3–5

RNN Recurrent Neural Network. 64–67, 120, 122

SL Statistical Learning. 11, 25, 26

SQL structured query language. 12, 14, 81, 82, 87, 88

Contents

Part I

Methods

Chapter 1

Introduction

In the application of modern data analytical methods often facilitated by queries, software systems are tasked with serving as the custodian of the data and calculating aggregations to respond to user's dynamic requests. With many users interacting with such software, denoted as database management system (DBMS), often asynchronously, a challenge to address is the influx of demand in certain queries over others— from simple select and join to conditional aggregations using nested queries at runtime. The challenge in meeting the users' queries demand is compounded by the sheer size of data tables housed in a data warehouse (DW), size being a function of their tuples. Various user types may have distinct query patterns which may be crucial in prioritization of resources in order to speed up query processing by the DBMS.

There are numerous procedures which impact the time required for a DBMS to return a result in response to the user's query. The first procedure involves the quality of the query with regards to the order of table joins, the number of nested queries, implicit optimization by the DBMS, explicit query optimization by experts of the specific DBMS architecture, and the canonical form of conditional statements rather than a verbose number of conditions on which to evaluate tuples. For example, if a query is to retrieve all vacation spots to which people between 20 to 40 years of age travelled in the summer of 2019, and given the primary table queried contains foreign keys of vacation locations, dates, and a person's profile, then it is advantageous to join on

the date foreign key first, with the person foreign key followed by the location foreign key. This ordering of joins would avoid pulling up the entire list of locations one may have travelled and reduce the range of foreign keys and any attributes of interest retrieved from disk within a limited date range. An explicit optimization involves indexing methods, whether B+ trees, bitmap, or other, in which indices will arrange frequently queried attributes from a table to facilitate quicker expression matching operations, often in the query's filter for valid row values to return in the result or even a condition used to join on. Furthermore, explicit optimization uses views to reduce the amount of data available to be queried and, thus, the total range of rows any join operation has to handle; this is achieved by providing only relevant data to a user class or restricting access to certain data within the full tables stored on disk.

Second, the choice of analytical algorithms often run between a DBMS in and third party software adds to the query processing load alongside human user requests. Aside from running queries from a DBMS' native environment, one can run scripts in programming languages such as Python and R, which are often used for in-depth analysis and Machine Learning (ML). These scripts will run queries on the database to aggregate and build the sets alongside other users' queries. Data sets for ML are often among the largest in size (columns and rows) that a human user aims to interpret visually, requiring much of the database's resources to compute.

Third, the hardware architecture housing the database system contributes to query response time, as it includes permanent storage—referred to as disk space, central processing unit (CPU) and its number of cores, increasingly the use of graphics processing unit (GPU), and random access memory (RAM) closest to the CPU unlike the disk. For instance, the ratio of main memory to database size in permanent storage is crucial as it is widely used for query processing in popular DBMS systems, including SQL server by Microsoft and POSTGRESQL. Certain systems respond to user queries by first importing the required data into main memory to join across tables, filter on values and conditional logic, and finally compute the aggregations to return values in the result table. The less main memory, the less room available to fit data tables along with intermediary results of computations until a result is finalized and, as a consequence, there is more interaction between the two memory locations. These architectures requiring main memory for resolving queries are widely used in small organizations and enterprise environments. The cores which act on the data

brought into main memory determine the degree of concurrent computation in which a server can engage, whether it is executing a query per CPU core such as POSTGRE SQL server's one process per core policy, or other server architectures that split up segments of a large query to resolve faster with multiple cores at a time.

A solution that **aims** to reduce the query response time must be dynamic as the demand changes and account for the prior three facets. This thesis focuses on managing the supply of subsets of the data, available to an organization's employee users to query, within the main memory hardware, particularly in RAM (aside from cache) since it is the temporary storage of data in closest proximity to the CPU. The focus on main memory is driven by the need by widely used and reliable DBMS systems such as SQL server and POSTGRE SQL, to import data from permanent storage for fast access in main memory and execute table joins, filtering on row values and aggregation calculations. Thus, the delay in carrying data from the disk in response to queries leads to slower computation of the response to the query for the user.

Despite the availability of cloud storage solutions, including versions of the previously mentioned DBMS systems, there are risks of data privacy and a firm's competitive advantage being eroded by other organizations, given that the cutting edge cloud systems are offered by other data harvesting and analytics firms such as Google and Amazon. In the case of applications holding onto non-identifiable data (to an individual), it is safe to make use of Google's cloud database packages with ML plug and play-style methods for immediate churning of a client's data, or similarly Amazon's AWS services from the privacy perspective. Given this, we address the risk of allowing an individual's proprietary data be in the custody of a third party which is rarely held to the same limits of liability as the original data collecting firm's clients/user agreements. In particular, organizations holding Personal Health Information (PHI) are subject to safe harbour laws in North America and the GDPR laws for data protection in the European Union. Crucially, there is a trend of adopting these standards and the associated legal consequence of failing to enact proper safeguards for the collected data by custodians of PHI which are often inclusive of personal data [Ema13]. As a result, the storage of data on servers not belonging to the firm which acquired it imposes severe risks and legal liabilities to the custodian which originally collected the individuals' data; with this in mind, organizations are not opting to use casually secured database architectures or cloud computing solution suites offered by third

parties. Cloud computing requires a large-scale development team and a generous budget, which is why many organizations do not (yet) have their own in-house dedicated teams to manage their own cloud architecture as it is not their primary source of expertise and source of revenue. Only recently have some developers of cloud computing suites such as SAP begun to offer products which can be solely stored in-house and operated within an organization's array of information technology.

Another industry case is Wal-Mart and its Retail Link online portal for its associates to access sales and inventory data sets as they change over time to facilitate decisions in its just-in-time supply management strategy. Retail Link (trademark of Wal-Mart) is the organization's in-house proprietary system that adds value to its business to justify its immense development and ongoing costs. Wal-Mart is an example of a global organization that makes non-individual granular data available to hundreds of thousands of associate users world-wide via a cloud-like architecture (precise details of this proprietary technology are unavailable publicly) which is not feasible for many medium and small organizations; Wal-Mart's in-house development decision is to retain its competitive edge by securing access to the data. In the end, the more traditional setup of locally stored and maintained database systems for many organizations has remained a large percentage of users given the ongoing development and funding into SQL Server, POSTGRE SQL, and ORACLE's sql database variants; with an increasing trend in regulations implemented globally [AE20], these database systems remain a safe option in terms of legal liability, cost-benefit business cases, and reliable DBMSs.

Given this context, it is the **goal** of this thesis to formulate a framework that specifies the correct data to place in RAM and be executable in near real-time and within memory constraints.

DWs are large-scale repositories that are frequently accessed by multiple users and that run analytical applications that use aggregated data across multiple relations. For instance, in retail, decision makers often query the evolving trends in consumer behaviour based on time, location, and product lines [KR13]. The optimization of diverse user queries is a difficult task, since the usage patterns vary over time and the resources used by the analytical applications are heavily influenced by the data used (relations, tuples, and attributes), as well as the aggregation functions (e.g.

icebergs, summation, counting, etc.) [Han15]. In relational databases, the schema for warehousing data is a result of dimensional modelling in which the *fact* table houses numeric measures pertaining to the main events of interest linked to many *dimensional* tables containing the attributes of the measures [KR13]. A fact is supported by n dimensions; these offer the most context to the fact of an event when a join across all of these is performed. There are typically hierarchies exhibited by the attributes; in the retail example, the hierarchy is typically the following: date and time, location (city, state/province, country), product line levels, the composition of a complex piece of equipment, and more [KR13]. The level of aggregation is a key feature in the construction of an n dimensional join of tables surrounding facts (tuples of the fact table), which, in turn, allows a user to inspect these various levels of information in a single query result.

However, the n dimensional query results are referred to as cubes for short and, for reasons explained in the next chapter, contain large amounts of repetition among the attributes' values to prioritize the swifter computation of queries rather than efficient data storage on disk at the cost of otherwise frequent joins. The benefit of dimensional modelling is the reduction of table joins to build the context surrounding facts through *concept* hierarchies, albeit at the cost of frequently repeating attribute values in permanent storage; this is in contrast to more traditional relational database designs which require iterations of normalization to achieve tables with (nearly) distinct attribute values. As stated in [AV16]: "The need to merge such heterogeneous data sources has introduced an important problem for many companies; thus the problem of size of the data and [degree of] dimensionality." The *data sources* refer to intermediary data (flat) files or other databases which, once they are collected and their contents are integrated (the process of data staging), only a single database called a *data mart* within the DW [KR13] may be queried. In addition, the *problem* is alleviated by solutions which address data storage, cost of query processing [AV16], and managing the supply of data subsets to meet the demand by user's queries. This thesis builds on the framework first developed in [AV16] to reduce typical query response time in a DM and to extend its capabilities in estimating *user interest* exhibited through the submitted queries to the mart. There are three primary reasons for extending the module in [AV16] to specify the data that need to undergo the author's solution or improving query response time. First, the work in [AV16] addresses the

scenario with a single user; however, multiple users with differing interests is a more likely condition to be faced by companies and organizations. Second, the author's approach of accounting for user interest is summarized as: "This user interest is not static and therefore precomputing the personalized cube in advance for each user is not a practical solution." While it is acknowledged that user interest is not static, a differing view of data-driven predictions is presented in this thesis; here, the main contribution is applying a predictive modelling ensemble of Bayesian statistics, sequence pattern mining algorithms, and machine learning methods to precompute data cubes. Third, the evaluation of approaches in the literature shows a heavy reliance on the static assumption or infeasible protocols in day-to-day operations within a firm.

1.1 Motivation

The goal of this research is to predict, in near real-time, the data of interest for the whole group of users against such a DW. These predictions are subsequently used to construct so-called personalized Smart Cubes which contain aggregated data per user group that evolve over time. This approach reduces the computation time of queries, by providing user groups with access to current, relevant data.

The motivation for this research is, thus, to specify an end-to-end adaptive approach for resource allocation of cuboids from permanent storage into fast access memory in the form of materialized cubes within the DW domain. First, a database server needs to process multiple users' queries by joining various datasets and loading the result in main memory to begin calculations. In current systems, this process is reactionary to users' input and can be undesirably slow. In [AV16], the authors implemented a personalization scheme of a single user's query patterns and preloaded the data. The scope of [AV16], however, does not include many users simultaneously in the system; this research addresses this shortcoming. Furthermore, it can be acknowledged that the query patterns are subject to change (so-called concept drift) and noise; again, current solutions do not handle these aspects. To this end, a framework was developed to detect changes in user behaviour and to dynamically adapt the personalized smart cube for a group of users.

This research lies within the field of adaptive intelligent systems, with a focus on

machine learning and data warehousing. The methodology includes elements of statistical learning, machine learning , and time-series analysis to provide an end-to-end solution to adaptive predictions in the prior stated domain.

1.2 Thesis Contributions

In this work, a framework for multi-scale prediction and adaptation to changes in user interest is the primary deliverable; the secondary deliverable is a procedure to guide someone in assembling the components of the ensemble validated here to fit their use case, particularly with respect to an organization's exact data and DBMS constraints. A challenge addressed in this thesis is to limit the assumptions to the minimum number required by the prediction methods, unlike alternative frameworks discussed in the next chapter. The assumptions made in personalizing a collection of data for users often lie in the user interest estimation process per methods surveyed from literature; these are omitted in this framework in favour of data-driven models that predict this interest in near real-time. The ensemble of prediction methods is drawn from numerous data analysis domains: Bayesian parametric statistics, data mining, machine learning, regression modelling, and functional data analysis (functions of trends and signals in place of a data point or tuple). The methods carried into the final experimentation phase cover a wide range of interpretability and predictive ability trade offs, and each has merit in certain operational climates. Furthermore, the final three predictive methods are sufficiently different in their complexity to program and infer from, which is advantageous in ensemble learning as we try to boost the prediction correctness. The contrast among these methods enable organizations to implement the framework in an incremental fashion if the full ensemble poses a high risk in development planning.

Therefore, this thesis is aimed at extending the framework by [AV16] with a prediction of cuboids module adaptive to concept drift. The prediction functionality is a key improvement in reducing query response time while allowing the framework in [AV16] to handle multiple users in place of the authors' implementation on user interest and cuboid specification for the materialized set. The prediction module runs in near real-time to allow the smart materialization stage in [AV16] to run in time for the effects

to impact query response time. The dataset on which the framework is validated is a new synthetic set of four user group query profiles motivated by real world data and extended to enclose a mixture of trends, cyclic query patterns, and changes in user behaviour over time.

1.2.1 Summary

In summary the contributions are:

1. Predictive procedure on multiple time scales.
2. The predictive module is adaptive to changes in the users' interest over time.
3. An ensemble learner to leverage well performing and diverse combination of predictive models and change detection methods with adaptive strategies.
4. A data synthesis procedure for validating the framework in the absence of readily available data sets.

Altogether these points make up the LP-CDA framework developed in this thesis.

1.3 Organization

The remainder of this thesis is organised as follows. Chapter 2 examines various approaches for DBMS personalization and their contrasts, branches of computational learning to carry out predictions of data processes, approaches to handle changes over time of such processes, and, finally, the candidate methodologies to include in the proposed framework. Following the selection of methodologies to adapt from Chapter 2, the framework is specified in Chapter 3.

Part *II* focuses on the experiments conducted, their results, and the conclusions drawn from the analysis. The construction of the datasets, meant to represent real-world user behaviours is a key section in Chapter 4. The analysis in Chapter 5 extends beyond contrasting approaches and considers integration of these approaches on the

basis of various measures aside positive prediction rate in the case of limitations faced in hardware. Finally, the future work directions from Section 6.1 establish a near-term road map for exploration to extend the work in this thesis and the methodology [AV16] with which it integrates.

Chapter 2

Background

The chapter begins in Section 2.2 with a discussion of the framework on which the thesis builds, in addition to a contrast of previous research on query demand modelling (a by-product of user interest in data) in a DW. This thesis focuses on a meta data-driven analysis of user queries to leverage the smart materialization achieved in a prior framework. An introduction to analysis methodologies is provided in sections 2.3 and 2.4. A gradual narrowing of the vast field of data analysis is presented in this chapter, beginning in Section 2.3.1 and ending with the identification of a conclusive set of methods to carry into experimentation in Section 2.5. It should be noted that elements of signal processing and data mining are mentioned briefly where they can be applied to alleviate limitations often faced in Statistical Learning (SL) or ML.

2.1 Data Warehousing

In response to the continual increase in the amount of data tuples and the number of events observed and of interest to analysts for aggregation, the usual entity-relationship modelling of a database was replaced by dimensional modelling. The immediate difference is the shift away from normalization, the design of data tables to reduce redundancy in the values of attributes stored permanently in order to maintain the accuracy of data (one source of truth for attribute values), and the reduction

of storage space at the cost of numerous table joins for (routine) queries. For instance, processing a client's item or purchase order return to issue a refund requires a query to filter data on the client's personal identity attributes, order number (if available), or date of purchase, etc. The prior query scenario constrains the retrieval of tuples from multiple tables with the intention to receive a few rows or less to then be joined across tables; this translates to using the *where* and $inner/left$ join on the *clients*' table clauses, as described in the structured query language (SQL). A database's conceptual schema likely follows the entity relationship modelling (E-R) approach [KR13], which at least has the table holding onto clients, products/services, purchases, and stores/location of sale, which would be interlinked by a set of foreign keys. In these tables, a combination of values (rows) among the attributes (columns) appear only once; one record for each unique client, for every product/service, etc. A database design that can fit the use case described earlier is depicted in Figure 2.1. The numerous joins across tables are necessary in the best case where the client has an order number (ID) at the time of the return request and, in the worst case, where fragmented information about the order is presented by the client (the location of purchase and perhaps certain items in the order). The benefit of the normalization of attributes in the design presented in Figure 2.1 is ease of maintaining accurate data, as an attribute is recorded only once in most cases; a side effect is the reduction in data storage size since redundancy is reduced.

On the other hand, consider the analysts' use case of perpetual inventory control or analysis of product pricing over time and correlation with sales in various locations. These analyses require the aggregation of all tuples satisfying a set of constraints—the amount of data to load into main memory for computations is vast. The aggregation of data is typically of interest along the dimensions of time, location, and trends associated to groups of client/people demographics. This second use case differs from that associated with Figure 2.1, since the first example is focused on retrieving specific records whereas the second is focused on generalizing over a set of parameters. The complexity of operation required in the generalization of data competes for the DBMS processing capacity alongside the table join operations; this competition for resources in the case of a relational schema with degrees of normalization leads to long query response times [AV16, KR13]. The complexity of joining many tables routinely can be reduced if the schema is designed following dimensional modelling instead of E-

R, given that the storage capacity has vastly increased over the last several decades, foremost in disk and then in main memory (RAM, cache); computing capacity relative to the amount of data queried, however, has not increased at the same rate.

To determine and record important historical aspects of a business process often over some period of time and set of location s, the need to *warehouse* data covering the vast range of time and locations and granularity of detail within these dimensions of interest to an organization led to a new formulation of database schematic designs. The method refined in [KR13] is dimensional modelling, and within this method, the specific schema is star shaped. The aggregation for either business intelligence (BI) dashboards or in-depth pattern discovery and trend analysis using data mining is facilitated by Online Analytical Processing (OLAP). The star schema in Figure 2.2 is an example of a result of dimensional modelling which would fit the second use case described previously.

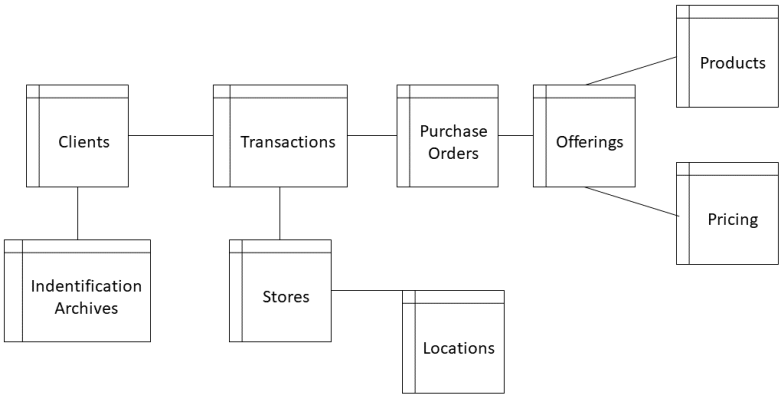


Figure 2.1: Partial schematic diagram (on the tables of interest) for the example provided in the text of a database designed using relational entity modelling. The third normal form is exemplified by way of the linkage of purchase orders being a collection of product offerings, where an offering is a specific pricing method or amount associated to a product at the point of sale.

The star schema exemplified by the authors of [KR13] in Figure 2.2 differs from the E-R schema highlighted in Figure 2.1 primarily in two areas:

- the sheer volume of attributes and the cumulative gigabytes to terabytes of data

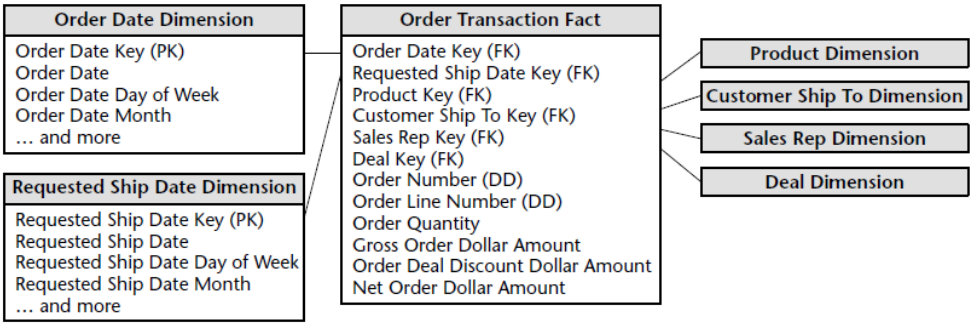


Figure 2.2: This diagram is Figure 5.2 from [KR13], depicting the order handling business process with selected details. Despite the fact that the detail (in number of attributes and their granularity) surpasses the relational database schema observed at a high level in Figure 2.1, the number of tables is proportional. In contrast, a join on four tables is required to obtain the details of the products purchased and their prices at the moment of sale in Figure 2.1, as opposed to the single join on product dimension and/or deal dimension to complement the dollar amounts stored in the fact table in this figure's schema.

subject to aggregation queries

- the number of joins is reduced to access the details of any dimension. For example, only one join is required with the fact table to obtain all the details of any given dimension unlike the three joins required to obtain the prices of specific products purchased and two table joins to obtain geographical locations of purchases in Figure 2.1.

The DM exemplified by the star schema in Figure 2.2 can be one of many within a DW in order to facilitate the numerous operational processes of an organization.

2.1.1 Materialized Views

In both E-R and dimensional schema environments, an aggregation of tables often referred to in many sequences of queries may be saved, as a quick reference, in the form of a view; this view is assigned a unique name which refers to an underlying query often joining a number of tables and filtering rows using the *where* clause in SQL. A view is beneficial for the following:

- ensuring consistent business logic is applied where it can be standardized, this lends to the use of updated data in calculating metrics all the while the backend undergoes continual changes. For instance, defining notions of *best selling items* where a view may be the query which subtracts product returns and defects in monetary value from the amount sold in total; business analysts could conduct their queries on this standard set of items and their measure of net sales.

- personalizing the schema for different classes of users, often eliminating the need for users to conduct repetitive joins for their use case. Consider the two roles: an analyst interested in calculating the perpetual inventory flow versus an analyst focusing on clientele and their distribution in locations of purchases (from figures 2.1 and 2.2). These two roles will often conduct their analyses on a different cuboid, and using a view will eliminate the redundant query writing by each analyst sharing a role.

- maintaining storage space. By default, a view is a reference to a query which is calculated at runtime; there is no permanent storage increase though the performance could be diminished if the view is large and complex to generate repeatedly.

In addressing the last point above on performance, the ability to *materialize* a view which remains in main memory would ensure the view referenced is already loaded in main memory and the basis for user-specific queries. This materialization is beneficial in reducing the query response time [Han15, AV16] and, in the DM context, requires construction of the *data cube*. As depicted in Figure 2.3, a data cube consists of all combinations of dimensions given an apex (the superset of data), and a combination represents the join between the elements; the purpose of the lattice arrangement is to enable an analyst to switch between different perspectives of dimensions of interest at runtime without issuing independent queries to construct the individual joins of combinations. In an OLAP environment, multiple joins within a data cube are often of immediate interest and subject to some back and forth querying; as such, it is reasonable to retrieve all these possible joins in one query.

Although the DW architecture referred to throughout this thesis follows the relational OLAP (ROLAP) paradigm, there are two other architectures: hybrid OLAP (HOLAP) and multidimensional OLAP (MOLAP) [KR13]. The ROLAP paradigm

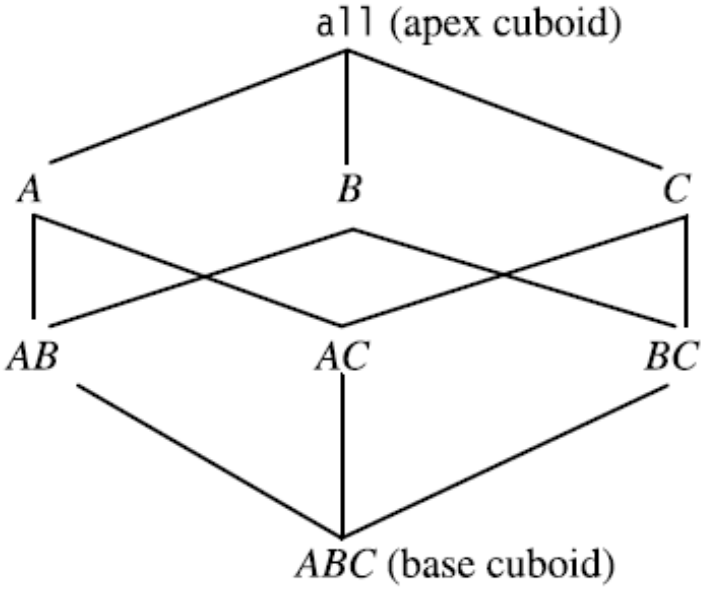


Figure 2.3: A general data cube for any three dimensions {A, B, C} in a given DM, obtained from Figure 3.1.4 of [Han15].

is beneficial in working with increasingly large amounts of data as the queries are computed on the data tuples as stored in tables directly, unlike in MOLAP where these queries act on a data cube instead. Thus, queries written on ROLAP database architecture can lead to multidimensional results viewed statically in tabular form with extensive aggregation across large data sets at the cost of more complex queries required to construct data cubes as needed. However, a well performing MOLAP architecture utilizes smaller datasets as the data are stored in a multidimensional array using sparse matrices—the data cube structure is fixed on initialization and this existing structure offers faster access and dynamic viewing of its contents. In a situation in which fast access to a ready made data cube is equally important as the ability to store large databases requiring multiple data cubes, the HOLAP architecture can be successfully applied. However, HOLAP does inherit the limitations of both architectures it integrates; whenever access to the relational tables is required for data not available in a MOLAP setting, the access is slower and more complex queries are required—this is the cost to perform more extensive aggregations than in a MOLAP setting.

For instance, in the ROLAP setting, the data tables depicted in the orders DM from

Figure 2.2, where $A = product$, $B = Date$ and $C = Customer\ Ship\ To$, then the cuboid AB is the join of the fact table with these two dimensions to allow for aggregation of types of products sold and their monetary value by either categories, product lines, or other grouping attributes, within a specified period of time. A trend analysis of the monetary value per product group over time is facilitated by the AB cuboid. As another example, the BC cuboid and analyst can monitor the overall amount of transactions either in frequency or monetary at different locations in the world over time. Using the figure of a data cube as guide, it may be noted that there are at least 2^d cuboids in the whole cube for d dimensions selected. Furthermore, an account of the concept hierarchies present in some dimensions such as dates and geographical locations where the granularity of detail can be chosen by users leads to the following equation:

$$total\ number\ of\ cuboids = \prod_{i=1}^{d}(L_i + 1) \tag{2.1}$$

Where L is the number of levels of granularity for dimension i [Han15]; for date, four levels may be {day, week, month, year}; and *customer ship to location* may be {city/town, region/county, province/state, country}.

Materializing a view containing a data cube is costly given the number of cuboids, and without any intervention on the side of database programmers, the redundancy of attribute value spanning the cuboids creates a need for optimization. For this reason, the authors of [Han15] state it is unrealistic to precompute the whole cube, and in [AV16], the authors go further to state that precomputing altogether a data cube is not worthwhile, rather it is better to optimally compute it near the time it is referred to by user queries. In this thesis, we are in agreement with [Han15] and will leverage the optimal construction of data cubes from [AV16] in order to precompute fragments of the cube (certain cuboids) of which any other non-materialized cuboids could be quickly generated at runtime.

2.2 Prior Research on Personalization of View Materialization

In a previous study [AV16], a dynamic smart cube construction and smart view materialization is applied to a user's query pattern in a DW environment. The *smart construction* aims to achieve minimal loading of data into main memory so that any cuboid in a data cube could be materialized faster than loading the default cuboid in the DBMS. Next, according to the user behaviour witnessed by the system through the query pattern, the most frequent subset of cuboids is constructed from partitions of the data cube to store as materialized views. It follows that the user behaviour changes along with the frequent cuboid set; the views are *smartly materialized* to reflect the set of frequencies, subject to main memory size constraints. The constraints to the problem explored in [AV16] and in this thesis are storage space and query response time. The authors of [AV16] show through their experimentation that the dynamic smart (constructed and materialized) views achieve query response times comparative to competing approaches at the time using less storage space, meaning more of the frequent data could be fit in the materialized views. In addition, an early result of the research in [AV16] reaffirms that materialized views of relevant data cube subsets for queries increases the query response time.

In contrast to [AV16], we are introduced to a method relying on a framework named *MapReduce* in [SY17] to construct large data cubes in the 216 gigabytes to 1.75 terabytes range within the hour given a fleet of 400 machines. This contrasting approach focuses on reducing the number of "copy-add" operations in calculating aggregations from many rows of data and distributing other operations concurrently across the set of machines. Of significance, the method in [SY17] requires a large collection of machines integrated to work in parallel and is reactionary to user query behaviour. While the constraints in the cost model for [SY17] involve the algorithmic complexity in constructing a data cube, whereas those in [AV16] involve *user interest*, run-time, and size of the resultant data cube. The common dimension in validating the two approaches is the run-time or algorithmic complexity, which contributes to query response time improvements. However, the primary difference is the focus on dynamic user behaviour in [AV16] under some assumptions, whereas [SY17] does

not have this focus. The operational setting in the two works differ significantly with respect to real world applications, in that [SY17] is intended for large-scale computing architectures at the business enterprise scale whereas in [AV16] the experimentation phase takes place on a single computer acting as the host and server client where it is a reduction resource consumption which scales up with increases in data cube demand and variations. The concurrent demand in multiple data cubes and change in this demand was not a point of discussion in [SY17], while it is a primary consideration in [AV16].

In addition to the two prior approaches of reducing query response time with a focus on data cube construction, other methods focus on the selection views to materialize so they can resolve the bulk of user queries as they arrive. View selection methods operate a cost model balancing the size in memory of views, frequency of queries which a view can resolve, and the cost of maintenance of such materialized views [MB12]. Given a set of queries observed over a period of time and their frequencies, a *query workload* is used as training data for view selection strategies varying from immediate to deferred updating materialized views as the data tuples change in an incremental manner or are re-constructed from scratch. Additionally, whether the selection algorithms involve *query rewriting* in such ways which would use the existing set of views or rely on the *syntactical analysis* of workload (table relations within the set of queries), the benefit of a set of views is limited by the relevancy of the workload on which it is trained. As a result, the detection of change in relevant workload is not monitored explicitly since the cost model used to decide which views should be materialized is expected to reflect the shift in user behaviour. However, need to detect a change explicitly is discussed later in this chapter, as many models are not sufficiently robust in their formulation to implicitly adapt. Finally, a limitation shared by the methods in [MB12, AV16], is the reactionary approach to make the determination of what constitutes user queries of interest rather than making the distinction between what was (historical patterns) and what is underway (concept drift) to define the materialized view.

Thus, this thesis focuses on improving the dynamic aspect of smart materialization in [AV16]'s framework in order to address many users' behaviours in near-real time. Given a supply of cuboids in a *data mart* $\subseteq DW$ queries of some number of users, it is unknown if the system can predict in near-real time the demand of cuboids to

initialize the smart materialized view. The allocated main memory constraint limits how much the system can supply to meet the demand at any given time regardless of the accuracy of the demand prediction. If the materialized view has a degree of overlap with all data needed to resolve queries in a time period, then *by transitivity, a materialized view satisfies the query response time in proportion to this overlap*. Thus, query response time is not an explicit measure; instead, the prediction correctness is indicative of overall response time improvement.

A high-level description of the main paradigms among the approaches detailed in the next chapters follows. First, the data used as input s to any of the approaches is **user blind**, since the methods are not meant to focus on user-level analysis or even user groups. The reason for this is that approaches which make assumptions about users in order to group them together for joint consideration in making representative materialized views introduces a bias that is difficult to update and validate in near-real time. The benefit of most styles of approaches surveyed by [AS15] does not outweigh the complexity of computation and errors from bias. In a DW environment, OLAP personalization continually uses subjective input of domain experts to create and maintain profiles or to define the rules of dynamic personalization [KB12, AS15]. The effectiveness of prior personalization relies on correct human input, and predicting the elements that should be in a materialized view is sensitive to timeliness of updated profiles; a detected change resulting in a change in a user profile needs to be computed in time for the remaining system functions to make use of this updated profile in materializing a view. In order to predict user queries we treat actions of all users as actions of the system, and in turn pose the question: "how will the DBMS predict the cuboids to smartly materialize given the demand the system experiences over all; instead of predicting the exact query specification per user?" As a result, complexity and a degree of bias attributed to outdated or lesser reliable personalization are not incurred by the approaches explored here. It is also true that the equal weighting of user actions allows for minority/majority user class situations, which can lead to extreme levels of user satisfaction. The approaches under simulation focus on meeting the demand of cuboids and ensuring their presence, subject to available memory, in a materialized view ahead of time tends to minimize users' wait times for query results. However, the recommendation role in OLAP query optimization [KB12] is beneficial, in addition to the dynamic materialization achieved in this work.

The approaches in this thesis involve computing predictions on multiple scales of time, making predictions at different scales of time, and then using these predictions as inputs into the final decision-making process, deciding which cuboids to materialize. First, to attain our goal requires us to consider three **overall dimensions** = {*user*, *query*, *time*} at any moment to make a prediction; instead, by forcing the analysis to be user blind and fixing the time of consideration, an approach's complexity is further reduced. At the moment a prediction time is fixed, and as the system continually knows the values of time without error, we can consider that at the local (per scale) level, analysis time is not variable; thus, the **narrowed scope** = {*query*}. Furthermore, the query is broken into its cuboid components that appear in main memory when it is answered, and our modelling will use this metadata in the form of histograms (counting cuboids overtime) to describe the user-driven events in the DBMS.

In summary, candidate approaches in exploration address the following: main memory constraints (upper bound) within which to fit materialization, being user blind, operating in a multi-scale analysis manner, and, by transitivity, minimizing the query response time.

2.2.1 User Behaviour Modelling

A lattice is a segmentation and arrangement of dimensions resulting in subspaces in a DW [AV16, KR13]. The **lattice prediction (LP)** process summarizes the events in the query as subsets of data available in the DW [AV16]. Suppose the following data are observed in a cycle: {A,B,D,F}. These can then appear in a lattice as {ABDF} or {A,BDF} or {AB, DF} or all remaining combinations. In LP, the element AB represents the case that data A and B **jointly** occur within a query (consequence of a *join operation*) in a cycle by any number of users. Instead, {A,B} is the case that data set A occurs in a query **separate** from the query where the dataset B is observable. In a cycle, if the estimated observed data sets form the cuboid set {AB,B}, this set is a distinct observation from {AB}. The estimated observed cuboid set is then transformed into a ranked set as the output of the LP given as input to a component of this system to prioritize the smart views to materialize [AV16].

The authors in [AS15] use the term *adaption* to describe the system making edits to the formulation of the query submitted by a user, or the system adapting its own resources to cater to as much of the demand exhibited by the users' queries; this thesis accomplishes the latter by way of schema adaption. The adaption is performed in two parts: deciding which dimensions to filter out of scope for materializing a view as in Chapter 3, and constructing a smart cube most efficiently, achieved by *smart materialization in [AV16].* The branch of DW personalization labelled *recommendation* is synonymous with the recommender systems employed by e-commerce and decision support software; this set of approaches assumes the user will likely need to select a complementary or a better formulated query among the system's suggestions. Intervening in the workflow of the user to either reaffirm the query to submit or accept another query in place of the original leads to a baseline increase in cost on the time-axis proportional to the time it takes a user to decide. Alternatively, the adaptive approach actively reworks the query submission without waiting for user input. In both approaches, the topic being addressed is the basis on which we recommend or adapt user queries?

User profile-based approaches leverage domain knowledge overwhelmingly, which in practice depends on well-timed updates in near real time. In addition to the survey in [AS15], a summary of database personalization approaches is provided in Table 2.1 and explored further below. Of the two branches of OLAP, personalization is observed in published works; this thesis focuses on neither, however, because there is no *recommendation* to the user nor a forced edit on their posed query to the DW. The injection of domain knowledge (bias) is a manual process by way of DBMS admin queries or digital forms acting as interfaces for non-programmers to specify preferences; this clearly highlights the time delay of user profiling in the methods covered in [AS15,KB12], and assumes the bias is correct. In other words, the analysis of user queries is left to the domain experts and then fed into the system to act on it (putting aside the potential problem of competing or contradictory profiles).

Next, a series of modelling languages such as PRML [GPMT09], also covered in the survey, generate a rule-based multi-level hierarchy given "explicit extraction of the user knowledge which may disturb the user and slow the personalization process" [AS15]. The prior approach relies on the user to self identify a shift in their behaviour, meaning they must decide if their actions are outlying, an abrupt change, or part of

Approach	Benefits	Limitations
PRML [GPMT09] and extensions	⟶ rules-based adaption of schema ⟶ extensive conditional logic to reflect user behaviour	⟶ explicit rule extraction, assumes declared rules are reflective of actual behaviour ⟶ no concept drift adaption, relies on users maintaining their relevant set of preferences
MyOLAP with West Algorithm [GR09, BGR11]	⟶ granular, row-wise data selection to materialize in addition to dimension selection within a DM ⟶ individual level of personalization	⟶ explicit user preference extraction, assumes declared preferences are reflective of actual behaviour ⟶ no concept drift adaption, relies on users maintaining their relevant set of preferences
Extended MyOLAP [AGM+11]	⟶ association rules via data mining applied to implicitly extract preferences ⟶ benefits of root approach carry over	⟶ bias carried over from root MyOLAP approach, assumes MyOLAP describes actual data brought into main memory ⟶ no concept drift adaption by the DBMS
MDX query adaption [RT08]	⟶ ranks preferences to prioritize their effect on personalization	⟶ explicit preference extraction and bias as seen in some above approaches ⟶ narrowly works with MDX queries, does not consider SQL or no sql frameworks.
Smart Cube Materialization [AV16]	⟶ individual level personalization given known and stationary behaviour ⟶ achieves main memory and near-real time constraints ⟶ reduction in space utilized compared to competing smart cube materialization schemes	⟶ does not demonstrate use with n number of users ⟶ does not empirically model user behaviour ⟶ no adaption to concept drift
[KB12]	⟶ leverages principals of dimensional modelling covered in [KR13] to partition data in DM that are queried on	⟶ qualitative analysis with explicit preference extraction ⟶ no concept drift considerations or empirical data analysis

Table 2.1: Comparison of a selection of schema or query adaption approaches, most of which, except [AV16], are surveyed in [AS15]. These were selected to cover a diverse set of ideas explored previously, however with consistent limitations.

an incremental shift [GZB+13]. As an alternative to PRML and its variants, our framework for prediction of cuboids in this thesisbe is intended to be merged with the smart view materialization algorithm achieved in [AV16]. Granularity of users' actions, though modelled differently, is a shared concept between PRML and the multi-scale analysis applied in this work.

Authors of [KB12] conduct user profiling similar to [AV16]; their method is classified under the *schema adaption* section in [AS15]. The authors state: "We will follow a qualitative approach, since we think it is more natural for the user to express preferences among predicates directly". These predicates, however, assume correct preferences are submitted by the user, meaning they are reflective of the preferences that would otherwise be inferred from meta data, and that the user profiles are kept up to date. For real-world applications, it is not possible to scale the authors' approach to large groups of users of varying roles and posing varying complexities (by number of joins and tuples) of queries.

An approach similar in nature to syntactical analysis for view selection mentioned in prior sections is the method for personalization introduced in [RT08], where an MDX query is modified to align with priorities set out in a user profile. Unlike an SQL query, the MDX-type queries construct a multidimensional view instead of the relational table with two axes (rows and columns) and operate more conveniently with data cubes as both MDX and data cubes can have any number of dimensions or axes. However, it is entirely feasible to query cubes using SQL and build relational result sets by reducing a multidimensional query result into a row by column table (this is exemplified in the appendix) by using the group by clause. Otherwise, in MDX the group-by function of SQL is implicit in the multidimensional result table and specifies the measures or attributes of any number of axes (SQL is limited to two axes); ultimately the two query languages differ in syntax. This query personalization method from [RT08] can be applied in any OLAP setting (hybrid, relational, or multidimensional) as they all store data cubes as an intermediate step or in permanent storage. Due to the explicit extraction of preferences to determine the way a user's query is modified as submitted, this method suffers from real-world considerations such as accuracy of submitted preferences by the user and continual relevance, and it excludes SQL queries which are used frequently in ROLAP and HOLAP environments.

Further in [AGM+11], the authors mine association rules on queries and preferences explicitly written by users in the MDX query; an extension of the MDX syntax particular to this method is required to be consistently and accurately used by the user. There are significant barriers to adopting this process by everyday users of OLAP, as it requires an additional step beyond the prior personalization methods, which is learning new syntax and incorporating extra clauses in the MDX query to convey a user's preference for every query of interest to the user for resolving faster. The method in [AGM+11] has an initial cost of adoption in terms of time required to learn and apply from every user hoping to benefit from smaller data cubes generated per their explicit preferences. A shared advantage with [AV16] is the row-wise filtering of relevant tuples, and this is appropriate when the personalization is reactionary to a user query; however, this poses a significant obstacle when trying to anticipate the data required by the user. Since a mistake in preloading the filtered tuples could render a materialized table useless, the approach taken in this thesis is to anticipate the dimensions required within a data cube and leave the row wise reduction in the cuboids as per the smart cube construction step in the framework of [AV16].

Thus, approaches explored in this thesis are based on evidence (cuboids) left behind as a result of users' queries (historical data), and the need to profile users to approximate tendencies in data queried is avoided as we infer directly from meta data.

2.3 Statistical and Machine Learning

Stemming from elementary statistics, the SL branch focuses on inferences about the population and groups of interest, to examine and specify relations between attributes (inputs) and a response (output) variable. There are strict methodologies for validating a model fit to a data set and the claims made about the apparent relations. For instance, with parametric regression, validations of the assumptions made by this methodology after fitting a model, in addition to the inferences that may be made, are provided in [FEH15a, She09a]. In contrast, in ML, the relations among variables tend to be difficult to interpret, and thus the predictive performance of the models is the primary focus.

The approaches in SL and ML to conduct affirmatory, exploratory, or predictive

analyses differ as they are tuned to work in different environments with respect to the number of observations and dimensions. ML succeeds at churning out models leveraging complex patterns in the data usually at the expense of interpretability, as the models tend to favour variance over bias ([JWHT13]). In contrast, SL methods require aggregation of the raw data or transformation to a higher degree before using data as input, as is often the case for correctly specifying distributions, applying appropriate smoothing, and denoising as strict prerequisites; in ML, these are seen as improvements (optional). The contrast is reported in the text [JWHT13], as well as between the texts [FEH15b, RS05a, MQJH15, FV06] in SL to [Han15] and part 2 of [PMV17] in ML.

A significant contrast is the treatment of clustering in these approaches. In SL, the goal is to understand within and between cluster relations, and the groupings are known prior to analysis; this approach is still part of supervised learning as seen in part 1 of [PMV17], specifically [NM17]. In contrast, in unsupervised learning the clusters are unknown and at most the kernel defining the nature of these clusters' shape is known prior to analysis; this is introduced from a SL perspective in chapter 14 of the text [HTF09], followed by a ML perspective detailed in the texts [Han15, PMV17]. The quality of clusters is evaluated alongside prediction quality in the supervised learning setting seen in [NM17] using 10-fold cross-validation and formal statistical tests, whereas the unsupervised setting relies on *interestingness measures* from [GH06]. The differences in clustering for the two branches of analytical learning are evident in the approaches mentioned at the beginning of this chapter where the clusters (groups) of users are a prior as opposed to using unsupervised learning to learn the evident groupings from the data; this is discussed in further detail in Section 6.1.

A commonality between the two branches of learning from data is classification, as both treat this prediction problem in the supervised learning setting. The many forms of classification often of the nonparametric form are alternatives to some of the initial approaches in the SL domain like logistic regression [She09b], as well as variants of KNN regression and random forests as introduced in [JWHT13]. Often, there is consistent use of SL devices for iteratively improving prediction correctness of the classifiers across the two domains, such as *bagging* the various trained classifiers on subsets of data to smooth them into a unified classifier, and *boosting* the weight of

observations during the estimation phase or when calculating the classifier's prediction correctness measure [JWHT13].

2.3.1 Prediction of Events Methodologies

This section describes the class of methodologies under consideration to investigate and build approaches in order to implement LP.

Regression. To fit the data under analysis in a single set of axes and build a multiple regression model requires our dimensions of the data to be fully ordered in terms of the values on the axis. The *query* dimension includes cuboid elements, which can only be partially ordered [Jud16]. Alternatively, building regression models for each cuboid (categorical data in logistic regression [She09b]) will allow the use of prediction intervals. The complexity of implementing the regression paradigm while satisfying the near-real time constraint is challenging for an initial set of approaches in this thesis' context, since it requires automation of training, variable selection [She09c], and validation [She09a, FEH15a] of the model given concept drift. For the validation requirement, the models constructed using regression methodology must pass a validation phase prior to inference. In this phase, the assumptions by regression must be affirmed or the variables must be transformed iteratively (to a sensible degree) to achieve affirmation. For parameterized models, heavy human input is required, as detailed in all the texts we cited previously. Should a regression model not meet the necessary assumptions in [She09a, FEH15a], then it requires human expertise to assess and improve the model in a parameterized setting. In a nonparametric setting with relaxation of linearity between response and predictors to be a smooth relation, the lack of parametric structure does not alleviate computational cost or the time required to analyze and react to the diagnostic (Q-Q) plots [She09a]; the nonparametric setting enables a degree of flexibility, however, that is not available in the parametric setting [JWHT13].

Furthermore, models with varying covariance structures [She09d] are likely in analyzing data for this application (Section 4.3). If the granularity of analysis is at the user level, then stating covariance structures per user per time scale can be a way of achieving personalization. Also, the logistic regression [She09b] and its variation

binary logistic models are good candidates. However, the difficulties with regression models discussed previously remain despite the advantages offered by the various models.

Altogether, regression modelling is valuable when some permanent structure of the data-generating process can be discovered. Since the data processes' observations are pooled together according to the user blind design simplification, the structure is not expected to stay relevant due to the many changes in individual user behaviours underlying the aggregate behaviour.

Next, a methodology to address **seasonality and trend-cycle estimations** in near-real time appears in detail in the text [DB16], making use of forecasting and predictive capabilities of regression. The other methodologies cited in the text, many of which are ARIMA-based, are used by major statistical agencies in North America for socioeconomic and natural science time series. The cycles which the system in scope considers for analysis are fixed intervals and, if it estimates the cycles, they can overlap multiple fixed intervals (cycles of time scales will not necessarily match up to cycles of the time series). This adds complexity to an approach, and is considered unnecessary at this stage prior to exhausting simpler approaches in experimentation. The categorical treatment of the cuboid dimension requires many models (a time series per cuboid with or without interactions between these series), which are elaborate and complex. In addition, the seasonal adjustment component is not primarily needed since a cuboid in use for a pattern of user(s) following seasonality can also be in use by user actions outside of this pattern (recall the pooling of all user actions into a single time series). If the granularity of the analysis is the user or user group, then a seasonal component may be part of modelling and is dependent on reliable user profiling.

Otherwise, when the analyst has discrete data points which describe a smooth variation, then, in principle, the data to model are a function describing this *continuous* variation, though discretely observed. In this thesis, from this chapter forward, the count of events is our discrete value and the change in frequency over time describes a curve rather than just points. Each variation of a cuboid's frequency of use over some time scale is a functional observation, and **functional data analysis** is the study of these situations [RS05a].

For functional data that we expect to exhibit different behaviours in various periods, the notion of a model per each of these periods is shown to be more informative than applying a single estimate of the curve's shape for the whole data process [Sch02, RS05b, Ram02]. The modelling of data periodically is often associated with Fourier analysis [RS05b], and the generalization of signals over various periods to expose underlying structure is associated with multi-scale analysis when using wavelets instead of Fourier bases. While the "...Fourier series [is] the traditional basis of choice for long time series, but newer techniques such as B-splines and wavelets can match and even exceed ..." the "... computational efficiency" offered by the Fourier series [RS05b]. Furthermore, "[t]he wavelet expansion of a function f gives a multi-resolution analysis... [t]hus wavelets provide a systematic sequence of degrees of locality. Note that multi-scale is often referred to as multi-resolution in the literature. In contrast to Fourier series, wavelet expansions cope well with discontinuities or rapid changes in behaviour... means that it is often reasonable to assume that an observed function is well approximated by an economical wavelet expansion ... even if it displays sharp local features" [RS05b].

However, the purpose of Fourier bases, B-spline, and wavelets is to smooth functions: "[t]he use of basis functions is a computational device well adapted to storing information about functions, and gives us the flexibility that we need combined with the computational power to fit even hundreds of thousands of data points" [RS05b]. Wavelets do not offer a prediction mechanism, as do regression and Bayesian (as other probability distribution-based) methods which offer prediction and credibility intervals, and otherwise offer smoothing and de-noising functionality [Sch02]. The time series analyzed in this work is expected to vary over different periods; as such, it is not feasible in the initial approaches to spend time discovering underlying structure to then utilize other prediction methodologies, for which the structure becomes irrelevant soon after discovery.

2.3.2 Multi-Scale Analysis

Time series are approached from a *time domain* perspective in which the relations are modelled in accordance to a point in time; time is therefore always a feature in this domain. However arranging observations in reference to time can be limiting to the

modelling phase where tendencies in the data process require missing features to be included, so that a relation between the response and time could be captured, where otherwise no relation exists. Symptoms of the prior are evident at the moment of prediction along a time variable which yields inconsistent correctness of the prediction or is consistently poor. However, if predictions are made without variance of time, then the correctness of the yields are more accurate than prior (with respect to a prediction interval). A challenge is that removing time as a feature alleviates an impasse in modelling, however it renders the predictions less useable if knowing the future value of the response is required in the context of time. Instead, if there is a way to maintain time as feature yet partially exclude it during the estimation of patterns and relations in the remaining features' data, then the predictions can be made in the context of time. For various signal processing applications such as pattern detection in images and reducing the amount of data needed to store images the interested reader may consult [RJ09]. In addition, applications in economics are discussed in [Sch02, Ram02].

The time domain denoted as D_t and frequency domain denoted as D_{Hz} are briefly discussed in this section. A **stationary** time series is one whose statistical properties remain *idle*, which is to maintain a constant value and exhibit no change or concept drift. Such constant properties include mean, variance, correlation, auto regressive, and others. The opposite of this scenario is a **non-stationary** time series.

In D_t the variation of amplitude of the signal with time is recorded. Recording events for a 20-minute period is a D_t approach.

In D_{Hz}, over the entire time period of recording, the goal is to record the peaks in D_t and represent the tallies as a data point plotted in D_{Hz}. This is the number of times each unique event possible in the sample space has occurred during the total period of observation. In other words, plotting types of variation in the period, rather than noting all the *time* values for each occurrence, is a form of aggregation focusing on the types of variation and their frequency rather than the exact time they occurred [Sch02].

For instance, the frequency of a sinusoidal $f(x)$ is the number of cycles in a **period**: $Hz = \frac{1}{period}$. It is typical to work in D_t since our inferences must be applicable to this domain; however, the analysis to yield these inferences is not constrained to

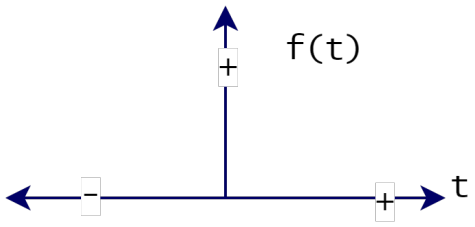


Figure 2.4: D_t is depicted, the axis of time (t) and the values of the series over time are recorded along axis $f(t)$.

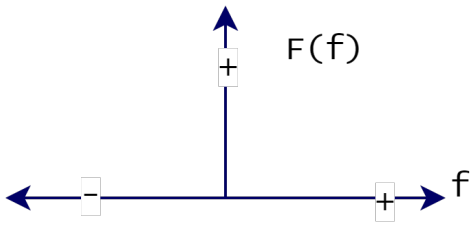


Figure 2.5: D_{Hz} is depicted, the possible values of the series are along the f axis and their frequency values in a period specified are recorded along the $F(f)$ axis.

just D_t. The tendencies of the data are analyzed in both D_t and D_{Hz}. Given the stationary series $f(t) = 1$, this horizontal line with a constant slope in D_t amounts to a singular point in D_{Hz} since there is no variation in f over t; figures 2.6, 2.7, 2.8, 2.9 are identical to those in [Sch02]. In D_{Hz}, the concern is with the **scale** and not location, because the location is closely approximated to be in the period (an interval of values along the **time** t axis in D_t).

At a high level, the need for multi-scale analysis is driven by the desire to focus on the variation of the data by reducing a dimension which adds complexity to the modelling process, namely time. Decision making is dependent on the D_t to discern between **long-term and short-term behaviour**. Individual time scales enable the

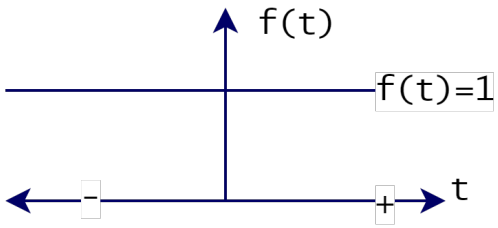


Figure 2.6: A stationary time series in D_t is depicted.

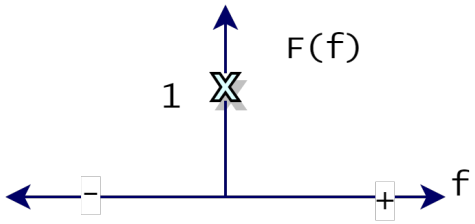


Figure 2.7: The series in Figure 2.6 is represented in D_{Hz} here. In D_{Hz} the series' variation is represented as $F(f) = \delta(f)$.

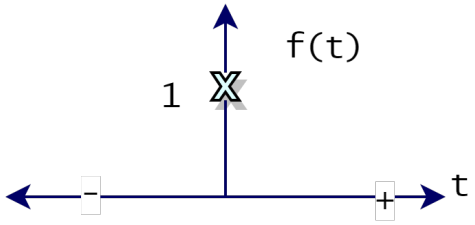


Figure 2.8: A stationary time series given by $f(0) = 1$ *zero otherwise* in D_t is depicted.

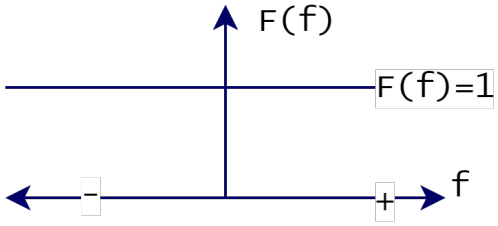


Figure 2.9: The series in Figure 2.8 is represented in D_{Hz} here. In D_{Hz} the series' variation is represented as $F(f) = \delta(f)$.

forecasting mechanism to work with more detailed information than the "crowded" full signal. As stated in [Ram02], "Smoothness is sometimes an important property for a [model]... if the...[$f(t)$] ...is thought to be smooth, then smoothness is a desirable property" It is not possible to achieve high resolution in both D_{Hz} and D_t. Overall, "high" resolution refers to frequent sampling of data over a feature of choice, in our case the focus is *time* and as a result we may capture detailed variation in the data; similar concept as having more decimal points to represent a real number than only a few.

The common model $y_t = f_t + \epsilon_t$, where y_t is an observed element of the time series Y, $f_t \in F$ is the actual signal (the underlying data-generating process in the stochastic approach), and ϵ_t is the noise. The general tendency of analyses makes the unstated assumption that F is smooth and ϵ_t is the non-smooth variation, making F less smooth. Using these assumptions, we can approximate f_t by the use of some form of averaging over the noise [Ram02]. When smoothing is done in the situation that F is {non-smooth, contains discontinuities, regime shifts or concept drifts}, smoothing is counter productive since the very structure of the signal will be lost or at least distorted [Ram02]. In the case that F is non-smooth, de-noising is needed instead of smoothing. As stated in [Ram02], one should avoid "thoughtless use of averaging procedures..." which mask the "signal characteristics being sought".

Next, the topic of **cross-scale complexities** is introduced. It may be the case that at the longest scales F is smooth, however, at periods representing smaller business cycles (see primary application in Section 4.3, the series F is non-smooth, and at even smaller scales the series is both smooth and periodically non-smooth [Ram02]. Thus, one might consider different optimal model properties at different time scales.

A "good fit" to a time series is not indicative of the quality of forecasting that can be achieved by the model yielding this "good fit." The fit of a time series over historical data periods can be "very good", and the ability to "forecast past the very near future is very bad" [Ram02].

2.4 Concept Drift

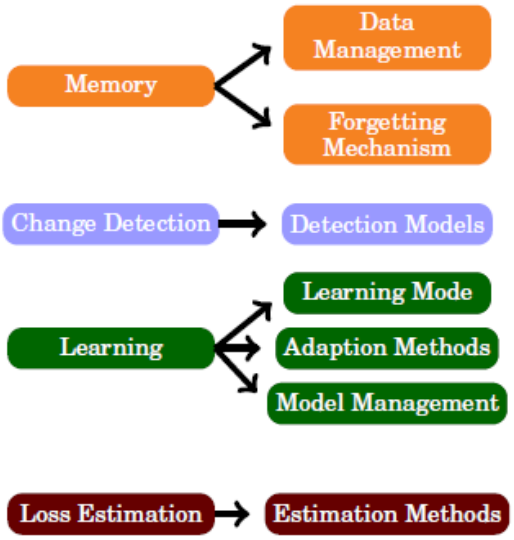
The change in user behaviour observed through their query submission is better specified using the convention consolidated in [GZB+13]— referred to consistently in this section—in which the datasets introduced in Chapter 4 explore a mixture of changes in user behaviours exhibited (real drift) and the samples of these behaviours witnessed over time (virtual drift). The approaches in Chapter 3 aim to adapt to a sampling drift among others, since the databases housing the data marts are deterministic in the mapping of query (X) to cuboids(Y),rather than real drift which would involve observing changes in the DM schema or varying cuboids being called to answer identical queries (syntax-wise). It is possible that the traffic of queries (X) can change over some period of time, and that this change can affect Y and leave $P(y|X)$ unaffected; the relation between predictors and the outcome is unchanged, however, the underlying distribution of the data-generating process (the users' submission of queries) is expected to change. In this thesis, Y is the histogram providing the number of times a cuboid (or its subset w.r.t number of rows in the database) is pulled into main memory for computing answers to a user's query. Referring to Figure 2 of [GZB+13], subsequent chapters of this thesis make no assumption on the type of drift and, in fact, the data set synthesis specified in Section 4.3 intentionally mixes abrupt, incremental, and recurring drift with some unintentional instances of gradual and outlying behaviour.

Following the requirements set out by the authors of [GZB+13] to specify the approaches in Chapter 3, the authors address: (1) detection of concept drift in a timely manner, (2) adaptation to changes, and (3) implementation of the decision given the prior steps in time for the desired effect on the next period of time in interest of analysis within the constraints of a storage limit. The approaches in Chapter 3 do not focus on distinguishing drifts (true positive detection) from noise (false positive detection); instead, they implicitly address noise by using a histogram of outcomes Y since less prominent instances are less likely to be predicted and inconsistent instances are less likely to trigger a false positive detection—though explicit noise handling is covered in Section 6.1. The high-level procedure outlined in Section 2.4 of [GZB+13] is applied in the proposed approaches specified in Chapter 3, with the iterative sequence of actions to *predict* from historical data, *diagnose* the correctness of predictions, and

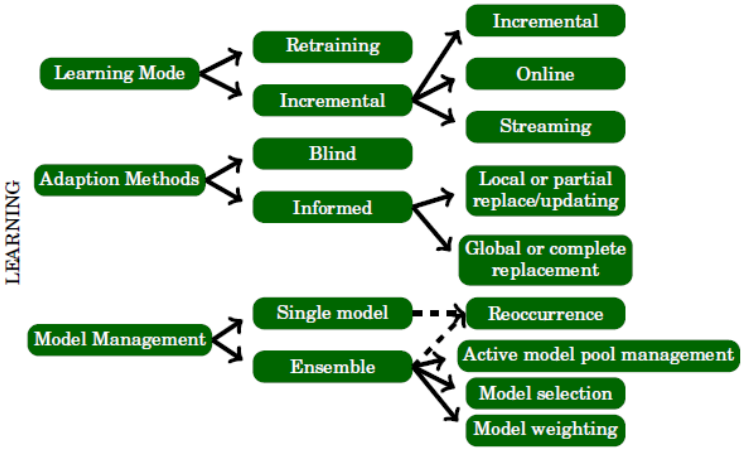
update the prediction models.

As the update step occurs in each iteration in order to add and/or remove observations within a sampling window, the determination of the most appropriate sliding window to apply is addressed. A **window of data** is a bounded interval of time in which data is collected in sequence and analyzed, the window typically shifts along the time dimension to capture new data as it becomes available while leaving older data out of scope from analysis in the next iteration. The iterative shift of the window of data may be done using a fixed window size (the amount of time a window covers) or a variably sized window where the size changes per algorithmic conditions. Between the two classes presented in [GZB+13], **variable sized windows are applied in this work's framework** since they allow for consideration of two situations that are certain to occur and they should directly affect the decision of which observations to include in the training set and which to *forget*. The considerations that the authors in the concept drift survey mention are: "...when the data is noisy or concepts reoccur, recency of data does not mean relevance. Moreover, if slow change lasts longer than the window size, windowing may fail as well" [GZB+13]. Hence, the window size adjustments are made by a heuristic detection or lengthened (shrunk) by a factor of the change amplitude, as covered in Chapter 3. Accumulating more observations to smooth out consistent behaviours (concept) is leveraged by an increasing window of samples to carry out LP. Forgetting samples suspected of irrelevancy by shrinking the window size at the later points in time (for this thesis' application) is beneficial to carry out adaption specified in the next chapter. Selecting whether to forget *abruptly* or *incrementally* is a topic explored in Chapter 4. Interested readers are referred to [GZB+13] for a detailed description of windows of data, nuances of types of concept drift and other terminology referred to in this section.

For the use case of this thesis, the primary focus is the change detection (see Figure 2.10a) and the areas where an adaption strategy can make changes to the system under investigation (highlighted in Figure 2.10b). The purpose of explicit change detection is to extract data points from an inflow of data over time associated with a change in this *data-generating process* (phenomenon in the real world which leaves behind observable evidence in the form of the data the system collects). Sometimes, there is less interest in the individual data points (tuples) and instead we care more about the interval of time in which the change occurred; this too can be identified



(a) Four Modules of Adaption



(b) Learning Capabilities

Figure 2.10: The modules are functional areas which contribute to detection and adaption strategies; their involvement in the *mix* implemented in daily operations is highly context driven. The learning is governed by the methods which aim to keep up with change and make valid inferences for a set of conditions at some point in time; a change adaption strategy will likely form an ensemble of these. This diagram is obtained from [GZB+13].

with explicit change detection methods. As the authors of [GZB+13] cover many examples of detection methods, only a few examples will be reviewed in this section in order to focus on their contrasts while representing the different lines of thinking on this topic. As per the taxonomy consolidated in [GZB+13], there are also four dimensions (no relation of dimensional modelling in the database context) in which methods of change detection can be plotted as being members of varying degrees, providing better visibility of their contrasts. The four dimensions are: (1) sequential analysis, (2) control charts, (3) monitoring (statistical distributive) differences, and (4) heuristic.

The cumulative sum (CUSUM) of the values over time can reveal shifts in the data-generating process distribution. The values are measures of a quality characteristic at time t, or more generally $x \in t$, where t is a grouping of measures [Pag54]. The function specifying the detection of change under CUSUM is:

$$g_t = max(0, g_{t-1} + (x_t - \delta)) : g_0 = 0 \tag{2.2}$$

Where δ is the tolerance for change. A change detection flag is raised when $g_t > \lambda$ where λ is a user-defined threshold, and $g_t = 0$ for the next iteration of evaluating g_t. The measure cumulatively summed could be: the mean value of the process over time, median value, other aggregations which would be a defining *characteristic* of the data-generating process of interest to monitor if a when a change occurs. In the next chapter, the quality characteristic to monitor will be discussed at length as the percent correct predictions (x_t) per cycle (t). As a result of CUSUM, control charts can be generated as in 2.11, for a visual inspection of change over time (or sequential groupings of the data process aggregated in groups t); extensive variations can also be found of this general method adapted in many industrial and electrical systems applications. While this method is primarily involves sequential analysis, it yields control charts as well.

Next, the methods in (2) are based on known distributions, such as using Bernoulli trials when events in a sequence follow the binomial distribution as this distribution is well fit to track the error rate of a process over time. There is an expected (and tolerable) error rate in the predictions however, if this is exceeded than a change is *detected* as a result. The difference in this *dimension (2)* is the reliance on an assumed

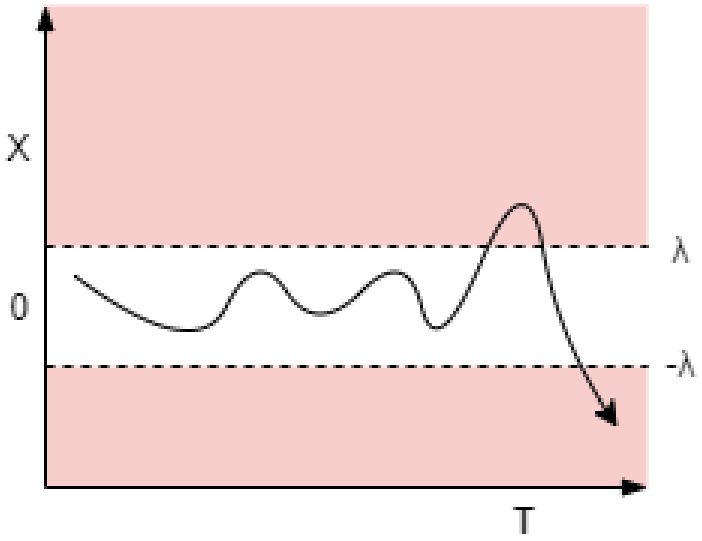


Figure 2.11: Generalized control chart for CUSUM computed on a data process. The two shaded (red) areas are "out of control" zones, as per the terminology in [GZB+13]. These denote a change detection for every instance (interval) the quality characteristic $x > \lambda \vee x < -\lambda$.

distribution for detecting *enough deviation* compared to a threshold relative to this distribution, as opposed to relative to the process in (1) and the example of (2) in Figure 2.11. Crucially in statistical process control charts, the error rate is expected to decrease as a model learns from an increasing set of training instances; otherwise, an increase in error rate indicates the distribution of the data observed has changed beyond what the model has predicted. Visually, methods belonging to (2) contrast Figure 2.11, where steep upward sloping of X over t is deemed *normal* until it exceeds the absolute threshold λ; the prolonged (measured by $\Delta\sigma$ of the error rate's binomial distribution) climbing of error rate would be the triggering event for a *change detected flag* instead of the hard threshold.

Onto (3)the methods aim to compare the distributions of the data or x_t across two different *windows* of historic data; usually one acts as a reference of desired variance in the measure x while the second represents current measures used to draw inferences (descriptive statistics, predictions, etc.). The reference window is fixed whereas the current frame is a sliding window of data (often along the time axis, making the process a *data stream*). Two of these methods are listed below:

- (weighted) Kullback-Leibler divergence: a measure of the difference between two distributions (one per window of data). The measure of difference can then be tested using the hypotheses: H_1 : the distributions are two different data processes, and H_0 : there is no significant difference in the two distributions
- Hoeffding inequality-based algorithms as demonstrated in the ADWIN algorithm, whereby the harmonic means of the X measures in the two windows are compared to determine if the change in the means is "large enough and distinct enough" [GZB+13]. This inequality is applied in more recent change detection algorithms.

In Chapter 3, three change detection methods are to be used in combination with three predictive models in which each pairing forms an *approach*, and these approaches form the ensemble. Relating back to the overview of the learning module of a change detection/adaption strategy in Figure 2.10b, the ensemble is managed via *model selection* because the approaches within are intended to have little overlap in their inferences and tend to function favourably in different situations, thus only one approach will determine the system's actions at a given time. Within the ensemble, the adaption is $informed$ and some methods will trigger the predictive model to *retrain* after a change is detected, while others will rely on *incremental* adaption. The difference in these processes is the need to re-estimate the predictive model parameters from initialization or whether an *updating* function is available to make changes to initially estimated parameters; the impact is a vast difference in runtime between the two ways. The *memory* module shown in orange in Figure 2.10a is controlled by the adaption methods to be specified in the next chapter.

2.5 Conclusion

In conclusion, the challenges to be addressed in the framework proposed in Chapter 3 are *threefold*. First, we require the user input to be limited to the hyper parameters of the framework only and eliminate time delay related to human adjustments made to model inputs, contrasting the methods discussed in Section 2.2.1. In turn, this limits the **human bias** (heuristic) in predictive models as it is not within reach

of an automated change adaption strategy driven by functions on factors of change. User profiles forced onto the system or query-specific grammar to describe user patterns are difficult to integrate with generic change adaption methods which take their cues from change detection methods in sequential analysis and monitoring statistical differences (as in Section 2.4). The generality of our framework allows for it to be applied to domains other than those explored in Chapter 4. *Second*, we attempt to build an **ensemble of predictive models** which handle partially ordered sets of items (lattice of elements) efficiently in order to achieve near real-time forecasts. *Third*, to regain the loss in correct predictions per cycle of time we employed methods that **detect changes** leading to degraded predictive correctness and **adapt the predictive models**.

Chapter 3

Lattice Prediction under Change Detection and Adaption Framework

Implementing a version of the multi-scale analysis in combination with prediction and change detection algorithms, leads to change adaption strategies and the formation of the LP-CDA framework detailed in this chapter. Similar to the effect of drill-down and roll-up queries within the OLAP context, the ability to zoom in and out of a data signal or a time series can be achieved analytically by applying multi-scale analysis introduced in 2.3.2. Of particular interest is the ability to zoom in on shorter time intervals for immediate and short trends in a time series and simultaneously zoom out to lower scale of time for the over-arching trend. The framework specified in the next sections applies a dedicated predictive model for each scale of time considered and aims to consolidate the predictions into one actionable materialized set of cuboids. Given the nature of the data under investigation detailed in Chapter 4, the daily scale would have us analyze the previous days' cuboid demand to predict the demand for the next day's cycle. Time scales worth analyzing include any intervals of time within which the prediction computations can finish, and intervals for which the DBMS expert thinks a materialization update is necessary. If a **scale of time** is more granular than the time required to compute the predicted cuboid set and act on it, then this scale is too granular to implement. Those tasked to manage the DBMS are aware of some of its nuances about general user tendencies and some technical

details on the execution of queries in the particular system. In this case, domain expertise is useful in deciding on the initial frequency of predicting and materializing so as to materialize well in advance, though not too frequent that the DBMS is left with significantly less resource usage time to compute user queries. Also, inspecting a sample of query logs can reveal the general duration of the query classes such as what it means to be a short-, medium-, and long-running query — relevant to the largest sensible time scale. As an example, suppose 77, 71, 67 hours is the group of longest running queries for some months' data, then long queries can be said to take three days to run, implying a time scale of *every 3 days* for which to predict. In contrast, despite a higher frequency of queries which run for a couple of hours, the DBMS expert may limit materialization to the daily scale so as to not take away from computing resources of the user queries, which are attributed to frequent I/O's from swapping and constructing views.

There are two roles for the time scales: the first role is to determine the training data window size, and the second role is reserved for a single scale to specify the intervals on which predictions are acted on to update the materialized view. On the latter role, there will be a singular scale for predictive models' results to be pooled into a materialized set of cuboids; this scale is denoted as the critical scale. At each cycle of the critical scale, the materialized view can change, and the set of time scales for which to predict includes this scale. While there can be multiple and possibly conflicting predictions across various scales, there will be a single time point to manifest these predictions into the materialized view. For each of the time scales for which to predict, the system allocates main memory, and their sum is the total memory cost of the materialized view.

3.1 Framework at a Glance

The framework evaluates the correctness of the predictions made by contrasting the cuboids in actual demand during the next time period, and those specified undergo smart materialization as in [AV16]; this is the key performance indicator (*kpi*). Then, a detection algorithm measures the amount of change in this *kpi* at the end of the *critical* time scale (the day case in this thesis' context). If enough change occurred,

then an adaption procedure initiates, modifying the memory allocated to each time scale's prediction method and this method's hyper parameters; the adaption is proportional to the *kpi* deviation from a constant threshold at the time of the detected change.

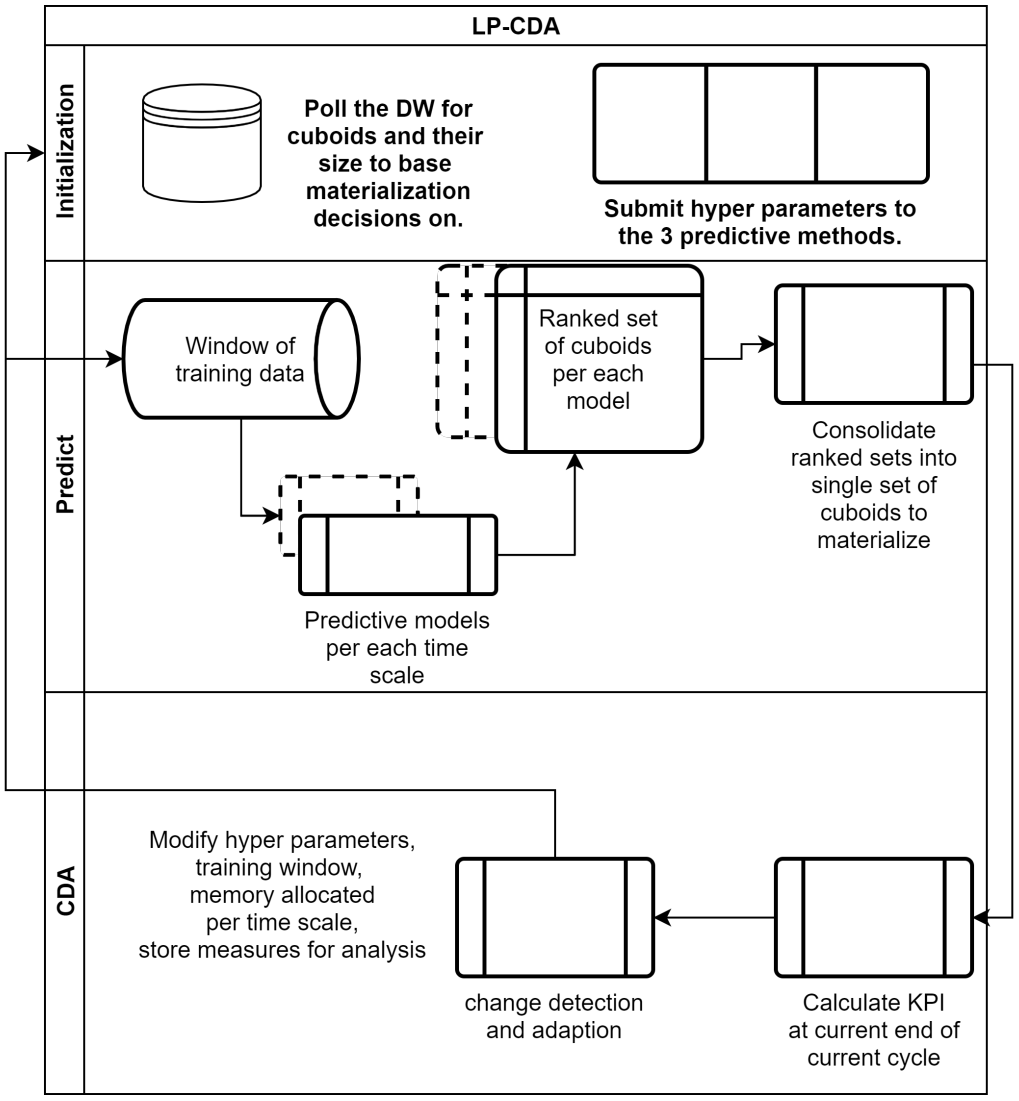


Figure 3.1: Overview of the LP-CDA framework.

For the purpose of validating the methodology (in Chapter 4) and providing evidence to distinguish methods for real-world applications (Chapter 5), the implementation of the LP-CDA framework at a high level is described as follows:

1. Initialization of constants and prediction method hyper parameters. These include:
    a) The time scales on which to predict, their order of priority; the highest being referred to as the critical scale. Then the amount of memory reserved for each scale:
    b) A key value pairing of the DW cuboids and their size in memory units.
    c) The initial hyper parameters for prediction methods.
2. Read in historic data as old as the window of time indicates for training data; this is dynamic.
3. For each cycle in the critical scale, do:
    a) Predict the next cycle's cuboid demand, produce a ranked list of cuboids. If it is the end of other scales' cycle, then run the LP method on these cycles' training data as well.
    b) Produce a materialized set by including a cuboid in order of predicated rank so long as there is memory space left.
    c) Compute the *kpi* and execute the change detection and adaption (CDA) procedure, if a detection flag is raised the change adaption procedure is carried out.

The flow of logic in the overview of the framework above was designed primarily for investigation into the performance of methods in various combinations. Particularly, steps 3a and 3b reflect the procedure undertaken at the start of a day, and 3c at the end of the same day. However, it was computationally advantageous to keep all these actions related to day i in the i^{th} iteration of the simulation loop; see Figure 3.1 for a visual breakdown. The third point in the overview is specified in Algorithm 1 as implemented to conduct the experimentation, where the [ ] denotes a list or array (a data structure linking elements in some order), and () denotes a function call with its parameters enclosed within the parentheses. Using [ ] brackets is synonymous with using subscripts; however, they are much more readable since they avoid the

reduction in font size.

The time scales introduced at the start of this chapter control the flow of the predictions that are implemented in the looping and conditional statements in Algorithm 1. In the context of the data sets detailed in Section 4.3, the LP-CDA framework is specified with two scales of prediction, daily and weekly, since the events of interest are tallied on the hourly scale. Then, a priority of these time scales is chosen to be the daily predictions followed by the weekly predictions, meaning at the end of each day, all that day's events are collected and become the most recent addition to the training set. Similarly, at the end of each week all of its events are added to the weekly model's training set. Since there are two prediction models that are meant to instruct the cuboids that are members of the materialized view, these independent models may contradict each other as their training sets differ. The differences are due to the scale of time to which a model is constrained which controls the rate of relative increase of the training set, and the change adaption procedure which controls the shrinking of the training set. Of note, the LP method called on line 12 of Algorithm 1 at each iteration of the outermost loop simulates the passing of time at the *smallest scale of interest* (i.e., day scale), whereas the weekly scale LP instance is called once every 7 days on line 15.

The reason that only daily and weekly time scales are of interest is data dependent. The context introduced in Chapter 1 and solidified in the data sets specified in Section 4.3 lead to the following observations:

- OLAP queries of intense aggregation are likely to run for multiple hours.
- The meta data is accurate for events within an hour's interval of time; this is the smallest granularity for which data is available.
- There is a concern of using up too much RAM and CPU in conducting the prediction analysis and implementing the results of this analysis using smart materialization too frequently.
- *Domain knowledge* indicates that certain topics of analysis for groups of people often occur for a number of days less than a week; this is inferred from visual inspection of aggregated data in Section 4.3.

Thus, we interpret the above to indicate that predictions made at the daily and weekly scale are sufficient, and that the day scale takes on the role of the *critical scale* as a consequence of the fourth point above. The third point implies that the hourly scale is too frequent in terms of cost (computing resources time) to benefit (up-to-date materialized view) ratio. Recalling the multi-scale modelling consideration discussed in Section 2.3.2, the weekly scale's training set serves as the base signal and the daily scale is a detailed higher resolution signal; both signals describe the same data-generating process.

Another aspect of the time scales is their priority, which determines the scale's model that can add elements to the materialized set first; in the full sense of the term *set*, there can be no duplicate elements. For example, on the last day of any given week, both lines 12 and 15 of the algorithm execute; however, the highest priority scale's model has the first opportunity to add a cuboid ranked highly to the materialized set (line 21) as a consequence of line 6. If a cuboid is estimated to be in high demand under both scales' models, and there is sufficient available space as on line 21 of the algorithm in the daily scale's partition of the materialized set, then it cannot be also added as a result of the weekly prediction. As a consequence, if the same cuboid prior drops in demand in the day scale's model the following cycle, whether this cuboid joins the materialized set depends on the available memory allocated for the weekly scale

In addition, to allow for populating the materialized set, only the critical scale is affected by the CDA procedure. Specifically, the following computation is performed at the end of each critical scale cycle.

$$kpi\ at\ time\ t = \frac{instances\ (cuboids \in smart\ materialized\ view\ \cap\ total\ required)}{instances\ (distinct\ cuboids\ required\ by\ DBMS)} \tag{3.1}$$

It then follows that the CDA() procedure in the case of a detected change will shrink the critical scale's training window from the oldest cycle's data by a factor of the difference in the kpi and user set constant threshold. Also, the same factor is applied to the LP method's hyper parameters.

The following sections specify the different methods for LP and CDA in this hybrid

Data: Table containing the column space (query q, mean μ duration in hours, standard deviation Θ in duration) for a user role.
Result: A set of vectors in the form, (q, start hour, duration, day). It then follows to replace q with the set of cuboids using a separate lookup table.

```
1  begin
2      Initialize()
3      for cycle ∈ critical_scale do
4          leftoverMemory⟵ 0
5          updateTime()
6          for scale ∈ Priority do
7              startAt ⟵ Windows[scale][0]
8              endAt ⟵ Windows[scale][1]
9              toPredict ⟵ endAt + 1
10             memoryAvailable ⟵ size[scale]
11             if scale == "day" then
12                 result[scale] ⟵ LP(scale, startAt, endAt)
13                 windows.update(scale, [startAt, toPredict])
14             else if scale="week" and current["day"]mod 7==0 and
                truncate(current["day"]/7,0)=current["week"] then
15                 result[scale] ⟵ LP(week, startAt, endAt)
16                 windows.update(scale, [startAt, toPredict])
17             if current["week"]=0 then
18                 memoryAvailable ⟵ totalAllocated
19             end
20             if result[scale] ≠ ∅ then
21                 Iterate over the ranked result[scale] list for each
                    element∉ materializedView add the element to the
                    materializedView[scale] set so long as
                    memoryAvailable − size[element] ≥ 0
22             end
23             leftoverMemory ⟵ memoryAvailable
24         end
25         Iterate over the ranked result[critical_scale] list for each
            element∉ materializedView add the element to the
            materializedView[critical_scale] set so long as
            memoryAvailable − size[element] ≥ 0
26         kpi ⟵ computePerformance(critical_scale)
27         change ⟵ 0
28         CDA()
29     end
30     test
31 end
```

Algorithm 1: LP-CDA

ensemble voting scheme in which the winner takes all. In Chapter 5, an analysis is presented that compares LP methods and also concludes if CDA is necessary.

3.2 Bayesian Prediction from a Beta-Binomial Distribution

This first prediction method is simpler than those that follow in this chapter; however, a Bayesian prediction on the Beta-Binomial (BBB) may be on par regarding the prediction correctness with the alternative methods since statistical inference with an underlying binomial distribution has been found to be effective in two change detection methods discussed later in Section 3.5.1. This method is considered to be the most interpretable prediction algorithm of all considered here; its advantage is that domain expertise can be most effectively *injected* into the model via the hyper parameters to skew the distribution towards the *real world* behaviour, as opposed to a black box approach using neural networks or an intermediary between interpretability and black box algorithms, the frequent pattern mining introduced in Section 3.3. The term *interpretability* refers to the ability to explain how a predictive model is fit and the decision-making process executed to arrive at a result [JWHT13]; for instance, the effects of data features on an outcome predicted is more difficult to explain if that model is A Neural Network (ANN) instead of a more accessible/descriptive model such as BBB. The domain of predicting cuboid demand over time is similar to the prediction of *number of home runs in a game per season for a given baseball player*, which has been addressed using BBB, as covered in [Fos15] and other similar examples. In the sports context, a beta-binomial distribution is constructed after each game whereby the hyper parameters are *updated* with new information over time; in this way, the consistency (or lack thereof) of the home runs per game is reflective in the narrowing (widening) of the bell curve around the mean. Since the input matrix to the LP-CDA framework contains the number of times a cuboid is required per time unit, and that the goal is to materialize a set for a collective of users despite their individual query patterns, the cuboids are treated as independent *players* on the field. Dependency among the cuboids is not explicitly modelled in this predictive model, as it can be implicitly inferred in the ANN model later introduced. Two considerations

in the sporting context which run parallel to the cuboid context are described below:

- The event of scoring a home run is considered dependent on the previous distribution of this event, and independent of other players' actions, since we consider the effect to other factors to be embedded in the historic distribution of past games' home runs for a particular player.
- The games are independent of each other, meaning the cycles of time are independent of each other.

Furthermore, consider the two contexts described as vectors denoted *(event, measure, cycle of time)* in order to highlight their similarities: for baseball it is

$$(player_x, \#home\ runs, game_t),$$

and in this thesis' application it is

$$(cuboid_x, \#of\ times\ required, cycle_t).$$

Why not rely on counting the number of times a cuboid is required and rank them by highest count as a way to *predict*? Why not use the normal distribution to describe the demand of a cuboid? Why not use a moving average or exponential smoothing function as shown in Table 3.1?

First, the distribution is not assumed to be constant in both contexts, and thus a geometric average will not capture skewness over time [Wel98]. Other moving averages and exponential smoothing may be considered, however, they require that the pace of change be constant, as indicated by the parameters in the schemes in Table 3.1. Various averaging schemes were explored on small series of measures for a few cuboids from the smaller data set (namely set 1 in Chapter 4) in order to evaluate their application in the BBB method. Since the skewness of measurements over time can be quite drastic, the first three schemes in Table 3.1 did not forecast the next measure close to the next value in the sequence. The double moving average requires a minimum of eight data points in a sequence; this is a severe limiting factor as it becomes clear in Section 3.5 that often it is possible to be left with only two points in the sequence (two time periods on which to train). As a result, the exponential

smoothing was most tolerant in handling small sequences and delivered forecasts closer to the actual measures in the sequence through early experimentation. In both contexts discussed previously, the rate of change is expected to differ; indeed, in early experimentation, the averaging schemes would not perform well relative to the three prediction methods applied to *data set 1* in Section 4.3. One feature of the beta-binomial distribution is its narrowing or widening of the bell curve proportional to the frequency of the measurement seen over time, which differs from using the normal distribution since it is an updatable distribution, and does not forget its previous shape. Using a normal distribution would only account for the most recent variation and mean of the measure constrained to a fixed shape, being non-representative of variable change in the distribution over time.

The beta-binomial distribution is defined using the form on page 85 of [SM09] as:

$$p|p{\sim}Beta(\alpha,\beta),\ \hat{\alpha}=\alpha_0+k,\ \hat{\beta}=\beta_0+n-k$$
$$Pr(p|n,k)=Beta(\alpha_0+k,\beta_0+n-k) \quad (3.2)$$

for n trials with k successes. The prediction posterior is $Pr(p|n,k)$, as in equation 3.2. The probability of having exactly j successes in the next m trials given k successes in the prior n trials is the prediction interval with 95% credibility [BS13], denoted as $[j_1, j_2]$.

Instead of the interval, a point estimate can be used for predicting j successes in the next m trials, but through preliminary trials of four methods (Bayesian point estimate, Bayesian prediction interval, frequentist point estimate, frequentist prediction interval—infeasible due to real-time probability estimation as a prerequisite), the interval is chosen for better average case accuracy. Also, the interval enables an informed manner to break ties between events that are predicted to occur at least the same number of times, which is relevant for highly positive correlated cuboids. Suppose that two point estimates are $j = 20 \in A$ and $j = 21 \in B$; then by these measures, lattices A, B are almost equivalent in their predicted 'use' in the next cycle instance. However, with intervals, suppose the results are $[18, 22] \in A$ and $[20, 22] \in B$; then there is a tighter bound for B than A. This difference in bounds is affected by consistency (success, failures) of past n trials and the amount of history (the value of n); there is more insight into which lattice is probable despite both intervals being

subject to a 95% credibility interval.

The values of $[j_1, j_2]$ are determined to satisfy:

$$Pr(j \leq j_2|m,n,k) - Pr(j \leq j_1|m,n,k) \geq 0.95 \quad (3.3)$$

Find the tightest bound $[j_1, j_2]$ among the probabilities (P) denoted as (P, j) pairs from the cumulative mass function (cmf) of the beta-binomial distribution. For instance, given:

$$[j_w]P_w,\ [j_{w+1}]P_{w+1},\ ...,\ [j_{i-1}]P_{i-1},\ [j_i]P_i\ :$$
$$\sum_{w=1}^{i} j_w = \frac{1}{2}m(m+1)\ F(j) = P(J \leq j)\sum_{w=1}^{i} P_w \quad (3.4)$$

In equation 3.4, m retains its definition as above in this section, the w is used to denote j_w, where $w \in 1, 2, ..., m$ to avoid confusion between the j_1, j_2 mentioned thus far which are selected j among all of j_w's. Also, as per the cmf function described in [Wal13], it is defined for $-\infty < j < \infty$ (x in [Wal13]). However, in equation 3.4, we define this cmf for the domain $1 < j < m$ because our methodology will only consider positive values and m is the upper bound which is an input to equation 3.3.

Then, choose j_1 *and* j_2 such that:

- $j_1 \leq 0.025$ is the best solution (closest to $j_1 = 0.025$) of all j_i's
- $j_2 \geq 0.975$ is the best solution than all j_i's

It should be noted that the significance of credibility [BS13] is 95%, thus the tails of the distribution are computed by $1 - 0.95 = 0.05$ and $\frac{0.05}{2} = 0.025$, then the equal-tail boundaries are $0 + 0.025$ and $0.95 + 0.025$. It is preferable to compute the distribution in real time based on the updated α and β values, and to choose the j future trials (maximum of m) for which the probability satisfies the previously listed conditions.

For instance, the prediction example shown in Figure 3.2 depicts the result of equation 3.4. The process of fulfilling equation 3.3 is described next:

- Select the prediction interval for j future successes in the next m trials as $[j_2, j_2]$ to have probability $P(j_2) - P(j_1) \geq 0.95$. By inspection of the computed equation 3.4 shown in Figure 3.2, $0.9886 - 0.0292 \geq 0.95$ is the tightest bound yielding a prediction of lattice $(A) = [j_1 = 10, j_2 = 19]$.
- Thus, the chosen lattice A is predicted to occur 10 to 19 times in the next 20 trials in the next cycle of the scale from which the prediction inputs were drawn.

```
> library(extraDistr)
> n <- 100
> k <- 70
> m <- 20
> alpha <- 1
> beta  <- 1
> bayesian_point_estimate <- m * (alpha + k)/(alpha + beta + n)
> jvec <- seq(0, m-1, by = 1)
> probabilities <- pbbinom(jvec, m, alpha = alpha + k, beta = beta + n - k)
> probabilities
 [1] 1.335244e-09 3.925617e-08 5.686014e-07 5.398876e-06 3.772061e-05
 [6] 2.063557e-04 9.183707e-04 3.410423e-03 1.075618e-02 2.917888e-02
[11] 6.872028e-02 1.415124e-01 2.563000e-01 4.105894e-01 5.857286e-01
[16] 7.511380e-01 8.781487e-01 9.546188e-01 9.886056e-01 9.985556e-01
>
```

Figure 3.2: An example of a cmf from the beta-binomial distribution computed in the R programming language using the $extraDistr$ package.

Finally, the Bayesian point estimate is

$$\frac{m(\alpha + k)}{(\alpha + \beta + n)} \tag{3.5}$$

and the expectation of beta-binomial distribution is

$$E(X) = \frac{\alpha}{\alpha + \beta} \tag{3.6}$$

First, an initial α_0, β_0 needs to be defined; this definition is known as the prior in the case of Bayesian prediction. As stated in [BS13], $\alpha_0 = 0.5$, $\beta_0 = 0.5$ is the best non-informative prior. At the end of the first cycle of any scale, the system would have witnessed $Y \in Cycle_i$. To construct the prediction interval for the ith cycle of a time scale, the $\beta - binomial$ distribution from $(i-1)^{th}$ cycle is used. To this point, only prediction of one event over time has been detailed in this section; the process described previously should be repeated for all cuboids per time scale. Though cycles of different scales overlap, the larger cycle may extract a set of lattices that is not a

super-set of the set extracted by the smaller cycle. Lastly, the averaging schemes in Table 3.1 forecast the m number of total events to occur in the next cycle of times.

Scheme	**Description**
$\hat{Y}_{t+1} = \frac{t-1}{t}\hat{Y}_t + \frac{1}{t}Y_t$	$\hat{Y}_t$ is the forecast and Y_t is the tth actual observed value.
$\hat{Y}_{t+1} = \frac{t-1}{t}Y_t + \frac{1}{t}\hat{Y}_t$	Y_t influences $\hat{Y}_{t+1}$ more than $\hat{Y}_t$; the opposite is true in the above scheme.
$\hat{Y}_{t+p} = \sum_{i=0}^{k-1} w_i y_{t-(1+i)} : \sum_{i=0}^{k-1} w_i = 1, k = 4, w_0 = w_2 = \frac{1}{3}, w_1 = w_3 = \frac{1}{6}, p \succcurlyeq 1$	Symmetric alternating weighing scheme, w_i is the weight of the $(t-1)$th observation. For this instance, $k = 4$.
$\hat{Y}_{t+1} = \alpha y_t + \alpha(1-\alpha)^1 y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + ...\alpha(1-\alpha)^k y_{t-k} + (1-\alpha)^{k+1}\hat{y}_{initial} : k = 3, \alpha = 0.1, \hat{y}_{initial} = \sum_{i=0}^{\lvert\{c_1,c_2...\}\rvert}(\frac{1}{k+1}\sum_{j=0}^{k} y_j \in c_i) \bigoplus \hat{y}_{initial} = \sum_{i=0}^{\lvert\{c_1,c_2...\}\rvert}(Median(\{y_0, y_1, ..., y_k\} \in c_i))$	Exponential smoothing, $\hat{y}_{initial}$ is the average of the first $k+1$ observations from the past or it is the median of the first $k+1$ observations. Better results were obtained using the median instead. Let α be the rate of forgetting (smoothing) and $k = 3$ for example.
$\hat{y}_t = MA'_t = \frac{MT'_t}{k^2}, MT'_t = \sum_{i=1}^{k} MT_{t-i+1}, MT_t = \sum_{i=1}^{k} y_{t-i+1}, k = 4$	Double moving average.

Table 3.1: Here, k is the window size, correlating to the size of the training set and the length of time which the distribution covers. We observe a y value per cuboid per cycle of time t.

3.2.1 BBB Algorithm Specification

In this section, an implementation of the method described in Section 3.2 is specified in Algorithm 2; this is an LP method that fits in Algorithm 1. It should be

Cycle Instance:	**Beta-Binomial Parameters**
At instance $(i) = 0$	$\alpha_0 = 0.5\ ,\ \beta_0 = 0.5\ ,\ \beta \sim Binomial(\alpha_0, \beta_0)\ ,\ E(X) = \frac{\alpha_0}{\alpha_0 + \beta_0}$
At instance $i = 1$	$\alpha_1 = \alpha_0 + k\ ,\ \beta_1 = \beta_0 + n - k\ ,\ \beta \sim Binomial(\alpha_1, \beta_1)\ ,\ E(X) = \frac{\alpha_1}{\alpha_1 + \beta_1}$
At instance $i \geq 1$	$\alpha_i = \alpha_{i-1} + k\ ,\ \beta_i = \beta_{i-1} + n - k\ ,\ \beta \sim Binomial(\alpha_i, \beta_i)\ ,\ E(X) = \frac{\alpha_i}{\alpha_i + \beta_i}$

Table 3.2: The updating of the beta-binomial prior for any cycle i of the same time scale.

noted that the use of an exponential smoothing function to estimate m was introduced previously on lines 11 and 14 of Algorithm 2, with parameters $\alpha = 0.3,\ m_0 = median(m_k, m_{k-1}, ..., m_1)$. The exponential smoothing is defined in Table 3.1, and the methods collected in [Wel98] are summarized which are largely relevant today as building blocks in modern methods. In addition, the function to update the beta-binomial distribution exemplified in Table 3.2 appears on line 8 of Algorithm 2, a vector $= (\alpha_{new}, \beta_{new})$; recall that in the pseudo code () is a vector of parameters for a function whereas [] encloses an ordered assignment of values to a variable such as a list or array. Then, the $cdf(\beta - binomial)$ is calculated for a cuboid's demand with the usual parameters seen in Section 3.2 in line 21, and is provided an *object* containing the $\beta - binomial$ relevant logic. Finally, the results are gathered for each cuboid in a matrix (line 26) and then sorted on two variables (line 27); the purpose of this is to rank all events by most minimally expected (lower bound of their prediction interval) followed by smallest upper bound. That is, the most in-demand cuboid is a measure of the lower bound of its prediction interval, and the *confidence* in the entire prediction interval is a measure of its narrowness. For instance, if an interval for a cuboid is $[\frac{1}{10}m_{future}, m_{future}]$, then the prediction is that the event can occur any number of times; without sufficient evidence as to the number of times one way

or another, this prediction interval is not informative and is of little actionable value with respect to materialized views, relative to other cuboids' intervals.

3.3 Frequent Pattern Mining

The data handled by the LP-CDA framework are a time series of events, as described in the data mining text ([Han15]) in its Section 8.2; these methods can be applied to the scenario under investigation in this thesis. For instance, the challenge being addressed is to *budget* cuboids in high demand as part of a materialized view given "sequences of values or events obtained over repeated measurements of time. The values are typically measured at equal time intervals (eg., hourly, daily, weekly)" [Han15]. In [Han15], the approaches aim to *mine* (extract) patterns, whilst this thesis has framed the task at hand to be *prediction*. The difference between these ways of determining that which comes next is materially distinct in the algorithms applied.

In Section 3.2, the LP method determined the likelihood of each event in the next m cycles of a scale (period of time). With the data mining approach, and specifically *sequential frequent pattern mining*, the algorithms determine the sequences of events that are most likely to occur. It is then a point of decision as to how to rank the set of mined patterns in terms of most to least likely among the set's elements. The distinction is that mining methods will determine likeliness relative to other events in the sample space (training set), and it then follows to separately establish an ordering based on this relative frequency (in the interest of this thesis). The BBB method, however, independently calculates the likelihood of each event, and only following a ranking of these independent predictions does a notion of relative likeliness emerge.

Next, background information on sequential pattern mining to provide context for their application to the scope under investigation is set out in Chapter 1. Two aspects of the *similarity search method* [Han15] are incorporated into the LP-CDA framework: the *window* over which to normalize, and the *gap* of non-patterns between components of the frequent patterns. The former aspect is synonymous to the sliding window introduced in Section 2.4 and applied in Algorithm 1. The latter aspect handles the cases in which "*sub-sequences* match if they are the same shape, but differ due to the

Data: $scale \in day, week$ and $first, last \in \mathbb{Z}^+$
Result: A matrix with column space= $(cuboid, min, max)$ in which the last two columns hold the start and end of the prediction interval for a cuboid.

begin
  $m_{future} \longleftarrow 0$ and $m_\alpha \longleftarrow 0.3$ and $k \longleftarrow last - first + 1$
  $historicSet \longleftarrow get\_data\_set(scale, first, last)$: *matrix of /cuboids/ rows and /hours/*/days/*/weeks/ number of columns; each cell contains the number of times an event occurred.*
  $totalTraffic \longleftarrow \{\emptyset\}$ and $betaPriors \longleftarrow \{\emptyset\}$
  for $cycle \in historicSet$ **do**
    $m \longleftarrow \sum_i^{\#cuboids} historicSet_{cycle,i}$
    for $cuboid \in historicSet$ **do**
      $betaPriors \longleftarrow [\alpha_{cuboid} + historicSet_{cycle,cuboid}, \beta_{cuboid} + (m - historicSet_{cycle,cuboid})]$
    end
    $totalTraffic.append(m)$
    $m_{future} \longleftarrow m_{future} + m_\alpha((1 - m_\alpha)^k)m$
    $k \longleftarrow k - 1$
  end
  $m_{future} \longleftarrow m_{future} + ((1 - m_\alpha)^{last-first} * median(totalTraffic)$
  round m_{future} to 0 decimal places
  $j_1 \longleftarrow \{\emptyset\}$ and $j_2 \longleftarrow \{\emptyset\}$
  for $cuboid \in betaPriors$ **do**
    $\beta Binomial \longleftarrow beta.betabinom\_gen(name = "betabinom")$
    $cdf \longleftarrow \{\emptyset\}$
    for $i \in range(0, m_{future})$ **do**
      $cdf.append(\beta Binomial.cdf(i, m_{future}, betaPriors_{cuboid,\alpha}, betaPriors_{cuboid,\beta}))$
    end
    $lower \longleftarrow 0.025$ and $upper \longleftarrow 0.975$
    $j_1.append(binarySearch(cdf, lower)$'s right most result)
    $j_1.append(binarySearch(cdf, upper)$'s left most result)
    $result \longleftarrow$ join the $cuboid$,j_1,j_2 lists on their index to form the final three-column matrix, where j_1 is the minimum number of times a cuboid is predicted to occur in the next m_{future} queries, and j_2 holds the maximum predicted times.
    $result.sort$(first by j_1 descending and second by j_2 ascending)
  end
end

Algorithm 2: BBB

presence of gaps within a sequence (where once of the series may be missing some of the values that exist in the other) or differences in offsets or amplitudes" [Han15]. In his study, Han illustrates the notion of gaps as a way to tolerate variances in the same pattern in Figure 3.3, as it pertains to continuous data; this concept is visualized differently when applied to nominal data, as discussed in the next section.

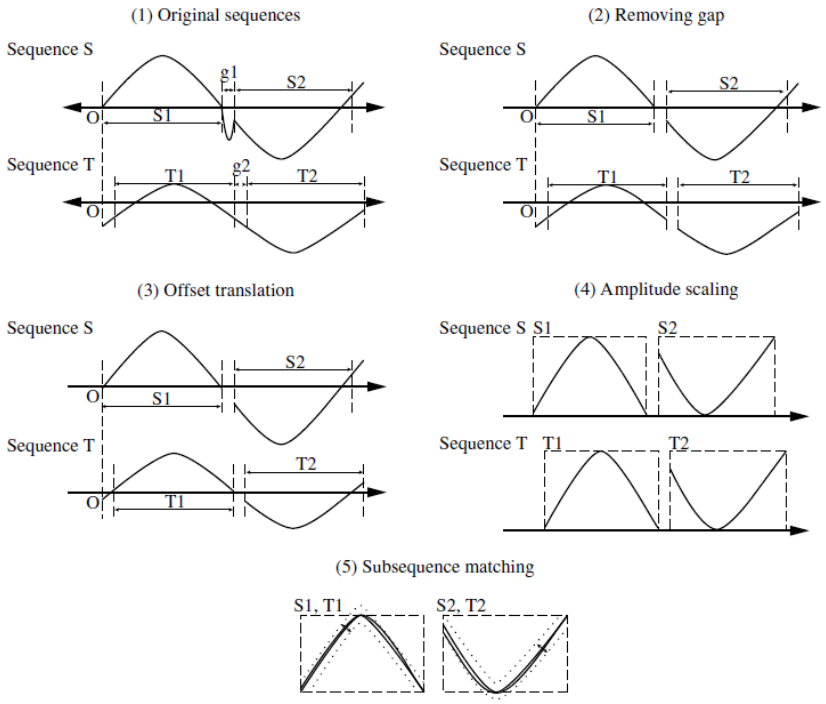


Figure 3.3: Gaps between patterns in different instances in terms of trend analysis in continuous time series data; image borrowed from Figure 8.5 of [Han15].

An algorithm which covers windowing and gaps in sequences of item sets (an ordering of categorical data) called *Gap-BIDE* is specified in [LW08], and is the focus of discussion in the next section. The data under investigation (Sections 2.2.1 and 3.2) are only partially ordered as a consequence of the algebraic lattice structure described in [Jud16], yet sequences implies a full ordering. Thus, applying alphabetical ordering on the labelling of cuboids enforces the full ordering of the dimensions (individual letter labels when compounded form a cuboid), as opposed to the default interpretation being *the set of dimensions*. This may seem to be a cosmetic transformation, however, it ensures that {ABC, AB,AC}, for instance, are the only representations the system will benefit in the proper counting of (sub)sequences which appear in the equivalent representations: {BAC, BA, CA}.

Following the descriptions in Section 8.3 in [Han15], there are two classes of algorithms which relate to objectives:

a) complete set; Spade and Prefix-Span mines every observable (sub)sequence,

b) closed set; BIDE and its variations mine sub-sequences with no super sequence under the same support.

Some definitions allow a sequence of events to be denoted as an ordering of elements, such as a vector $s_1 = (ABC, DE)$ being a sequence of two elements. Given another sequence $s_2 = (AB, D)$, then s_2 is a sub-sequence of s_1, and s_1 is a super sequence of s_2. The prior example is from the text [Han15]; however, an extension is necessary to visualize the concept of a gap. Let the gap parameter be an integer and, for example, a gap of 0 or 1; this sets the tolerance for a zero- or one-unit inconsistency between similar patterns.

Let $z_3 = (AC, D)$, then $z_3 \sqsubseteq z_1$ since $AC \sqsubseteq ABC$ as the missing element B is a one-unit gap between the two compared sequences.

The **support** of a sequence is the number of times that it has been observed in the window of time, $sup(s_1) = \#(s_1)$ instances in the data. Often the frequent pattern mining methods require a user-defined notion of *frequent*; this can be fulfilled by setting the minimum support value to be the threshold when deciding if a sequence is frequent.

Regarding the two concepts of full/closed set pattern mining, the distinction is explained in [Han15]: "all subsequences of a frequent sequence are also frequent, mining the set of closed sequential patterns may avoid the generation of unnecessary subsequences and thus lead to more compact results ... " The prior statement emphasizes that the discovery of all combinations of events to mine (full set) results in slower runtime execution due to the number of operations compared to closed algorithms. The algorithms in full set tend to differ in the amount of storage, such as the recursion (PrefixSpan algorithm) versus the usual iterative design (SPADE algorithm). Indeed, using the SPADE algorithm implemented in the R programming language as a LP method was not feasible due to the near real-time constraint past the first data set (specified in Section 4.3). Table 3.3 provides a brief overview on candidate

algorithms for the data mining representative LP method. Of the two algorithms, portions of Prefix-Span's framework was selected by authors of [WH04, LW08] to formulate the BIDE algorithms, specifically the notion of prefix-projection and using tree data structures to facilitate traversal and omit storing large temporary data tables (as SPADE) to conduct closure checking. Projecting a longer sequence by traversing a tree structure beginning at some prefix is the method which Prefix-Span uses to eliminate the need to store data tables in main memory as does SPADE; the difference is that BIDE will perform **bI-directional extension** pattern closure checking on the prefix traversal tree in order to reduce the number of patterns explicitly considered. Unlike both algorithms in Table 3.3, BIDE traverses the reduced (due to closure checking) search space in a *depth-first* manner; this aligns with the method's approach of a detailed screening of a pattern for consideration given there are likely to be less patterns to explore than the usual breadth-first search case.

While there are other closed set pattern mining algorithms, the main competitor to BIDE is CLOSPAN, which both *significantly* outperform SPADE and Prefix-Span on dense, sparse, and semi-dense data sets, as shown in [WH04]. It follows that BIDE outperformed CLOSPAN, conditional on the absolute minimum support threshold being $\leq 60\%$, otherwise the two would interchangeably outperform each other, given certain outlying conditions. For the purpose of testing the LP-CDA against the scenarios detailed in Chapter 4, the $support_{min} \leq 60\%$ condition is critical. Since the data to mine represent many users' query usage patterns, it is unlikely to find enough cuboids more than 60% frequent overall such that the smart materialization will significantly impact query response time.

In summary, a comparison of BIDE in relation to its counterparts is provided in Table 3.3:

1. Scan the input data for frequent unary sequences, compute their supports.
2. Given each frequent unary sequence (these form the initial set of prefixes), prune each sequence (takes effect on $\geq 2-ary$ sequences) if it satisfies the pattern closure assumptions; then the sequence is not to be explored further. Otherwise, the prefix is extended (prefix projection method) using bi-directional extension, and the forward and backward extension of a prefix is considered. A detailed overview of the BIDE procedure is provided in [WH04], in which the author states a prefix is not necessarily the start of a traversal in the prefix tree; this algorithm addresses a limitation of Prefix-Span that allows the algorithm to accept forward *exclusively* or backward prefix-projection (extending or lengthening of the prefix to grow it into a longer sequence).

From the methods surveyed in [Han15], BIDE and, specifically, its variant denoted Gap-BIDE, were carried into experimentation in Part II for their capacity to handle the data sets in question under the constraints outlined previously.

3.3.1 Gap-BIDE

Crucially, Gap-BIDE, which extends the BIDE framework to efficiently mine closed set patterns in dense and sparse data, also tolerates *gaps* in such patterns. The notion of gaps exemplified in Figure 3.3 on the continuous number line pertaining to trends in functions, is now addressed by Gap-BIDE on categorical data. In particular, the algorithm is intended to extract frequent patterns which would otherwise be disqualified due to insufficient support as a result of a *temporal* gap. For instance, this algorithm has been tested on biological events or protein sequencing, and has been tasked to extract frequent patterns in relation to an undesirable behaviour (bugs) in large-scale software products [LW08]. In the case of frequent cuboid mining, the gap in dimensions is of higher interest than temporal since the LP-CDA framework relies on fixed time intervals where dependence between events is not explicitly estimated

Algorithm	**Description**	**Procedure (in sequential order)**
SPADE	⟶ sequential pattern discovery using equivalent classes ⟶ full set method, using breadth first search with apriori pruning ⟶ forms **candidate** sequences, these are combinations of unary terms to compose increasing in length sequences ⟶ due to the numerous joins between data tables of subsequences to calculate the support of supersequences, it is slower at runtime as the sequence length increases relative to Prefix-Span and closed set methods. These joins are not only operationally intensive, they also require large amounts of redundant data to be stored and brought into main memory to yield the result.	1) One scan of the data set to find all unary (sub)sequences and tally their supports; the support of a sequence relays the number of times it was found in the data set. 2) The support of 2-ary sequences is the join of all pairs in the unary sequences data table compiled in step 1. 3) $k+1^{th}$-ary sequences is the k^{th} join involving the prior k^{th} long sequence data table. 4) Repeat until **no** new supersequence has its $support \geq support_{min}$ threshold.
Prefix-Span	⟶ a divide and conquer method of mining by way of recursive discovery of the subsequences ⟶ full set method applying breadth first search ⟶ does not calculate any **candidate** sequences; thus less sequences are mined (unless the data set contains all permutations of unary sequences), requiring less storage space and less operations than SPADE	1) Begin with unary prefixes (subsequences) and initiate the prefix-projection method, registering the index (position) of each prefix instance in the data set. 2) The next (+1 in sequence length) sequence is identified recursively from its prefix. The recursive nature of this algorithm leverages a tree data structure which stores a unary term at each node, where the traversal of a path in the tree reveals the supersequences to each node. In this way, only sequences which appear in the dataset will appear in the tree.

Table 3.3: Comparison of a diverse set of methods for sequential pattern mining analyzed in [Han15]. These were selected for an exploration of a diverse set of approaches.

during the prediction step. Also, as highlighted previously, the BIDE framework prunes the search space (of patterns) to continually track or *grow* into longer patterns at the initial discovery of each pattern—refer to step 2 in BIDE's summary—thus avoiding the need to store candidate patterns.

The hyperparameters and their initialization are:

- minimum support threshold is two-fold: $min\_sup_{day} = 40\%$ and $min\_sup_{week} = 60\%$
- gap constraint of $[M, N] = [0, 2]$
- input data, the window of cuboids for k cycles of a given time scale, each sequence is a lexicographical ordering of some DM's dimension(s)

The initialization of the above parameters is based on early experimentation with Gap-BIDE on fragments of data to observe tendencies in runtime and effectiveness.

To accommodate gap constraints, the following components were replaced in the BIDE framework [LW08]:

a) **frequent gap-constrained pattern enumeration**: recursively extending each pattern discovered as in Prefix-Span; however, rather than collecting the index of each instance to later traverse the prefix tree in a single direction, Gap-BIDE applies the bi-directional extension of non-closed patterns. In addition, the traversal accounts for a tolerated range of gaps in any sequence discovered, and is actioned on by skipping over $x \in [M, N]$ items within a sequence.

b) **gap-constrained pattern checking scheme**: if a sequence's support remains the same when extended in either direction along the prefix traversal tree, then it is closed; further extensions of the sequence are not explored. If a sequence's support satisfies the minimum support threshold, it is retained in the result set.

c) **gap-constrained pruning**: if an extension of a given prefix has the same support as its shorter version due to a gap allowance, the later sequence is retained, subject to the minimum support threshold, and the former sequence is omitted.

Gap-BIDE is specified in detail [LW08] and remains unchanged in its application in this thesis; it is implemented by [gpbed]. There were numerous transformations applied to the input data to integrate the stand-alone algorithm with the LP-CDA framework in Algorithm 1; Gap-BIDE is called on lines 12 and 15 in certain approaches, denoted in Algorithm 1.

3.4 LSTM RNN

It has been hypothesized that a portion of ML methods attempt to emulate the human brain in the form of ANN, though this may not reflect the actual science of neurons and their synapses in the human brain as continually discovered—it has lent itself to algorithms which outperform many others in predicting numerical responses or categorical labels of sequences of data. In addition, it famously showed great results in natural language processing (NLP) to fulfill the requirement of tagging portions of

free text (unstructured data) with their role in sentences particular to the grammar of a language, this to enable sentiment analysis and comparison of meaning across differently written texts or messages [SPH19]. Although ANN algorithms require a significantly larger training set than statistical methods, the trade off is more intricate refinement (model higher degree of variation in the data) to estimate relations among the observations and their response. Specifically, in order to fully process a sequence of unknown length to then anticipate or distinguish future patterns, a Recurrent Neural Network (RNN) is most suitable among regular ANN and convolutional neural networks (the latter found success in computer vision).

A feedforward ANN is a RNN that has the additional ability of *internal memory*; this ability is implemented by *gates* and the feedforward arrangement of neurons (cells in ANN speak) in the feedforward manner, depicted by Figure 3.4. By default, ANN is a collection of independent cells which, when combined, can estimate complex deterministic functions, given an input value inclusive of its dependency on time or other variations in the data process. As a result, a cell in an RNN is fed the historical input value as transformed by its previous cell in the network; the degree of dependency between previous values in a sequence can be extracted by the fitted RNN. Figure 3.5 provides an example of a multiple cell RNN which can process two observed inputs at time (or timestep) t and $t + 1$, whereby having the ability to predict the response at $t + 2$. In contrast to Figure 3.4, the structure in Figure 3.5 has its *hidden layer* connections feeding the extracted information forwards or up one layer to process the input at the next timestep. Given a long sequence of events, one can expect nonlinear relations to exist and contribute significantly to inferring the outcome, thus a number of hidden layers can be placed between the layer accepting the inputs and decoding the final output of the network. This formation leads to the claim that the overall network is "deep". As shown in the same Figure 3.5, the two portions connected by the hidden layer dotted lines are usually shown in a vertical placement, although for descriptive purposes they appear horizontally oriented in the figure, reinforcing the notion of "deep" learning synonymous with the aim to model non-linear variations in the data. Therefore, the estimation of an input depends on an estimated (trained) network of recurring functions controlling the throughput of values along the arrow lines (in the two figures), in order to predict time and historically dependent responses in data processes.

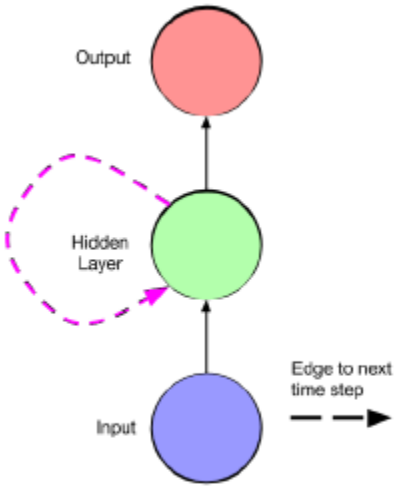


Figure 3.4: An example of a simple RNN taken from [LBE15] shows one *cell*, i.e. a network of one unit. This unit (cell) contains three nodes (circular components) and the transfer of information (gates) in the form of arrows.

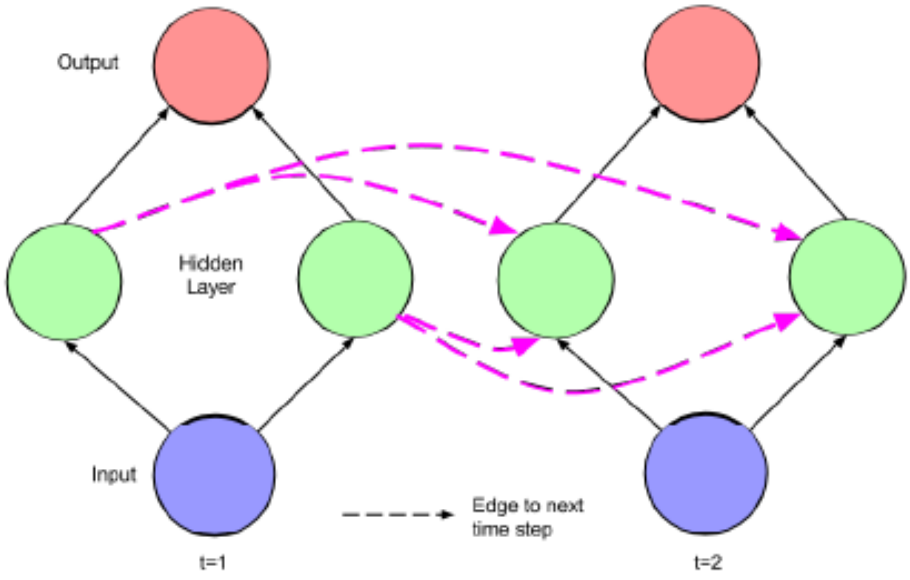


Figure 3.5: An example of a multiple cell RNN taken from [LBE15] shows information extracted from the first input on the left being transferred in the processing of the next input on the right.

The capacity to distinguish sequential relations in the generation of output values by RNN is challenged with long sequences due to the *vanishing gradient* problem [HS97, LBE15]. To extend on the prior, as the sequence of events grows longer, the older inputs have less of an influence in the network despite having a material influence in real life (the data-generating process)—this limitation is the vanishing of early information gradually (gradient) over time (length of sequence). To resolve the vanishing gradient problem, Long-Short Term Memory (LSTM) cells are used within the feedforward RNN architecture, whereby the use of *four* cell components achieves long-term memory (allowing the learning of long-term dependencies among sequential events) and regulates to forget appropriate amounts of information via its *forget gate* and use of *dropout layers* [HS97, Ola15]. Three gates are included in a LSTM cell: the input, output, and forget gate; each is a collection of mathematical functions. Aside from the three gates, the fourth component of the cell is the *internal state* which acts as a conduit, whereby the new input, the prior information from the previous LSTM cell, and the generated output of the current cell merge using either pointwise multiplication or by applying a sigmoid function. This internal state is the long-term memory and conveys the cell's conclusive decision on the input and historical data to the next cell in the chain in a given network layer. In turn, the receiving cell uses its forget gate to decide how much of the prior cells' numerical value should inform the current cell's decision on its output; this gate leverages a function to adjust the prior cell's internal state by a factor between 0 and 1. A more detailed formulation of the LSTM-RNN can be found in [HS97], a higher-level perspective is provided by [LBE15], and a more accessible technical review is offered by [Ola15]. The LSTM-RNN network applied here is based on the Python implementation [kered], with the remaining challenge being to manage the balance between an accurately (in terms of predictive output) built model and one meeting the near real-time constraint. Particularly, like the Gap-BIDE (Gap-BIDE) method discussed previously, the LSTM model requires re-training per each critical scale cycle (each day of the week); only BBB has updating functions eliminating the need to re-train from scratch. Another challenge is the general requirement of larger training sets than its peer methods in statistical learning for sequential/time series processing. This large training set requirement had to be lessened at some cost to *kpi* because, due to frequent changes in the training window, it is likely for us to encounter small sets. The balance between prediction correctness (*kpi*) and feasibility in terms of runtime was sought and achieved (in Chapter 5) using

the following model build:

The implemented LSTM LP-CDA framework takes a time scale to train the model and pulls in that scale's preprocessed data to further transform it specific to the KERAS (Python framework implementing the model [kered]) input matrix. Then, the training procedure is initiated to build a model which takes in the cuboid label and its count of occurrence at a given time step to then predict the count per cuboid in the immediate next time step. The layers of the LSTM-RNN used to fulfill the LP role in Algorithm 1, in order are:

1. The first layer takes on the encoded input matrix and uses 100 LSTM cells; the cell's internal state is computed using RELU (rectified linear activation function). RELU is chosen over other options (such as sigmoid) since it allows ANN to learn the non-linear dependencies between the features. This learning of dependencies is an advantage over the prior two LP methods discussed in this chapter.

2. The dropout layer with a rate of 0.4, which takes on a similar role between layers to the forget gate between LSTM cells. A distinction is that this layer "randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent **overfitting**. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged" [kered].

3. LSTM layer with 50 cells; activation function remains of the form RELU.

4. The dropout layer with a rate of 0.2.

5. The dense layer formulates the final result given the inputs; from here, the output is obtained for predicting the next time step.

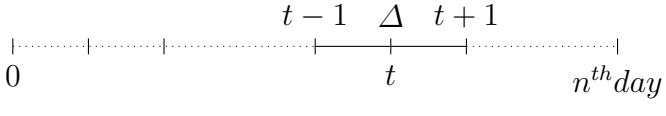
The parameters were obtained through early experimentation on the synthetic data, and not the exact sets constructed in Chapter 4. The training of the model is carried out over 200 epochs; each epoch consists of training the model on subsets of the training data one at a time [kered].

3.5 Change Detection

Continuing from Section 2.4, here three detection-adaption combinations are specified to be integrated with the predictive methods in the manner highlighted in Algorithm 1. First, a heuristic detection and adaption method is specified to fit this thesis' use case; then, two generalized combinations build on the foundations of CUSUM to better detect changes using *monitoring statistical differences* by way of Drift Detection Method (DDM) and its successor, Early Drift Detection Method (EDDM).

The types of change in consideration are:

1. a singular cyclic change
2. the change which is the start of a new general tendency in users' behaviours
3. a change part of several changes, a sequence of incremental change in the overall users' query behaviours

When the threshold frequency of cuboids in demand by the DBMS within a cycle of the critical scale is not accounted for in the materialized view, the system *detects* a change. At the time of detecting a change, it is not clear yet if this is an outlying change (1), an abrupt change (2), or incremental change (3). Suppose the general case: 

where we depict a change occurring on the t^{th} cycle of the daily time scale. In the **heuristic** change detection-adaption method, given that we detect the change for cycle $t+1$, the predictive process will only use historical data of $\{t, t-1\}$ cycles. Limiting consideration to the last two cycles' (**last two cycle policy**) historical data in the presence of change follows the logic of the posterior updating functions from Section 3.2. At the time of computing predictions for the $t+1$ cycle, the system resets the prior of the beta-binomial distribution for all cuboids back to the non-informative prior in Section 3.2. Then, the priors are updated using $\{t-1, t\}$ cycles' historical data; in this way, the $t+1$ cycle's prediction is *fairly* influenced by a sample of the previous behaviour and of the change in behaviour. The influence is dependent on the weight of each prior

cycle manifested by the total number of events within the cycle (variable n in Section 3.2). At the time of the change detection, there is not enough evidence in the system to differentiate between the types of change. The *last two cycle policy*, in combination with the adaption specified in this section, addresses the following scenarios:

I The $t+1$ cycle is more similar to the past behaviour ($t-1$ cycle) than t.

II The $t+1$ cycle is more similar to the t^{th} cycle than $t-1$ cycle.

   (a) If the behaviour in $t+1$'s following cycles is similar to cycle $t+1$'s, then no additional change in $t+2$ is detected and there is no subsequent change adaption by the system.

   (b) If the system detects further changes shortly (relative to the same time scale), then the system frequently runs its adaption procedure.

The situation **I** involves a type 1 change, then **II.a** refers to a type 2 change, and **II.b** exemplifies a type 3 drift. First, some definitions are provided:

- T is a percentage value; it is the threshold value for the minimum accepted performance. $T \in [0,1]$, setting $T = 0.75$ means a minimum of 75% correctness is accepted in our materialization at the critical scale without using change adaption processes.

- $kpi \in [0,1]$ is the ratio between the number of times a cuboid is part of a query and in the materialized view, and the total number of requests for any cuboid in a cycle. It is determined by:

$$kpi = \frac{count(cuboids \in Queries \ \ \& \ \ cuboids \in Materialized\ view)}{count(cuboids \in Queries)} \tag{3.7}$$

  within a cycle of the critical scale.

The general adaption in response to a detected change by the heuristic detection method ($kpi < T$) is specified in Algorithm 3, where lines 2 and 12 of the algorithm feed into the rate of change detection comparison analysis of Chapter 5. Then, line 4 is the proportional increase in memory available for materializing critical scale

predictions in the next cycle, up to a maximum of the total reserved memory between all scales of prediction (lines 5 to 7). Next, at line 9, the leftover memory after increasing that allocated to the critical scale is assigned to the non-critical scales in order of priority, and in this thesis' use case, it is only the weekly scale. The support measure for data mining methods is decreased in line 10, thus lowering the minimum frequency required for discovering candidate frequent patterns, as covered in Section 3.3. The *last two cycle policy* is applied to the critical scale's window of training data in line 11, where *windows[...][i]* is a vector of two entries in the form *most recent cycle#, oldest cycle#)*.

```
   Data: the lists sizes and betaBparams
   Result: update the data
 1 for each cycle of the critical scale do
 2     change_flag ⟵ 0
 3     if kpi < T then
 4         critical_size ⟵ size[critical_scale] ∗ (1 + T − kpi)
 5         if critical_size>size["allowable"] then
 6             critical_size ⟵ size["allowable"]
 7         end
 8         size[critical_scale] ⟵ (critical_size)
 9         size["week"] ⟵ size["allowable"] − size["day"]
10         support.update(critical_scale : kpi*support[critical_scale])
11         windows.update(critical_scale :
            [windows[critical_scale][1]-1,windows[critical_scale][1]])
12         change_flag ⟵ 1
13     end
14 end
```

Algorithm 3: Heuristic change detection-adaption

In Section 2.4, the change detection algorithm CUSUM and its many variations/derivations were introduced as a *sequential analysis* method of detection which also produces *control charts*. These would complement Shewart and other moving averages-based charts, which include a measure of the data process on the y-axis, time or grouping of observations on the x-axis, and setting upper/lower thresholds for "normal" behaviour of the measure in which to remain enclosed. For instance,

CUSUM detects small *systemic* shifts in the quality characteristic a data process is monitored under (*kpi*). Due to the implication that a change has to be drawn out over a period of time, it detects abrupt changes at a delayed rate relative to the actual start of the abrupt change, which are evident in the manual inspection of the *kpi* over time. Slow detection creates a missed opportunity for the adaption procedure to impact the *kpi*, where otherwise no-adaption would yield the same result, meaning the cost of the adaption is incurred without the benefit of it (outweighing the cost). Cost refers to the overhead in making changes to the resources allocated for materialization and actual CPU usage of running detection and adaption actions. For CUSUM, the abrupt change may not leave any evidence of the fact in the control chart since the extreme value deviation of a few instances of time would not outweigh the consistent variation captured up to these points in time; the consequences of delayed or even missed detection are the following:

- an undetected abrupt change (type 1) would not be evident even in the visual inspection of the chart, thus models which are not robust to change, such as decision trees, will be swayed off course from the general tendency in the data process.
- a type 2 change may have a lengthier duration bordering on an incremental change, however only a slight deviation outside the δ threshold will register in the chart. This significantly underestimates the magnitude of the change and misleads the period for which a predictive method was off course.

Variations of CUSUM have addressed these challenges and are summarized in a technical report [Pie13]. Due to the challenges and their patchwork resolutions on the CUSUM algorithm, the search remains for methods which build on the lessons learned over the many iterations of the algorithm and which would work in a *data stream environment*; this area of research complements the popularity in ML over the last decade. The *data stream* refers to a continual capturing of observations and discarding of certain data. The primary challenge is that one cannot, by limitations of the data process or sheer storage amount required, store all the observations to use as a training set and the basis upon which to fit a model. The streaming of observations, as they relate to this thesis' use case, become available as they occur dependent on the users' pace of querying the DM; these are examined by the prediction and change-

detection algorithms in fixed size time intervals (the *windows*[...][i] in Algorithm 3). Two methods which work well in near real-time are introduced in the next section.

3.5.1 (E)DDM Algorithms

A method developed in [GMCR04b], denoted DDM, would handle the abrupt changes with which CUSUM struggles. In turn EDDM, improves on the detection of *slow* gradual change with which DDM struggles, as highlighted during experimentation in [BGCAF+06]. Both DDM and EDDM stem from the idea of detecting differences in classification error as opposed to the variation of measure supposedly summarizing the data process over time (as in CUSUM and other control chart methods). To model change, the time series data is first represented as a sequence of Bernoulli trials indicating if the data process under investigation was misclassified as a binary value; consequently, these trials follow a binomial distribution. In addition, these two algorithms share the assumption that as the amount of trials increases over time, a decrease in the error *rate* should be observed if the behaviour of the data is consistent. The approximation of the binomial distribution using the normal distribution given a large sample is leveraged in both methods [GMCR04b, BGCAF+06]. Crucially,

- for each cycle t in the sequence of events,
- the error ($kpi < T$) rate is p_i,
- the (prediction error) data process has a standard deviation of $s_t = \sqrt{p_t \frac{1 - p_t}{t}}$

Of interest are the thresholds used by both change detection algorithms in this section, specifically:

DDM: $P_t + s_t \geq p_{min} + 3s_{min}$, when this condition is met on line 7 of Algorithm 4 a concept drift is detected; this is tracked for comparison across approaches by lines 2 and 22.
The primary measure of interest is the error rate of classification or the misclassifying error [GMCR04b]; classification is the denotation for predicting labels rather than values on the continuous or discrete scale—thus it refers to the prediction error (kpi) of the three methods discussed in this chapter.

- The parameters were adjusted through early experimentation given the challenge of a short transient period, i.e. $min\_num\_instances = 7$, referring to seven cycles of the critical scale, namely seven days.
- The default parameter inputs (line 10 of Algorithm 4) are $(30, 2, 3)$, as in [GMCR04a]; in this thesis' context, the primary limitation is the short transient period within which the detection algorithm must calibrate itself before pushing out verdicts on future observed cycles. By reducing the transient period from 30 to 7 days, the thresholds may need to be adjusted accordingly; indeed, with early experimentation the values that are hard coded in line 10 of the applied algorithm are used throughout the rest of the experimentation in Chapter 5.
- The function call on line 4 of Algorithm 4 conveys a misclassification occurred in the sequence of events. Conversely, lines 6 and 12 convey to the drift detector object that the prediction/classification was the desired outcome. Similar function calls with the same purpose as these also occur in Algorithm 5.

The detection or "out-of-control" threshold is most often inspected, while the "warning-level" is omitted from inspection; the authors of these two detection algorithms intended the warning-level to flag instances which nearly result in concept drift and for these be stored to replace the training set in the case of a detection flag being raised in the near future. Instead, the training window is adapted by a proportional shrinkage to the gap in kpi and threshold T, as on lines 8 and 21 of Algorithm 4. In addition, the training set is synthesized as per lines 11 and 12 after a change is detected in order to avoid downtime on the part of the detection algorithm where it

would have to wait the minimum number of instances to begin issuing its Boolean output. The initial seven instances are used to set the *tone* in *desired* observations ($kpi \geq T$) in place of *actual* for either DDM or EDDM to compare the next *actual* difference in T and the kpi. In early experimentation, many configurations of the first seven instances to relay to the detection algorithm and the effect on kpi lead to the settled scheme in the two Algorithms 4 and 5. This usage of the algorithm differs from the authors' prescribed use, in which the instances recorded between the "warning-level" and "drift-level" are stored aside and used in the training of the predictive model after a change is detected. Instead, the generalization of individual query behaviours into a single stream places the prediction methods at a disadvantage of not differentiating between trends which are strongly cyclical—correlated with time progression, and involving patterns which do not adhere strongly to progression in time and are dependent on factors outside what is measurable by the DBMS or other query patterns. Given the prior, a decision was made to fit the adaption strategy on the assumption that recentness of data is a major indicator of evolving and stationary trends in cuboid use, since dependency among queries and other factors is not of focus in this first iteration of the LP-CDA framework. As a result, the adaption strategy aims to ensure that sporadic instances or periods of historic data, indicated by the "warning-level" threshold, would not lead the majority of a newly constructed training set after a change is detected, and that the recentness portion of a training set perseveres since the dependency across cuboids is not investigated in detail by the prediction methods at this stage in the framework's life-cycle. A strategy to address the prior noted challenge in adapting to change is discussed in Section 6.1.

EDDM: $\frac{P'_t + 2s'_t}{p'_{max} + 2s'_{max}} < \beta$, when this condition is met on line 7 of Algorithm 5, a concept drift is detected and tracked by polling the $change\_flag$ variable at the end.
In contrast to DDM, the measure of interest is the **distance-error-rate**—the distance between the misclassifying errors where:

- p'_i is the average distance between two errors,
- s'_t is the standard deviation of the same form introduced previously, except replacing the p_t terms with p'_t
- throughout a continual run (no change detected), the maximum distance p'_{max} and standard deviation s'_{max}
- $p'_{max}+2s'_{max}$ spans 95% of the distribution of distance-error-rates observed within *tolerance*
- Two tolerance thresholds are initialized by the user, only β is of interest in this application of EDDM. $\beta = 0.9$ in [BGCAF+06], due to the shorter transient period in the data sequences than that in [BGCAF+06], the *actual value used consistently throughout evaluations in later chapters is* $\beta = 0.97$. This threshold was chosen after initial experimentation with fragments of data sequences. In addition, the authors of [BGCAF+06] allowed for a transient period of 15 samples, while in this thesis only seven are used; this is to provide for a consistent simulation with the other detection-adaption methods which use only a seven-day period of data on which to train—as the prediction methods use only five-day periods of data.

The expectation for these two algorithms is that EDDM will raise more frequent detection alarms which yields the proportional adaption (as in lines 18 to 25 of Algorithm 5) which is less drastic than DDM (lines 18 to 21 of Algorithm 4) when slow gradual changes exist. It will be evident in Chapter 4

Both of these detection methods can be used with any (statistical/machine) learning algorithm in two ways [BGCAF+06]:

I wrapping a batch learning algorithm

II implemented within an incremental and online learning algorithm

As evident from the involvement of change detection in Algorithm 1, this thesis utilizes option I for manageable integration with the three distinct prediction algorithms, where BBB is the incremental learner and the other two (LSTM and Gap-BIDE) are batch learners.

```
   Data: the lists sizes and betaBparams
   Result: update the data
 1 for each cycle of the critical scale do
 2     change_flag ⟵ 0
 3     if T − kpi > 0 then
 4         ddm.add_element(0)
 5     else
 6         ddm.add_element(1)
 7     if ddm.detected_change() is TRUE then
 8         offsetWindowBy ⟵ round((1 − ddm.get_length_estimation()) *
             (windows[critical_scale][1] − windows[critical_scale][0]), 0)
 9         ddm.reset
10         ddm.set_params(min_num_instances=7, warning_level=0.9,
             out_control_level=0.988)
11         for h ∈ [0, 6) do
12             ddm.add_element(1)
13         end
14         critical_size ⟵ size[critical_scale] * (1 + T − kpi)
15         if critical_size>size["allowable"] then
16             critical_size ⟵ size["allowable"]
17         end
18         size[critical_scale] ⟵ (critical_size)
19         size["week"] ⟵ size["allowable"] − size["day"]
20         support.update(critical_scale : kpi*support[critical_scale])
21         windows.update(critical_scale :
             [windows[critical_scale][1]-offsetWindowBy,
             windows[critical_scale][1]])
22         change_flag ⟵ 1
23     end
24 end
```

Algorithm 4: DDM change detection-adaption

Data: the lists *sizes* and *betaBparams*
Result: update the **data**
for *each cycle of the critical scale* **do**
  $change\_flag \longleftarrow 0$
  if $T - kpi > 0$ **then**
    eddm.add_element(0)
  else
    eddm.add_element(1)
  if *eddm.detected_change() is TRUE* **then**
    $offsetWindowBy \longleftarrow round((1 - (eddm.get\_length\_estimation() - 1)) * (windows[critical\_scale][1] - windows[critical\_scale][0]), 0)$
    eddm.reset
    if *kpi*$<T$ **then**
      critical_size $\longleftarrow$ size[critical_scale]*(1+T-kpi)
    else
      critical_size $\longleftarrow$ size[critical_scale]*(1+kpi-T)
    end
    for $h \in [0, 6)$ **do**
      eddm.add_element(1)
    end
    $critical\_size \longleftarrow size[critical\_scale] * (1 + T - kpi)$
    if *critical_size*>*size["allowable"]* **then**
      critical_size $\longleftarrow$ size["allowable"]
    end
    $size[critical\_scale] \longleftarrow (critical\_size)$
    $size["week"] \longleftarrow size["allowable"] - size["day"]$
    support.update(critical_scale : kpi*support[critical_scale])
    windows.update(critical_scale :
     [windows[critical_scale][1]-offsetWindowBy,
     windows[critical_scale][1]])
    $change\_flag \longleftarrow 1$
  end
end

Algorithm 5: EDDM change detection-adaption

3.6 Conclusion

The culmination of the multi-scale concept, three methods fulfilling the role of LP in the framework and three others addressing the CDA module, is an ensemble of the form shown in Figure 3.6. The resultant approaches for combining the ensemble members lead to the experimentation detailed in the next chapter.

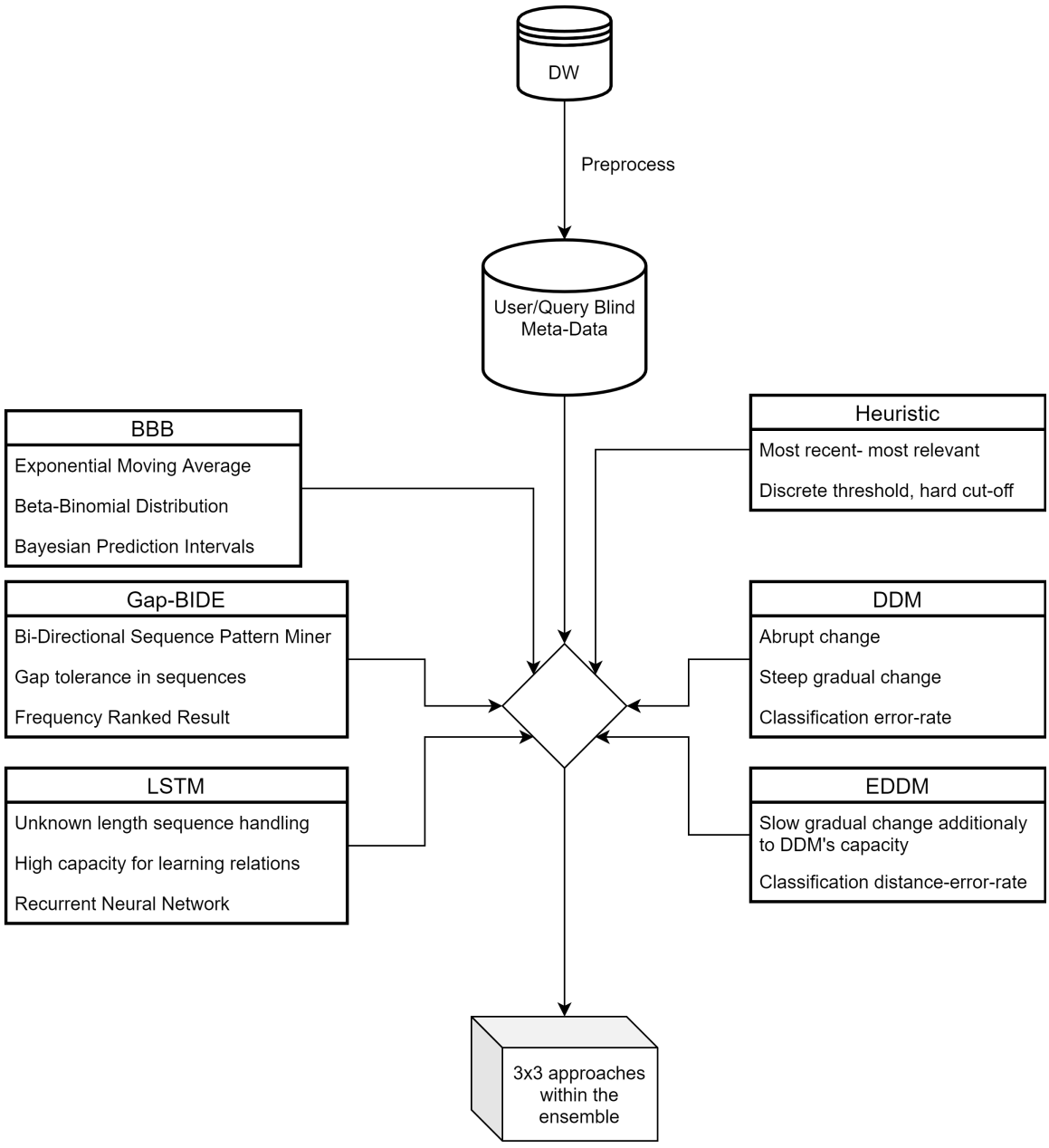


Figure 3.6: An overview of the methods forming the ensemble with which to experiment.

Part II

Simulations

Chapter 4

Experimental Design

In this chapter, the construction of the three data sets is described, in addition to the process of validating an approach's effectiveness in predicting a data set at each time step, and the programming and hardware in use.

The chapter begins with an description of the motivating proprietary data set [Air15] belonging to *AirBNB*, released on the data sharing platform *Kaggle*. The data set is incomplete in the amount of data for a DM usage, and missing the meta data this thesis aims to analyze using the LP-CDA framework. The first step, described in Section 4.1, was to break down the flat files in [Air15] into plausible DMs and synthesize two more from the ground up to cover some of AirBNB's main business processes. Then, a set of queries are assigned by their ID to four query profiles representing four user groups; these queries are handwritten to establish the structure while the filters are randomly filled in. Examples of the queries executed are provided in the A.1; the remaining queries share a similar structure but differ in the tables joined and some random variation in the where clause of the SQL queries. The queries were executed in turn 30 times to collect the mean duration in minutes and the variance measures shown in Table A.2. The synthesis of the time series that makes up the final data sets on which to validate LP-CDA is specified in Section 4.3, where the measures in Table A.2 are taken as inputs. Finally, the validation process of the approaches within LP-CDA is outlined in Section 4.4.

4.1 Data Sources

The challenge of finding a data set which contains meta data of the user activity within a data set led to the extension of a real-world data set into data marts, and then later fictitious user group profiles. The real-world data set [Air15] obtained on the *Kaggle* data sharing platform contains a small portion of *AirBNB*'s user sign-up and user sessions observations. *AirBNB* is a multi-sided online marketplace for renting dwellings (houses, rooms, apartments, villas, and other types) for short stays; it is multi-sided in that users can be landlords and/or renters. Extracted from the two flat files provided in the [Air15] data set are two business processes which could be facilitated in a DW using the DMs synthesized in figures 4.1 and 4.2. The structure of the four DMs is not specified in [Air15]; however, the two prior DMs are inferred from the two flat files describing user account activity, and the remaining two (discussed next) are inferred loosely by the *AirBNB* business model and the business processes covered in [KR13].

The *sign-up* flat file describes measures and attributes at the time a user created an account, and the *sessions* file holds the attributes of the action a user took—the firm's version of queries. From the values seen in both flat files, one can infer the business questions (queries) posed by AirBNB's analysts and values on which to filter; specifically, on reservation events and inventory control analysis as these are major functions in the business model—supply and demand analysis within the internal *AirBNB* market. Relying on the scenarios explored in [KR13], two additional DMs were created from scratch to model reservation facts and inventory snapshots, as shown in figures 4.3 and 4.4. As the LP-CDA framework focuses on meta data of the firm's analysts' queries within the DW, an additional step of data synthesis is required to compile a set of queries and their execution start times in order to form a time series of events which require various cuboids for varying duration. The prior time series is the focus of predictions and change adaption methods covered in Chapter 3.

Four user group profiles are synthesized to contain a collection of queries of similar structure where their filter (SQL where clause) can differ randomly; samples of the structure of queries are provided in Section A.1 for two profiles and the remaining two share similar composition with different dimensional joins. Each query is associated

to a cuboid, and the labelling of a cuboid is a short form of the joined dimension tables (non-intermediary, must contain attributes) ; this can be seen in the captions of the queries in Section A.1 and by using the *AS* operator in the query instances. Queries are called into action in the database according to a synthetic schedule detailed in Section 4.3. Three time series are generated in this manner to assess the approaches from 3.

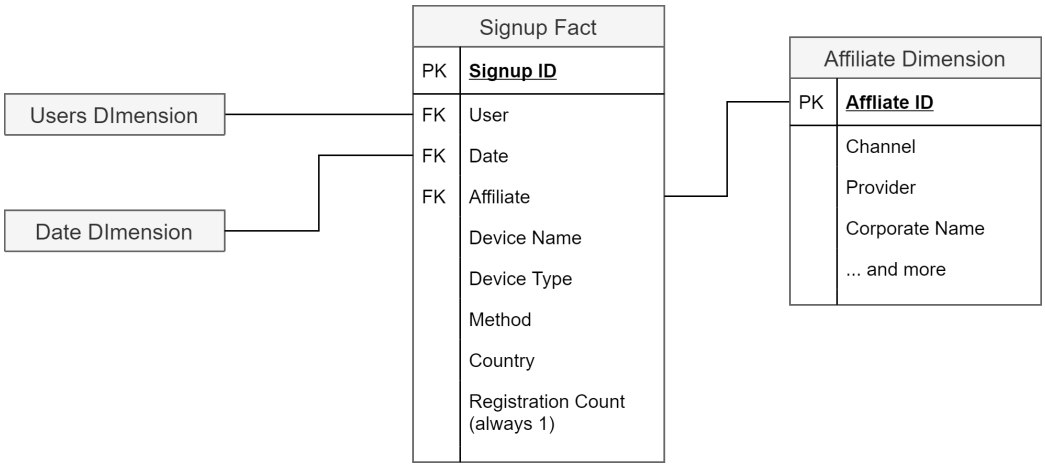


Figure 4.1: Description of the event of a user signing up for an *AirBNB* account. The DM is inferred from one of the two flat files available in the data set released on the *Kaggle* portal. An affiliate is a third-party company, in particular, other online websites which *AirBNB* uses to advertise itself; it is indicative from the flat data file that tracking the user journey from one of these external platforms is of interest to justify marketing costs per each affiliate. This is reflected in the second user group profile of queries.

4.2 Simulation Setup

There were two main branches of work undertaken to simulate the LP-CDA framework: the first branch focused on inferring database schema and creating new ones in place of missing definitions to formulate query execution meta data in a time series manner, and the second branch focused on passing on a matrix keeping count of instances of each cuboid and time axes to the LP-CDA framework to compute predictions and self-correct in the instance of concept drift.

First, the two flat data files describing user sessions and signup events were broken

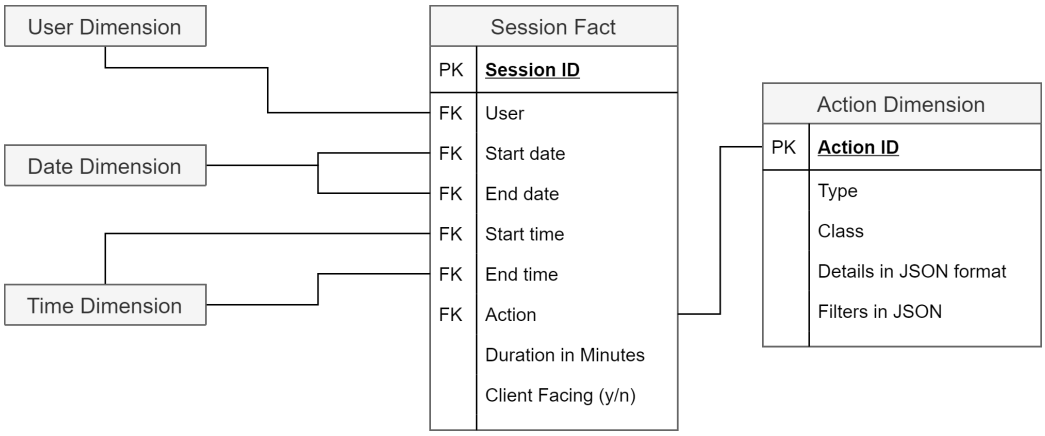


Figure 4.2: Second to the signup flat data file, this *sessions* DM is inferred from its corresponding flat data file that tracks a record of user actions at a descriptive level using attributes. The mention of details and filters in JSON is synthetic.

down into their set of dimensions and a fact for each of these two business processes of explicit interest by the firm AirBNB in the initial release of the data on Kaggle. The result of the prior process yields the DM for user signup events in Figure 4.1 and another DM for the process of tracking user actions in the AirBNB online platform, shown in Figure 4.2. With the exception of the date dimension and a few user-dimension attributes, all attributes and measures were inferred from the data in [Air15].

It is clear from the business model of the owner of data in [Air15] that property reservation is the bulk source of revenue and, while released data were not available on these new DMs, is designed for this business process which closely follows the retail sales dimensional modelling method in [KR13]. However, a difference is the dependency of location on dwelling seen by their direct link in Figure 4.3, unlike in the retail sales DM where location is independent of products sold (in place of dwellings in our case). An identical product may be available in multiple locations in the retail scope of business, whereas this would not carry into the property reservation business case since similar dwellings in structure have distinct legal, pricing, and capacity dependency on the region in which they permanently exist—in fact, the location is a defining feature of a dwelling in the AirBNB internal market.

While tracking reservation events allows analysts to learn about the demand of dwellings over time, locations, user demographics, and the influence of price dis-

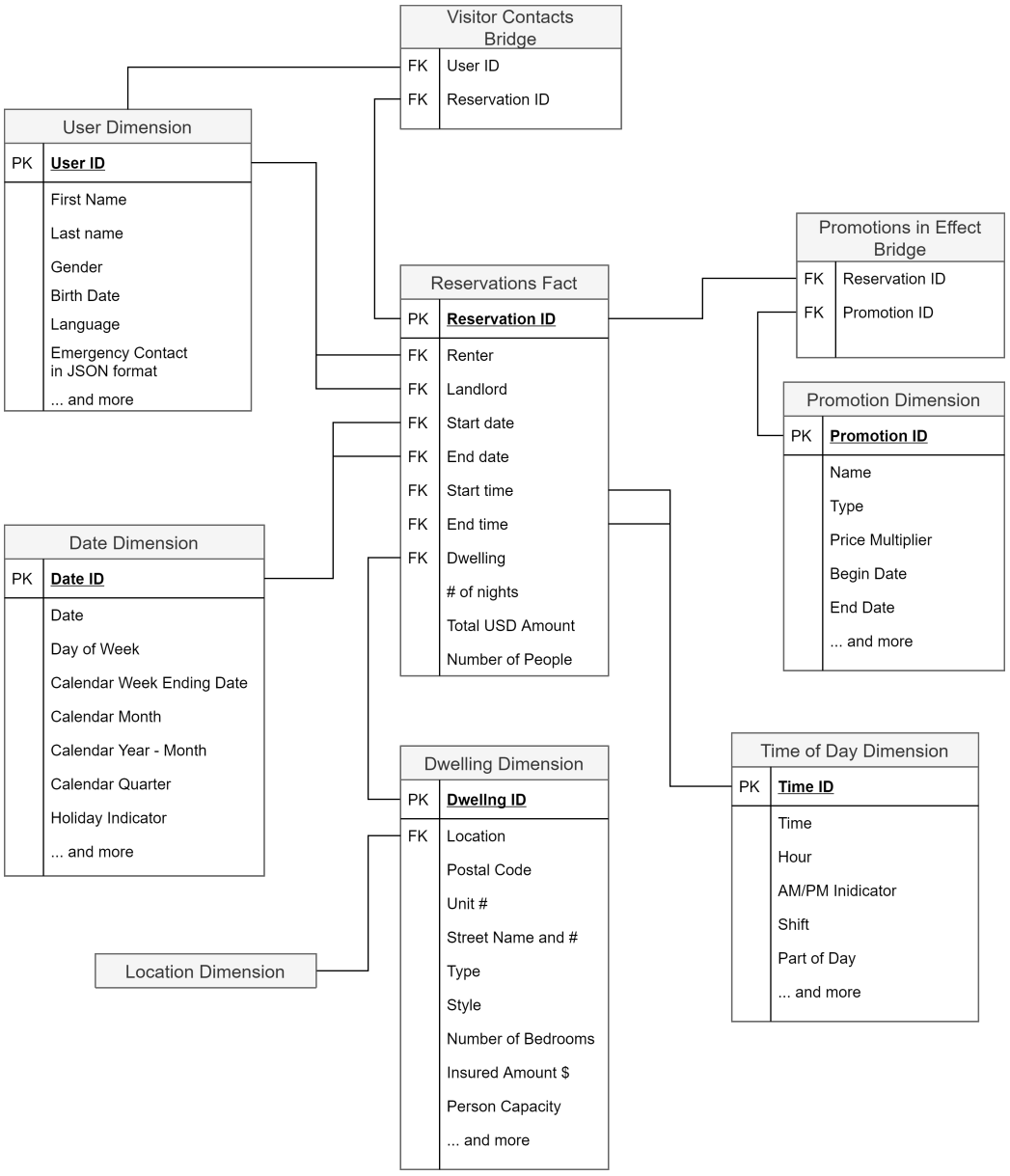


Figure 4.3: This DM describes the reservation of a dwelling to rent out for a period of time. It is an estimated business process given the multi-sided marketplace business model of the particular company *AirBNB* which released several flat files on the *Kaggle* online dataset sharing portal. The flat files were used to infer the DMs in Figures 4.1 and 4.2. Multi-sided marketplace refers to the facilitation of a business transaction between travelers and a host for their dwelling in the specific case of the *AirBNB* data from *Kaggle* (made available in 2018). Visually, a star schema is present; however, in this case, the location dimension is dependent on the dwelling reserved and as such is not linked directly to the fact table, as it would indicate it being independent of all other dimensions. Multiple links to a dimension table symbolize the number of roles a dimension plays via *views* in the database. The contact information of all the guests is necessary in the reservation even though they may not be users themselves or participate in the payment transaction to follow a reservation.

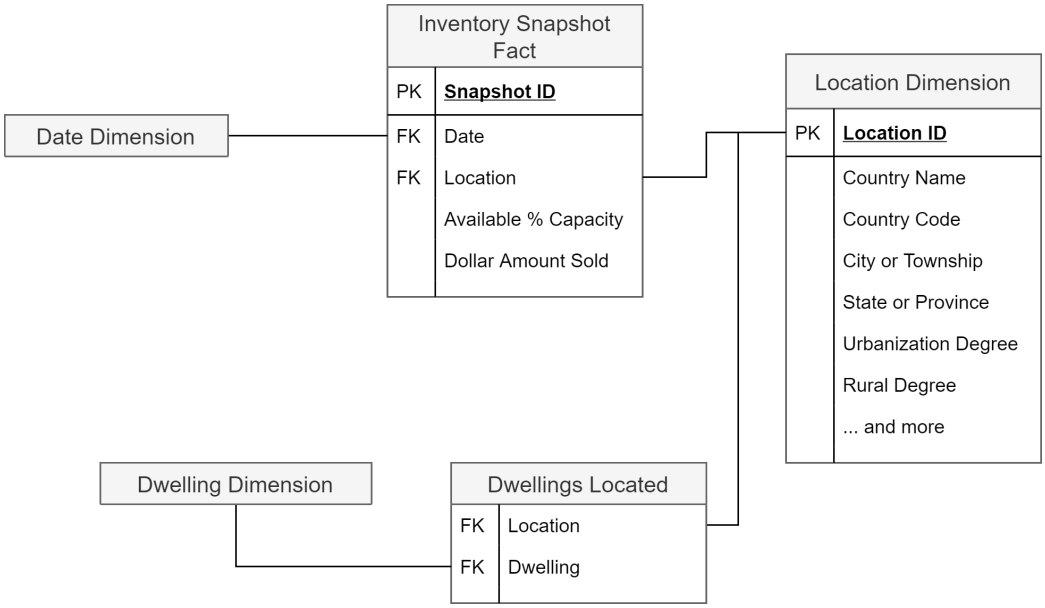


Figure 4.4: In contrast to the reservation DM in Figure 4.3 describing which dwellings are occupied, this DM describes the (partial or full) availability of dwellings; this allows analysis of the supply chain unlike the analysis of demand in reservations DM. The formulation in the figure is based on the snapshot design in [KR13]. In contrast to Figure 4.3, the analysis focuses on the location first for drill down and roll up operations, which is opposite of dwelling first in the reservations fact table.

counts, there are also questions on managing the supply of dwellings along similar dimensions to match up with this user demand, as shown in the inventory snapshot DM in Figure 4.4. In contrast with the prior three schematics, the fact table in Figure 4.4 is only a snapshot and requires updating at regular intervals (dictated by the business case and users) in order to provide relevant measures of dwellings available which reflect changes in the reservations DM [KR13]. Since the tuples in inventory and reservation fact tables are randomly generated, the refreshing of the inventory snapshot does not play a part in the simulation—the cuboids required by the firm's analysts are the focus of the LP-CDA and not the actual tuple results of the queries submitted for business case analyses.

Altogether, the four DMs were staged with a combination of SQL queries and *R* programming scripts to be uploaded into a *PostGre SQL Server* version 9.2 for Windows 10 operating system. A relatively small set of attribute values was extracted from the flat files in [Air15] and many synthetic values were added for more variation; these were then randomly assigned as rows to the fact table (by way of foreign keys) and made up measurements in the fact table. The fact table consisted of $500,000$ reservation facts (tuples), 1 million session facts, $200,000$ signup facts, and $1,000$ inventory snapshot facts. Following the loading of data in the DBMS, four user (analyst) group query profiles were created, as described in Section 4.2.1, in order to begin polling the duration of queries within each profile to establish a mean and standard deviation measure useful in synthesizing the time series in Section 4.3. In the end, the time series generated in the manner detailed in Section 4.3 in a tabular format was passed onto the LP-CDA framework to reinterpret into a matrix in which it was partitioned into a training window of observations on which to update the predictive models, and another test set partition to evaluate the *kpi* measure introduced in Chapter 3.

4.2.1 Query Profile

A profile contains a small set of structurally distinct queries which vary at the time of execution by their filter (SQL where clause) values; two of the four query profiles are exemplified in Section A.1. The remaining two profiles differ in the table joins, and otherwise are similar in structure to those shown in the A.1. Cuboids are often labelled with letters representing a dimension in a schema; the letter labelling is

exhibited in the *AS* operator in the SQL queries in the annex. The dimensions and cuboids are only referenced using this lettering scheme. (Table A.1).

The four query profiles represent the following business analytics roles that could be expected in a business model as *AirBNB*:

I Sales analysis, exemplified with queries in Section A.1. This analysis focuses on the reservations DM in search of patterns, such as a dependency on seasonality, regional events/holidays, promotions, or baseline location preferences, irrespective of other factors in the DW (social media, unforeseen events, etc.). Queries which interrogate the reservations DM in the sales analysis scope aim to either explore new data, re-affirm past trends for sales forecasting, or test hypotheses using models on the result of the query in third-party software (Python, R, SAS, SAP, MS Power BI, etc.).

II The focus is on acquiring new users and evaluating the customer pull methods enacted by the firm to establish the cost-benefit ratio of existing marketing decisions and better inform the next iteration of marketing campaigns. The affiliates seen in the signup DM describe third-party online platforms on which *AirBNB* advertises and expects these to redirect users to the *AirBNB* signup page; whether the user completes the signup procedure post redirection is another measure of interest of the analysts to track in the signup DM. The queries exemplified in Section A.1 cover the following:

- measuring the incoming traffic from affiliate sites
- comparing the affiliate redirected traffic with existing user profiles to either evaluate the level of engagement and financial commitments (as a buyer or seller of services) to perhaps cluster these users and inform the next marketing budget within *AirBNB*.

In this case, the analysis focuses on signup as much as retention of customers, hence there is an interest in the length of sessions spent viewing and searching listings over the course of time since signup (query A.8 in Section A.1).

III Analysis of the effect of promotions on reservations per region and the patterns of dwelling which attract sufficient attention for promotions to play a decisive

role in closing sales. Here, the dimensions of dwelling and location play a central role, rather than the user and financing aspects more typically assumed to affect reservations. Distinctions in the type of queries are shown at the dimension level in Table A.2.

IV The inventory control analyst focuses on the readability of dwellings in locations to match the timing set out by the sales forecasts. To accomplish this, an inventory snapshot is vital to evaluate the precise standing of dwelling occupations across the whole network. In contrast to the first profile which focuses on the demand of dwellings, this profile focuses on managing the supply to match projected demand.

The reader will note the four profiles are not mutually exclusive in their analyses; they are distinct in the overall set of queries executed and the granularity (focus) which certain data cubes are analyzed over others. It follows from real-world expectations that there is crossover in the queries performed, as the business processes the four profiles represent are dependent on each other.

4.2.2 Software

In the interest of achieving near real-time computation of the approaches, the Python programming language was used throughout. Initially, the work began in R, but it become evident the implementation could better leverage Python's *numpy* and *pandas* libraries in addition to $Keras$ for RNN. A summary of the libraries used to simulate the approaches from Chapter 3 is as follows:

- Gap-BIDE implemented by [gpbed] following the algorithm specified in [LW08]
- LSTM RNN-specific Python libraries are *Sklearn.preprocessing* and the $Keras$ package to feasibly experiment with RNN at pace [kered]
- Concept drift modules EDDM and DDM found in the $drift\ detection$ package within the $SKMultiFlow$ framework [MRBA18]
- The graphs appearing in this chapter were generated using the *matplotlib* library in Python [Hun07]

- Data transformations facilitated by *numpy* [Oli] and *pandas* [McK10]
- The Nemenyi post-hoc test using the *scmamp* library in *R* programming language. The figures 5.2, 5.3, 5.4, 5.5, and 5.6 were generated using this library's functions.

Experiments were run on a Dell Inspiron 16 laptop running Windows 10-64 bit and X64 base processor, with an Intel Core i7-6500U CPU @ 2.5 GHz and 2.6GHz access to 8 GB RAM.

4.3 Time Series Synthesis Sets

Three synthetic datasets were created using four user profiles varying in observation size, length of period under observation, number of correlated signals, and overall *difficulty* in fitting predictive models. There is a difficulty in terms of the (non) stationary signals and the types of concept drifts, their duration, and degree of overlap with parallel (along the time axis) signals, which represent the demand of a cuboid over time. As with variance in data posing the usual challenge of modelling uncertainty denoted by the text [JWHT13] as the *bias-variance trade-off*, the increase of variance across the three data sets can be seen in this chapter's figures.

To visualize the three data sets, boxplots are chosen to depict the variation of a cuboid over the entire duration of a time series, with one series per data set. Next, line graphs are inspected to spot change over time per cuboid, leading to visually spotting correlations among the cuboids all overlaid on a single plot, in contrast to pairwise comparison dot plots which are limited to viewing just two cuboids relative to each other and without a time axis; these figures are not included in this section as they span multiple pages and are cluttered. Next, in figures 4.5, 4.6, and 4.7, the boxplots show the summary statistics of all data sets. Sets 2 and 3 have many of their cuboids similarly varied, however the Set 3 has more outlying instances and the variation stretches out further from the mean of a cuboid (horizontal axis = $M, SU, P, W, etc.$) in many cases. The first data set is significantly different than the other sets, it is also a sparser data set than the rest. Overall, the easier set to predict

is the first in Figure 4.5, followed by set 2 from Figure 4.6, and finally, with the most variability and concept drifts, is set 3 shown in Figure 4.7.

An overview of the data sets against which the LP-CDA framework is validated is provided in Table A.1, and their construction is detailed in Section 4.3.1.

Data Set #	Description
1	100 observations over a 29-day period with 13 cuboids as consequence of the queries performed.
2	2208 observations over a 39-day period with 23 cuboids as consequence of the queries performed.
3	5978 observations over a 100-day period with 23 cuboids as consequence of the queries performed.

4.3.1 Data Synthesis

The algorithm to create the data sets has two components:

a) sample the arrival of queries and their durations per user group

b) manually select segments of query records (a vector of duration in hour units, date of instance, and cuboids involved to resolve query) from each user group sampled to aggregate these records into a data set. Referred to as set $x : x \in \mathbb{Z}^+$

To implement the first component (a), the sampling procedure was carried out for four user groups (found in Table A.2 of the appendix) consisting of the following tasks:

I Sample a Poisson distribution to specify the total number of events per unit time (particularly the hourly scale). Apply cumulative density function (cdf) inversion, as in Section 2.4 of [SM09].

> Given λ, t, λ is the average number of events each hour and t is the length of the period for which λ is relevant, then $n \sim Poisson(\lambda, t)$

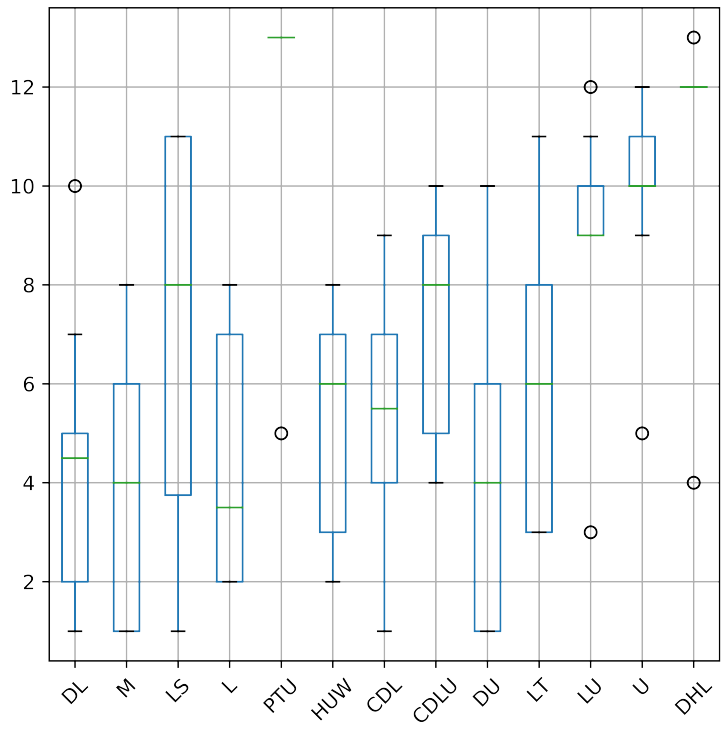


Figure 4.5: A boxplot of the rankings for Set 1, where lower values on the y-axis indicate a higher ranking (i.e., first place, second place, etc.) relative to its peers in the plot.

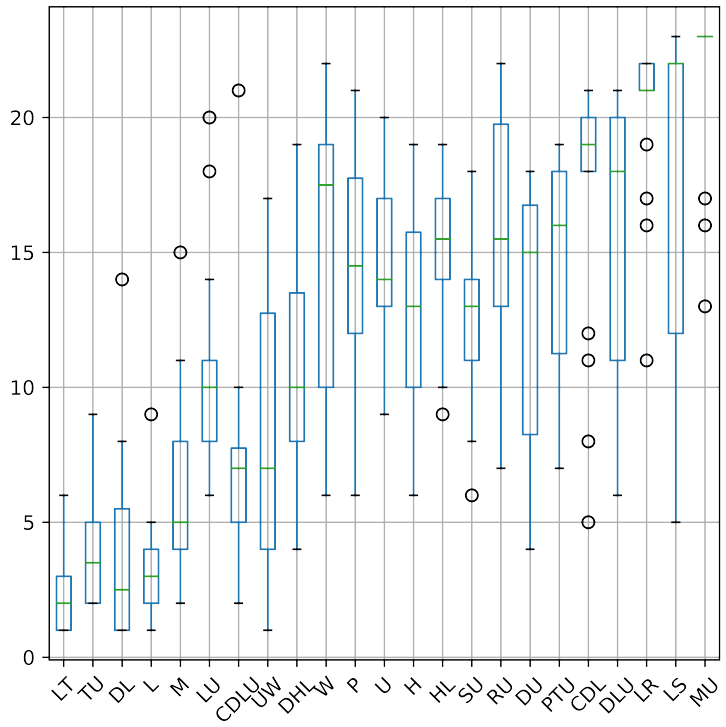


Figure 4.6: A boxplot of the rankings for set 2.

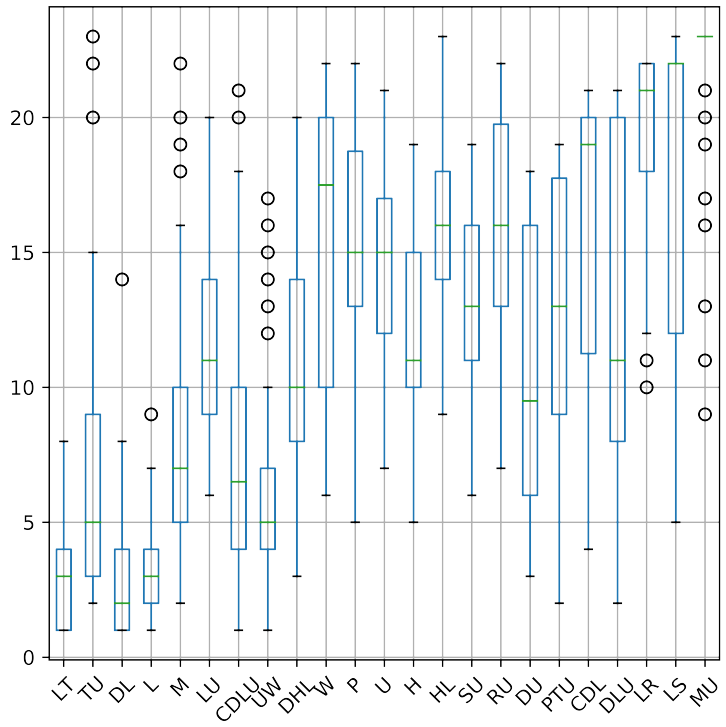


Figure 4.7: A boxplot of the rankings for Set 3.

II Using the prior likelihood of the unique queries within a user group [figure from appendix], a cdf inversion is applied to determine which query ID is the k^{th} query such that $k = 1, ..., n$, where n is from the prior step. The inversion is per the Alias algorithm specified in Section 2.2 of [SM09].

> Given the cdf of the set of queries $\in$ this group, the cdf is in a key-value array of the form:
>
> | Key: | 30 | 2 | 4 | 71 | 16 |
> |---|---|---|---|---|---|
> | Value: | 0.02 | 0.13 | 0.3 | 0.6 | ...≤ 1 |
>
> where keys are query identifiers (that map out to a set of cuboids) and values are *cdf* probabilities.

Then, sample $r \sim U(0, 1)$, where r represents a sampled *value* to retrieve its associated *key*. The array is traversed left to right and the first *key* for which its *value* $\leq r$ is chosen..

III Of the n queries sampled in the previous step, sample the normal distribution to obtain a duration in unit hours; the mean and standard deviation is specified per query in the same Table A.2.

> Given q the query ID, sample duration $d\ of\ q \sim N(\mu_q, \Theta_q)$, and repeat sampling $d(q)$ for as many times as q was sampled in the previous step.

IV Finally $\forall$ instances of q sampled from step 2, the *start hour* of each instance is sampled from the uniform distribution. Specifically, the start of an instance of q is $\sim U(1, t) : t \in 1, ..., 24$; the range of start hours considered is selected for user groups.

Altogether, the prior four steps combine in Algorithm 6; this procedure is run once for each user group of the total four. At the end of the process, it follows that the analyst manually pieces together parts of these four synthetic data sets to synthesize the singular dynamic time series seen in the charts of this chapter.

Next, boxplots in figures 4.5, 4.6, and 4.7 show summary statistics of three data sets whereby the $y - axis$ shares the same units as in the line charts. Sets 2 and 3 have

similar distribution of rank over their periods of time elapsed, however Set 3 has more outlying instances and the variation stretches out further from the interquartile range (the box). Instead, the interquartile range of a cuboid in Set 3 is smaller than a cuboid in Set 2. In contrast, Set 1 is significantly different than the two other data sets. Overall, the least variant set to predict is the first, followed by Set 2, and the most variable data set with the most concept drifts is Set 3, as shown in Figures 4.7.

Data: Table containing the column space (query q, mean μ duration in hours, standard deviation Θ in duration) for a user role.
Result: A set of vectors in the form, (q, start hour, duration, day). It follows to replace q with the set of cuboids using a separate lookup table.

```
1  begin
2      for each day ∈ 1, ..., ℤ⁺ do
3          for each partition ∈ day do
4              /* Each partition of a day is defined by (λ, t), where t
                  marks the amount by which to offset the sampled start
                  time; in this way, different traffic densities could
                  be considered throughout a day.                      */
5              n ⟵ apply(step 1)
6              listEvents ⟵ apply(step 2)
7              /* A histogram holding key-value pairs having q as the
                  key and the number of times it appears in the sampled
                  set as the value.                                    */
8              collectionSummary ⟵ histogram(listEvents)
9              for item in collectionSummary do
10                 durations ⟵ apply(step 3)
11                 startTimes ⟵ apply(step 4) + t
12                 /* the effective start time is offset by t.        */
13             end
14         end
15         days_vector ⟵ append(days, repeat(value = day, n times))
16     end
17     result ⟵ append_columns(listEvents, startTimes, durations, days)
18 end
```

Algorithm 6: Synthesize

4.4 Measures of Correctness and Validation

It was presented in Section 2.2 how the quickening of query execution is evaluated by transitivity through a *key performance indicator* of prediction correctness, that being the ratio of demand met over total demand in one critical scale cycle. The experimentation depends on the critical scale to be the daily scale, followed in queue to materialize by the predictions of the weekly scale. The challenge of meeting cuboid demand expressed by user queries is the objective of all of the approaches experimented on the data sets described in Section 4.3. It should be noted, due to the main memory size constraint, it is not realistic to expect the *kpi* to reach $1 \longrightarrow 100\%$, as it is dependent on the individual sizes of cuboids in the DMs inside the DW. For instance, if every three dimensional cuboid is similar in size (memory units), this size is large enough that two such cuboids could not fit in the allocated memory, and given similar demand exhibited, then the approaches may correctly prioritize their inclusion in the materialized view. However, due to insufficient space, only one cuboid can finally be included; the *kpi* reports the final set of cuboids able to materialize.

In addition, the *kpi* is calculated using the original size of cuboids in a DM instead of estimating what the reduced size *would* be after applying the *smart materialization* to construct the view in main memory specified in [AV16]. This means that the amount of memory occupied by the cuboids selected to materialize is at most (and often less than) what it is conveyed by the *kpi*. The decision to forgo the estimation is brought on by being mindful of the near real-time constraint, since the computation involved to estimate is identical to that of the actual materialization (minus the time to retrieve data from disk). It would require iterating over estimating the actual sizes of the materialized view, given a set of cuboids as the reduction in memory space is fully dependent on the intersection of data between all the dimensions in play, and to then update the materialization pool of cuboids; this would be repeated until the materialized view can no longer be reduced in size enough to allow for additional cuboids to join. Consequently, there are likely more cuboids which could be fit in the materialized view than what is indicated by the results in Chapter 5. In essence, the *kpi* is a deflated measure of what can be materialized according to a main memory constraint from the framework specified in Chapter 3; this aspect is addressed in Section 6.1. Furthermore, the choice of this *kpi* is explained in the following analysis

chapter—Section 5.1.1.

There are 24 experiments to conduct which lead to the evaluation of whether the approaches perform significantly better along the *kpi* measure. Other aspects of comparison, such as frequency of change adaption dependent on false or true positive change detection rates, degree of stability in *kpi*, runtime, and more are covered in Chapter 5. Each distinct data point in the three-dimensional space shown in Figure 4.8 represents one of the 24 experiments run and contrasted in the next chapter.

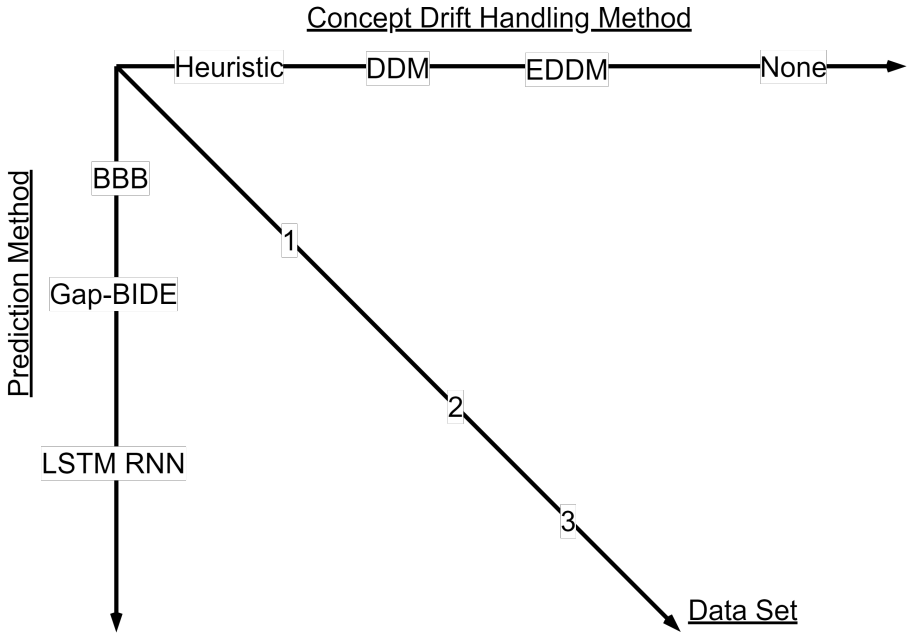


Figure 4.8: The 3-D space depicting the possible combinations of methods, where each combination is a data point is referred to as an *approach* thus far. The axes hold categorical values; no notion of distance or similarity should be inferred from this figure.

The lattice of cuboids is represented in a key-value pair array, storing each cuboid's size in the system's unit of memory. Each approach begins with the same amount of allocated memory, such that:

- total memory allocated for a smart view is $\left\lfloor \frac{1}{3} 5130 \right\rfloor$, where 5130 in MB was the estimated size of running a roll-up query with no filters on the reservations DM requiring $CDLU$ cuboid—this is one of the largest results set among the queries in the profiles explored.

- the day scale is prioritized over the weekly scale, and the system initially reserves $\frac{2}{3}$ of the total memory allocated.
- the remaining third of the allocation is reserved for weekly scale predictions; these reserves are subject to change once the simulation begins according to the approach applied

Other hyper parameters specific to an approach implemented are reported along with simulation results in Chapter 5.

4.5 Conclusion

Altogether, the business cases apparent in both the data released by *AirBNB* and publicly sourced knowledge about their business model lead to the interpretation of four user profiles facilitated by the four DMs constructed earlier in this chapter. The usage of the DM per user profile is exemplified in part in Section A.1 and the motivation for its use is discussed in Section 4.2.1; while these describe user interest in aggregation, Section 4.3 explains the generation of the detailed data-generating process on which the approaches developed in Chapter 3 would be validated in the next chapter.

This chapter provides an overview of the construction of three time series data varying in difficulty along the distribution in instances per unit time, overall number of abrupt changes, degree of slow or steep gradual change, and sheer number of distinct patterns (clusters of cuboids). The synthesis procedure of the time series was motivated by statistical processes fit for distributing events over time for a valid use of synthetic data related to real-world traffic of cuboids in a DW. Next, the performance of the nine approaches mentioned in Chapter 3 and shown in Figure 3.6 are contrasted to determine their role in the ensemble of prediction and change adaption algorithms.

Chapter 5

Analysis

Following the experimentation overview provided in the previous chapter, the results of the experimentation runs described in Figure 3.6 are gathered into a series of five tables in Section B.2 to perform a statistical test of significance on their contrasts. The three measures on which the approaches were evaluated are specified in Section 5.1; these were followed by an analysis of the results on the measures in Section 5.2. An ensemble summarizes the conditional use of certain approaches in concert with the LP-CDA framework.

5.1 Evaluation Measures

The analysis of experimentation results involved three metrics measured at the end of each time step of the simulation (iteration of Algorithm 1): the predictive *kpi* defined in Section 2.4 and reviewed in Section 4.4, the degree of change in memory allocated to the predictions formed on the critical scale as a result of change adaption, and the rate of changes detected that initiate the adaption procedure. These measures are of interest since the first addresses the percent of cuboid demand correctly predicted within a time interval, the second focuses on the amount of change on the system's side required to recover from below threshold predictive *correctness*, and lastly, the pace at which detection flags are raised which reflects how long the recovery lasted

since the most recent adaption procedure instance. The three evaluation measures are covered in the following subsections.

5.1.1 Predictive KPI

The key performance indicator (*kpi*) to evaluate the predicted set of cuboids per cycle of the critical scale is a semblance of demand, defined as the ratio between the number of times each cuboid requirement could be pulled from the materialized set versus if it required an export of data from permanent storage. The *weight* of each cuboid is the number of times it is called into action within a query join, with the weight varying per demand in each time cycle. If a cuboid requires x times within a given cycle and is covered in the materialized set, then all x instances are accounted for as having benefited from a materialized set. The *kpi* is specified in Equation 3.7 and described at a high level in Equation 3.1; it is the percent amount of correctly predicted cuboids per critical time scale cycle.

Next, the choice of the *percentage correct* measure is described over other more robust performance measures often used in modern day ML and statistical analyses.

Choice of Percentage Correct Predictions as a Measure

The *Percentage Correct Prediction (PCP)* [Her99] measure of a predictive method's performance is often not the first choice among the F-score, area under the Receiver Operator Curve (ROC), Matthew's correlation coefficient synonymous to Pearson's phi coefficient, Cohen's Kappa, and many more often cited as *robust* measures [Mau12]. Specifically, this $\textit{percentage correct prediction} = \frac{TP}{TP + FP}$ and is referred to as the *positive predictive value* in recent literature [Mau12]; it is the *kpi* measure used in this thesis on which many decisions of adaption are based. It primarily lacks in using all the data available from the result of classifying test data, as only the true and false positive metrics of the confusion matrix are used, and true and false negative metrics are omitted from the *kpi*. Consequently, the author of [Her99] demonstrates the lack of coverage in predictive performance by the PCP in regression models and this observation has extended to many modern day classifi-

cation methods highlighted in Chapter 2 and 3. In addition, the advantage observed by [Her99] of using PCP has also remained constant throughout the development of many other performance measures in ML. When comparing predictive methods from a variety of modelling families, as is done in this thesis, not all alternative measures of performance may apply [Her99].

In addition, regarding the dependency on predictive methods, a performance measure's suitability is also a factor of the data under investigation in terms of the stream or static nature of the data, types of concept drift, and biases inherent with certain measures. For instance, not all uses of the F-Measure outside the information retrieval domain are sound, as exemplified in [Pow15]. In addition, there is the area under the ROC measure that is appropriate in situations aside its intended use in the signal processing domain despite its overuse with other types of data as examined in the clinical setting by the authors of [BF11]. The PCP as a general measure can fit with many data and algorithm types; however, it is also an overestimate of the prediction performance of methods [Her99, Mau12].

If a more robust performance measure had been chosen, it would have been as a consequence of the particular data set and prediction method applied. In other words, the performance measure fits the type of data on which it is evaluated and the analysis model which is applied to yield the predictions under scrutiny. Given the situation in Chapter 4 of inferring data marts and supposing the query profiles to then synthesize the time series of the events which occur within the boundaries of the defined data marts and query profiles, we are left with many options to synthesize such a time series, and the breadth of concept drifts (abrupt, outlying, gradual) interwoven to span all possible sets would not yield similar properties. For example in Chapter 4 between Sets 1 and 3; Set 1 is sparse and contains bursts of high traffic activity whereas Set 3 is dense and contains many concept drifts (sometimes more than one at a time). As a result, it is more difficult for algorithms to discern the tendency of the data process in the future against what may be a temporary blip. The three data sets represent three ways of synthesizing data from sources and sharing different properties where each would be best fit with a performance method and prediction method different from the other. Thus, the choice of a robust and more focused performance measure would require a justification that, as a result of the data marts and query profiles, only certain types of data may be of consequence and a more specific and robust

performance measure other than PCP would be preferable. The data generation process in Chapter 4 and the LP-CDA framework developed in Chapter 3 counters the prerequisite for choosing a less flexible and more robust performance measure. Since the predictive methods applied are of different families and in each case the response variable to predict has been developed uniquely to the method—the procedure from reading in the data to initializing a predictive method is different among all three methods, and so too is the interpretation of predictions into a ranked set of events. In addition, the three time series represent three possibilities (out of many more plausible) with already different properties befitting various predictive measures (many highlighted in [Fla12, Mau12].

Finally, the PCP is a measure of the true or false positives which are of principal interest given the intention to place most frequent items in the materialized set. Other measures which are closely related such as specificity, sensitivity, negative predictive value, and accuracy involve the accounting of negative results. However, the cuboids of varying frequency are not strictly wrong relative to others. LP-CDA will aim to fill in as much of the allocated space for materialization, despite how few in-demand cuboids are; as a result, it is likely for the items more generally thought of as "negative" results to be selected into the positive class. Furthermore, there is no decision threshold established to divide the ranked list of items into positive/negative classes.

For the above reasons, the *kpi* at this stage in the work is the PCP; with reasons covered in [Fla12, Mau12, JWHT13], it is desirable to move towards using robust measures which fit the dynamic nature of the framework (type of data and prediction method). The latter part of the prior statement is left for future work, in which a a question to be answered could be: does an ensemble of performance measures succeed to deliver robust prediction evaluations while appropriately changing for types of data and diverse set of prediction methods?

5.1.2 Change in Memory Allocation

The initial memory allocated between the time scales for all predictive models is identical across approaches, as indicated in Section 4.4; however, the amount allo-

cated may change according to the difference in *kpi* and static threshold T defined in Chapter 3 over time. The timing of the change depends on the rate of detected change (Section 5.1.3) as per one of the three change detection methods specified in Section 2.4. The memory de-allocated from all noncritical time scales is governed by the adaption strategy, and is often a proportion of the difference in *kpi* and T at the time an adaption procedure is executed. It is of interest to measure how (not) drastically an approach re-allocates memory away from other time scales and prioritizes the critical time scale's predictions to materialize, as this indicates the low confidence in these other predictive models. When the change is minimal (close to zero), then both daily and weekly predictive models contribute to a successful performant (*kpi* wise) materialized set, otherwise the long-term patterns or trends captured by the weekly model are no longer relevant and only the daily model captures relevant patterns.

5.1.3 Rate of Change Detection

When a change is detected by a drift detection method from Section 2.4, a Boolean value is switched to 1 from its default 0 value and reset at the start of each critical scale cycle as per Algorithm 1; this value is recorded for each approach at each cycle of the simulation to contrast across all approaches. It is expected that different detection methods will detect changes at a different pace as not all are well suited for the many types of drift examined in Section 2.4. An approach with a higher rate of detected change indicates an unstable pairing of predictive method and change detection method compared to its peers in Figure 3.6.

5.2 Results

In this section, simulation logs at the end of each experiment (approach per data set as in Figure 4.8) are filtered for the measures of interest outlined in Section 5.1 to enable an analysis; from this analysis appropriate conclusions are drawn. For instance, the first three figures in this section show a comparison of the *kpi* measure for all three data sets (Section 4.3), followed by a figure comparing the increase in memory allotted to the critical scale of prediction as a result of adapting to change, and a comparison of

the rate of changes detected. The Nemenyi post-hoc statistical test is applied rather than using the more popular measures for algorithm comparison regarding predictive ability evaluated by measures such as: F-measure, accuracy and recall, area under the receiver operator curve. The diagrams in this section are critical difference (CD) diagrams that result from the Nemenyi post-hoc test, from which conclusions are drawn.

Given the diverse set of algorithms used in composition of the ensemble, their underlying assumptions also vary; due to this, a non-parametric test of significance leads to more valid inferences than a parametric test. Rather than describing the similarities among the approaches in terms of predictive performance, the analysis focused on their differences. Then, revealing how the approaches contrast along *kpi*, memory allocation between scales of prediction, and the rate of change detection/adaption through the Nemenyi post hoc test leads to conclusive decisions on when to use a specific approach in a DW setting.

The use of Friedman and Nemenyi tests from an instructional level are covered in [Fla12]. The average rank is deduced from the average measure over all of the data sets, which in the context of the experimentation here, is the average over all the time steps (mutually exclusive subset of a data set). As defined in the text, the Nemenyi test calculates the critical difference as follows: $[CD = q_a\sqrt{\frac{k(k+1)}{6n}}]$ where q_a depends on the significance level α as well as k number of approaches (12 approaches per data set simulated, as shown in Section 4.3). A rank of 1 indicates *highest on average* on the dimension of performance in question; these are plotted on the horizontal axis of the critical distance graphs in this section. Ranked approaches are also grouped (indicated by the thick lines below the horizontal axis) if they are less than the critical distance apart from each other consecutively; this provides a threshold to determine which approaches are *different enough* to consider or implicitly too similar to distinguish between. In other words, any pair of approaches which are not connected by a thick horizontal line can be considered statistically different from each other.

Since the inputs to the Nemenyi test are measures per approach (column wise) at each time step of the data set simulated (row wise), the degrees of freedom are high enough to make conclusions that relate to other similar data processes outside this

thesis; these input tables may be inspected in Section B.2. In the context of the significance testing, each day in the simulation is a set of its own; thus, a data set with 99 days worth of events means there are 99 subsets to measure and against which to compare the approaches. The number of degrees of freedom matters in the age of ML and *applied AI* as much as it does in traditional statistical inference. In validating the worth of the algorithms and approaches, it is not just the quality of the data sets that matters—the number of data sets is crucial as well; this is the conclusion reached in [BH17].

5.2.1 Predictive KPI Analysis

Specifically, the Nemenyi critical distance is determined for $\alpha = 0.05$ and a 95% significance, and is reported in Figure 5.1. It can be seen that the decrease in CD from Set 1 to Set 3 agrees with the forecast increase in difficulty from the first data set to the third discussed in Section 4.3. In other words, the differences in average ranks of the approaches are most apparent in Set 3.

```
set 1
Critical difference = 3.2941, k = 12, df = 300

set 2
Critical difference = 2.8751, k = 12, df = 396

set 3
Critical difference = 1.7133, k = 12, df = 1128

set 3 space comparisons
Critical difference = 1.7133, k = 12, df = 1128

set 3 change detection flag (0/1)
Critical difference = 1.3932, k = 10, df = 940
```

Figure 5.1: The critical differences calculated for five tables comparing the *kpi*, memory allocation redistribution, and rate of change detected. Where the critical difference is in unit *average rank*, k is the number of approaches, and *df* are the degrees of freedom.

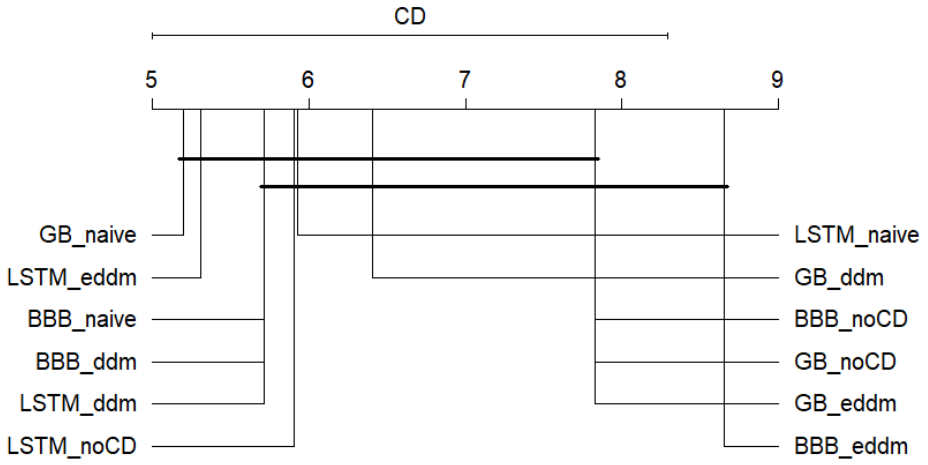


Figure 5.2: The critical difference plot for the Nemenyi post-hoc statistical test on all approaches' kpi simulated on Set 1, as shown in Figure 4.5

The **Set 2** results are reported in Figure 5.3. Three groups of similar performing (on the basis of their *kpi* measures) approaches are indicated by the three bold horizontal lines in the figure. The higher on the vertical axis a horizontal bold line is, the higher in average ranks that group is compared to the other bold lines. Of note, *Gap-BIDE* and $BBB - noCD$ (noCD referring to *no change detection*) are in the top-ranked group; this implies that approaches using no change detection or $EDDM$ tend to higher *kpi* on average than approaches using naive heuristic or DDM methods.

Set 3 yields the smallest CD between the approaches of Sets 1 and 2 in figures 5.2 and 5.3, respectively, indicating that the performance is much more distinguishable in the third data set between the approaches than in the second set. In addition, there are many more groups in Figure 5.4 than in previous CD diagrams in this section, as evidenced by more bold horizontal lines with less overlap between them. This affirms the expectation made in 4.3 that Set 3 would be the most difficult to predict; its results indicate the approaches' ability to keep up with changes in cuboid demand. Similarly, as with Set 2, the *naive* heuristic-based and $EDDM$ change detection methods are in the top two groups of average *kpi* rank ($kpi \rightarrow 1$) out of

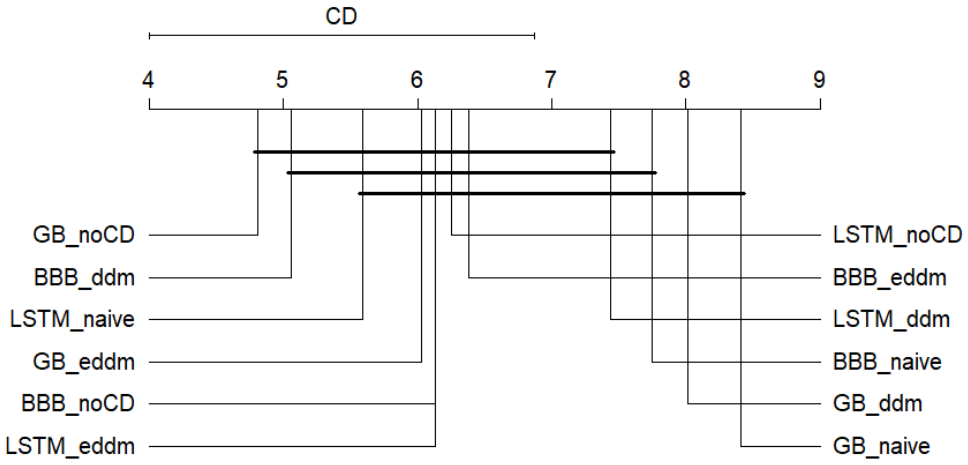


Figure 5.3: The critical difference plot for the Nemenyi post-hoc statistical test on all approaches' kpi simulated on Set 2, as shown in Figure 4.6

six groups. However, $Gap - BIDE$ is in the bottom half of the rankings; this is in contrast to its placement in the previous two data sets. The top performing groups include BBB using naive or DDM change detection, and $LSTM$ using naive or *no* change detection method. Surprisingly, an approach with no change detection and adaption strategy is in the top ranked group out of the many alternatives; Figure 5.6 may lend an explanation for this. From the CD diagram, it is a clear choice between BBB or $LSTM$ as the most reliable prediction method. The next two CD diagrams should lend evidence to guide our choice of the change detection algorithm to couple with these two prediction methods.

5.2.2 Change in Memory Allocation Analysis

In Figure 5.5 all $noCD$ approaches are ranked last as they have no variation in the amount of memory allocated between the predictions scales; variation is due to the change detection rate (false and true positives) and adaption strategies. The $noCD$ instances are the baseline against which to compare; in the previous three diagrams,

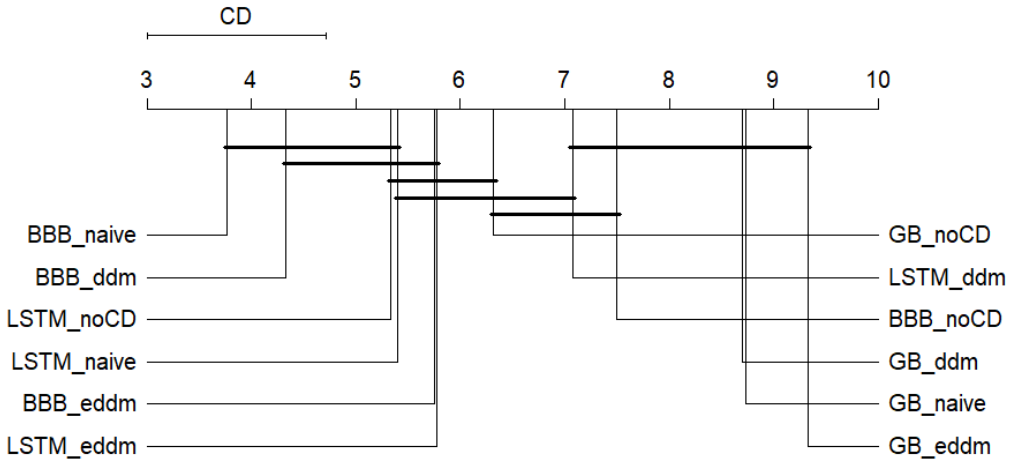


Figure 5.4: The critical difference plot for the Nemenyi post-hoc statistical test on all approaches' kpi simulated on Set 3, as shown in Figure 4.7.

the CD was relative to the entire collection of approaches. The top ranked approaches are those that deviated the more often and with a larger amount of memory than all other approaches. It can be seen that $LSTM$ with DDM and $Gap - BIDE$ with either naive or DDM had the highest amount/frequency increase in memory allocated to the daily scale which results in a proportional decrease in memory allocated to the weekly scale's predictions. Additionally, the prior observation implies the higher the difference between the two prediction scales in memory allocated to them, the higher the reliability of one scale's prediction over the other. In this thesis' framework, trust is lost in the weekly scale's prediction as change is detected and a more drastic adaption action taken in favour of the daily scale's predictions, as discussed in Chapter 3. Crucially, we focus on the approaches that placed among the top groups in terms of *kpi* measures in Figure 5.4, where BBB and $LSTM$ using $EDDM$ are grouped together in the last ranked grouping in Figure 5.5. This means their variation in memory is critically different from all other change detection variants in achieving the middle rank with respect to *kpi*. Instead, BBB with DDM or naive are grouped with $LSTM - naive$ while the latter is also part of the group with $LSTM - DDM$;

the two variants of BBB are critically different from $LSTM-DDM$. Before drawing conclusions on the change detection methods to use, the CD diagram regarding the detection rate should be considered.

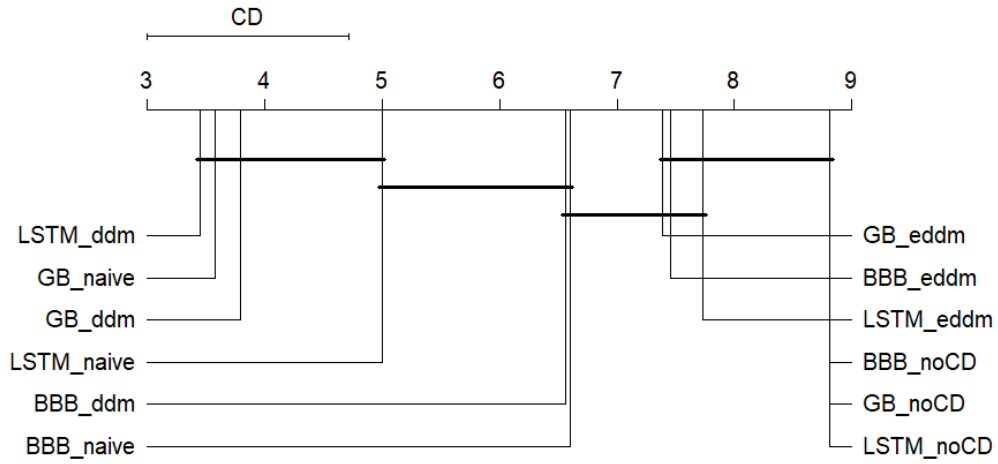


Figure 5.5: The critical difference plot for the Nemenyi post-hoc statistical test on all approaches' critical scale memory allocations as a result of change adaption simulated on Set 3.

5.2.3 Change Detection Analysis

Next, in Figure 5.6, the $noCD$ approach has a sequence of 0 values at every time step indicating no change was detected; this is the baseline approach against which to compare the alternatives. From the diagram, it can be seen that not all approaches using $EDDM$ are in the same group as $noCD$, indicating there is little difference in the true/false positive detection rate between $noCD$ (detection rate is a flat 0) and $EDDM$. In addition to the observations in figures 5.5 and 5.4 regarding $EDDM$, this leads to the conclusion that BB or $LSTM$ using this change detection method are the most stable in terms of predictive ability, frequency of change detected, and the amount of adaption needed to achieve predictive accuracy above the (75%) threshold.

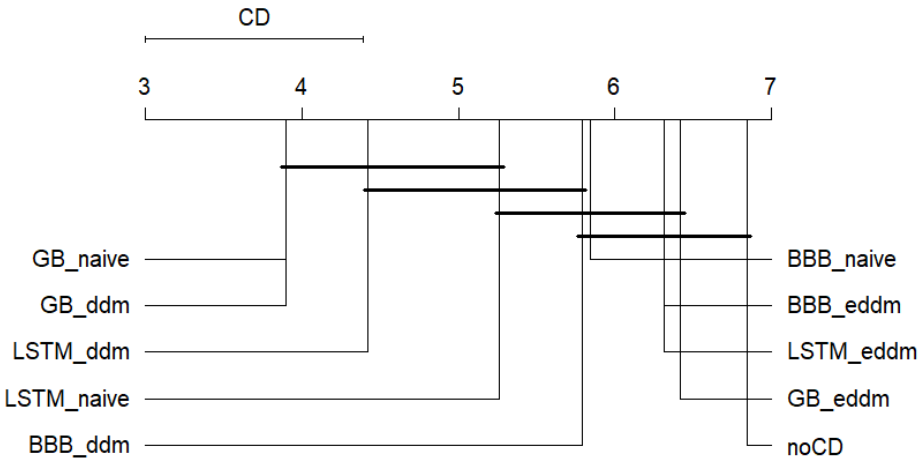


Figure 5.6: The critical difference plot for the Nemenyi post-hoc statistical test on all approaches' change detection flag simulated on Set 3.

5.3 Ensemble Learning: When to Use an Approach?

An ensemble can be constructed from the evidence gathered in 5.2 and in combination with the goals outlined in Section 2 in order to select an approach to employ. In terms of a conclusion regarding whether an alternative hypothesis is accepted over the null hypothesis, as noted in the many data analysis texts referenced throughout this thesis, data analysis is performed in the context of the data, and thus conclusions are conditional on the data. In addition to the context of data, the context of hardware and DBMS system used by an organization in its day-to-day work should be considered. While we cannot account for the many variations of current and future data warehousing architectures, we can offer guidance that administrators of the architectures should consider in selecting an approach from those tested.

Different priorities yield a different conclusive approach to implement, as shown in Figure 5.7; however, swapping a prediction or change detection method later is feasible

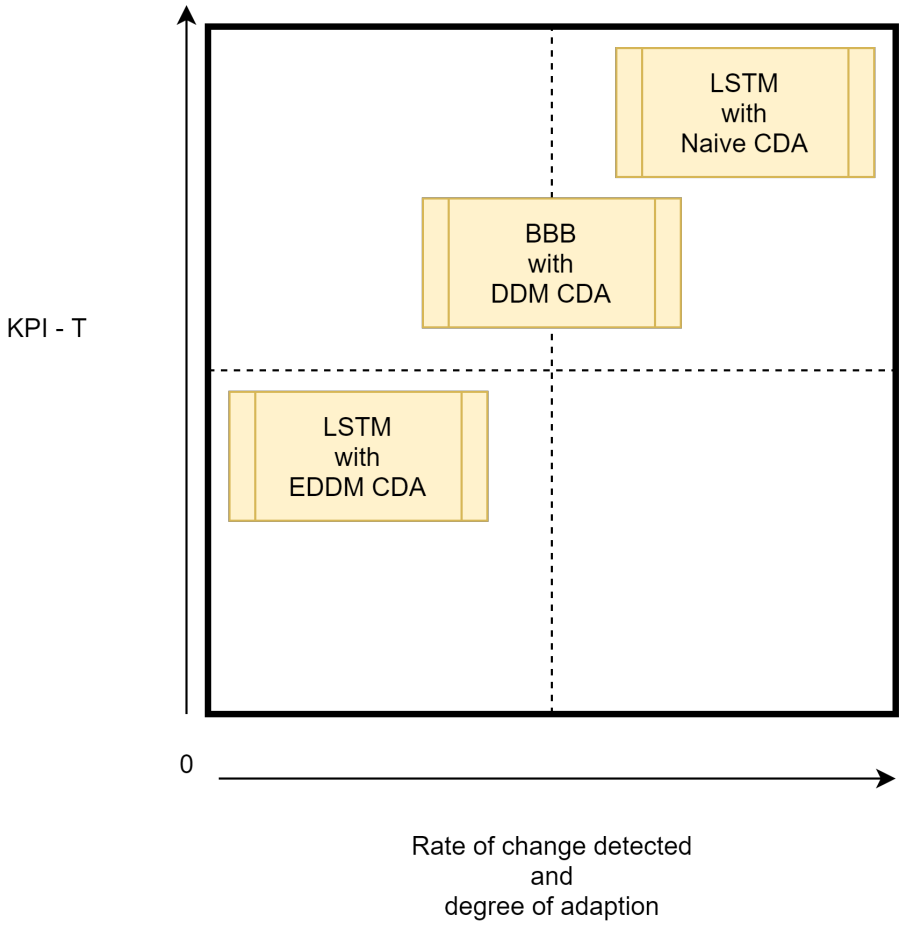


Figure 5.7: A qualitative summary of the approaches to carry into the ensemble. These three approaches were selected by inspecting the five critical difference graphs.

given the framework specifications in Chapter 3. While it is more complex to change the prediction method, it is much simpler to switch the change detection and adaption methods, based on the experience of implementation in this work. Scenarios should be considered in which the sequences of priorities shown in Figure 5.7 occur in an organization's production work.

From the resulting critical difference graphs in the prior section, we can reduce the number of candidate approaches along the three measures evaluated:

I per *kpi* performance, approaches which are within the top performing group consistently between Sets 2 and 3. These two sets are prioritized over Set 1since they are distinctly more complex and were not used as part of any initial

calibration in parameter selection of algorithms tested. See figures 5.3 and 5.4.

- *BBB - DDM*
- *LSTM - Naive*
- *LSTM - EDDM*

II per detection rate, relative to no CDA employed; Figure 5.6.

- *LSTM - EDDM* is equivalent to almost no change detected, despite achieving performant *kpi* results.
- *BBB - DDM* is part of the mid tier for this measure.
- *LSTM - Naive* is a member of both the top tier and mid tier.

III per change in memory allocation, a high degree of correlation with change detect rate is expected. See Figure 5.5.

- *LSTM - EDDM* fully correlated to its place in detection rate graph.
- *BBB - DDM*, similarly is in the mid tier.
- *LSTM - Naive* is a member of both the top tier and mid tier

In an effort to combine these approaches, we look to the ensemble learning domain within the ML field. "Ensemble learning can be broken down into two tasks: developing a population of base learners form training data, and then combining them to form a composite predictor" [HTF01]. Since there was not a single approach which scored high on all three measures consistently, it follows that a cooperative formulation may draw out their advantages and mitigate their disadvantages. For instance, a high detection rate impacts the server resources by having to focus on re-composing the materialized set and perhaps too often occupying CPU and context switching between critical tasks within the operating system away from user- (client) related queries. In the prior case, one might prefer LSTM - EDDM approaches over BBB - DDM and LSTM.

The goal of this study is to learn a *good ensemble*, and that has been achieved by way of the analysis on the results of the candidate approaches. "In terms of basis functions, we want a collection that covers the space well in places where they are needed, and are

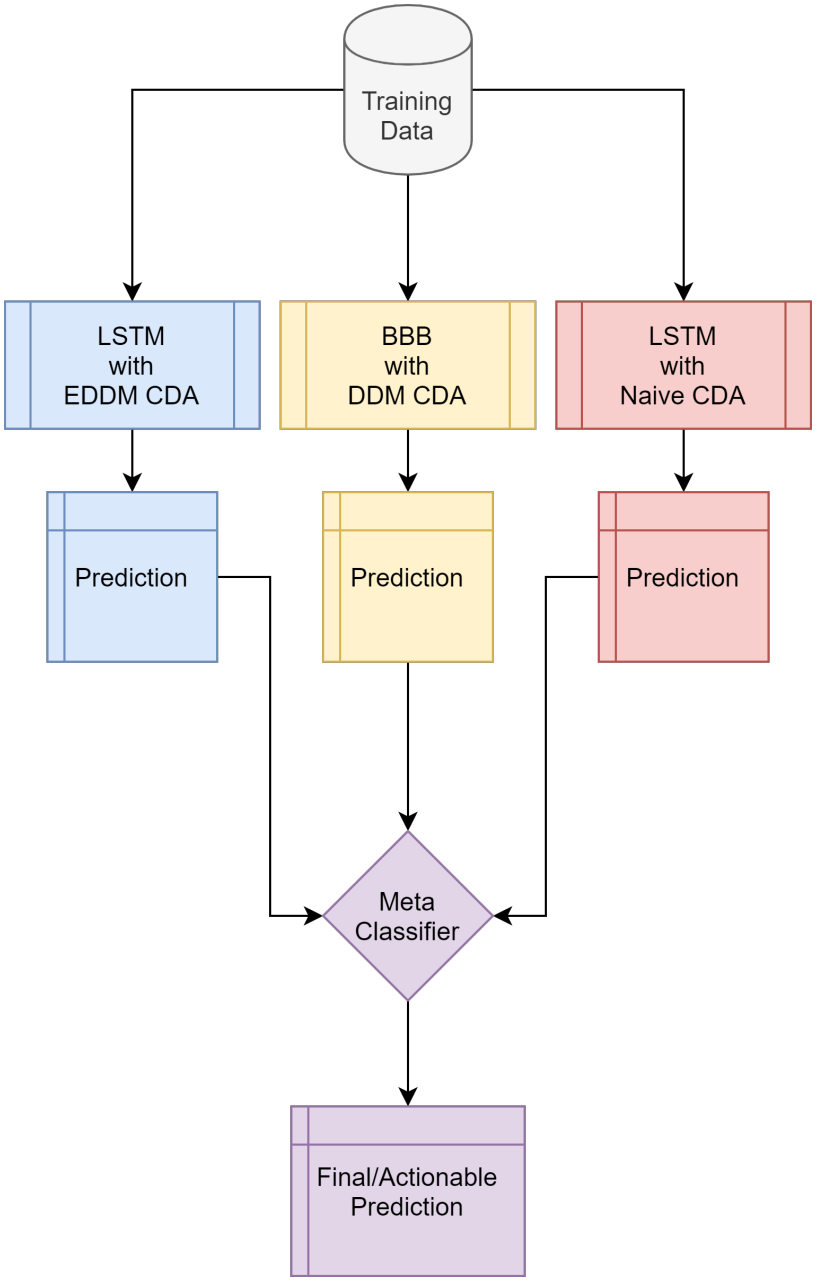


Figure 5.8: A stacking of select approaches to form a winner-take-all ensemble. The logic within the meta-classifier decides which approach's prediction to carry over as the final decision defining the materialization set.

sufficiently different from each other for the post-processor to be effective" [HTF01]. This statement refers to ensemble learners given regression methods for classifications, and while it is not identically reflective of the situation in this thesis, it is relevant. For instance, it is clear that the diverse families of algorithms that have been utilized to form the approaches in the experimentation are *sufficiently different from each other*, to the extent that not many of the ensemble learning configurations remain applicable. In addition, the *post-processor* referred to the lasso or regularization used in regression methods to better fit the model to the variation in the data and reduce bias—instead it is synonymous to the *meta-classifier* component to post-process the predictions of the three approaches. Of the many ensemble learning configurations available, including cascading classifiers, stacking, voting, bagging, and boosting, a hybrid of stacking and boosting is preferred for the scenario described in this thesis. Stacking the three approaches shown in Figure 5.7 and discussed above, enables the flexibility of using *sufficiently* different approaches for the same feature space (data relations to model).

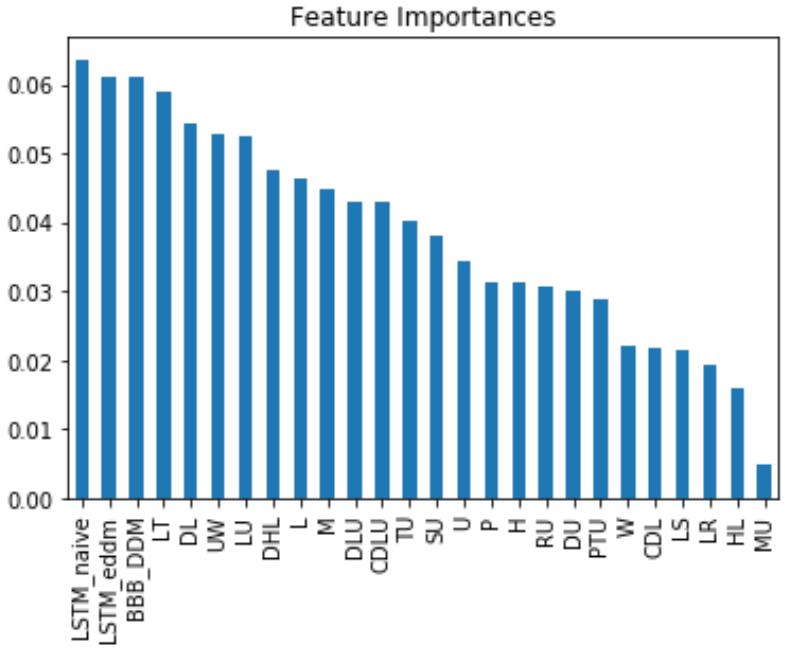


Figure 5.9: The importance assigned the features by the Gradient Boosting Regressor modelled per configuration A.9 outlined in Section 5.3. The y-axis represents the Gini importance measure where all values sum to 1. It is a normalized measure of the total reduction in the Gradient Boosting Regressor (GBR) model's criterion attributed to each feature.

The three identified approaches in this section form an ensemble through a stacked configuration, as depicted in Figure 5.8. The meta-classifier in this case is a GBR implemented in the Python programming Library [PVG+11], whereby a set of decision trees can be inferred from training on the data presented in Section B.1. The choice of GBR over traditional boosting methods such as AdaBoost is supported as early as [MBBF99]. It is further supported by the development of schemes using gradient boosting in the ML Python library *SciKit Learn* referenced in this section and in 4, funded by leading industry members using ML in day-to-day operations.

Boosting updates the weights of training tuples so that their skewed contribution to the residual will help tune a weak learner to model the misclassified tuples over some iterations. In the iterative optimization procedure shown in [JWHT13], the collection of stumps (decision trees of depth ≤ 3) aim to minimize the residual (prediction errors). In addition, the prediction of the model (ensemble of decision stumps) is indicated by the majority vote among all weak learners (each stump) weighed by their individual accuracy value. The prior point refers to AdaBoost, where boosting touches on the coefficients of tuples with respect to their contribution to the residual computation; however, finalizing the singular prediction of the response variable is determined by a vote (another form of ensemble learning).In contrast, GBR operates on the parameters of the decision trees (these can typically have a depth up eight levels) in order to minimize the residual, however, this is guided by using the gradient descent optimization on a loss function which is often different than those used in boosting methods [MBBF99]. Thus, boosting attempts to achieve the goal of minimal predictions errors with respect to training data indirectly by changing the weight of tuples to signify importance for the model to learn; directly, GBR tunes the parameters of each decision tree in the iteration to learn the next most rewarding (minimizing errors) group of tuples from the last learned tree in sequence.

Both boosting and GBR-based methods produce a collection of regression (specific to our application of these concepts) trees, and one tree is used per iteration of the algorithm. Although either boosting algorithm may be applied to any classifier or regression technique, it is beneficial to borrow the flexibility of the regression tree given we aim to predict a continuous variable (kpi) and expect a high degree of variability in the feature space that is otherwise very difficult with spline or other parametric regression methods; the regression tree is a non-parametric method. The

use of "collection" instead of "ensemble" is only to draw distinction between the three approaches shown in Figure 5.8 and the separate set of learners in the boosting method. The boosting method begins with equal weighing of each example or tuple in the training set to which the first decision stump is fit. Then, the residual (difference in y and $\hat{y}$) is calculated to quantify the shift in weight needed in order give higher importance to the incorrectly classified tuples. The distribution, in turn, affects the next decision stump to be fit in the iteration.

Distinctly, GBR relies on minimizing a loss function for which it is at its peak (of 100%) *cost* when a model yields no reward; whereas a model that can yield a cost of 0 (or 0%) leads to no prediction error. The latter case is unrealistic as it would imply the predictor (features) to response (classification label) relationship is not-complex and would not be of much interest to model; say $f(x) = 1 : x \in \mathbb{R}$. The loss function relates the residuals to the model's coefficients, as in the cost on the residual (and thus the proportion of reward - correct predictions) given a fit regression tree. The iterative minimizing of the loss function, involves:

- Calculating the derivative on the loss function to determine the *direction* towards the global minimum, often referred to as the negative gradient (downward slope) as opposed to the function's positive gradient (upward slope) to achieve high cost.
- Setting the learning rate between 0 and 1, this factor limits how much the coefficients can be updated (differ) from the last fit tree; a typical value is 0.1.

Altogether, the GBR method calculates the combination of regression tree coefficients that yields the steepest descent (greedy approach) within the learning rate constraint, iteratively over n steps. A formulaic representation of the gradient descent is found in [Fri99].

An overview of the GBR model employed as per configuration A.10 in Section B.1:

- Loss function chosen was the *least absolute deviation* among least squares, Huber, and quantile estimation as it is a more robust measure. Given the expectation that the data process will have a high variance, a robust measure is desirable to ensure the quality of predictions carry over to related data sets.

- The Friedman mean square error is minimized by each of the 300 estimators (regression trees).
- All other parameters reported in the appendix were determined by iterative experimentation to yield a high regression goodness of fit R^2 measure.

The R^2 measure is reported as an evaluation of the goodness of fit of the meta-classifier regression model, it is at 70%. It was evaluated on Set 3 via the data table shown in Section B.2, and includes the number of instances of cuboids (each their own column) per day, the time axis represented in 3 ways (day, week, iteration), the three *kpi* measures of the base learners listed in this section, and the response variable "MAX" *kpi*. The data set was first stratified along the response variable , y, as it appears in Section B.1 and "MAX" in the last table of Section B.2. It was then split into a test set containing 30% of the data, and the rest was used as training for the GBR model. The R^2 values reported in Section B.1 were not produced using a random state parameter because this would allow for both reproducing of the exact GBR tree and test-training data split, and enable the biased choice of random states which yield R^2 in the high 80s and above in extreme cases; this practice would be misleading.

A feature selection step also occurred between the two configurations of the stacked regressor, whereby features of less importance shown in Figure 5.9 were removed to construct the A.10 model (Section B.1). The result of A.10 was a better fit to the data set and a change in important features shown in Figure 5.10. It is evident that the decision trees within the ensemble of trees constructed using GBR had difficulty splitting on values to yield accurate predictions with the full set of features and relied mainly on the base learner *kpi* measure. Instead, with a subset of features, the A.10 configuration made better use of the features to form scenarios in which one base learner (the approach explored in this thesis) is better positioned to reach a higher *kpi* than other learners represented.

Therefore, the progression of time in the time series is a key indicator in combination with historic *kpi* data of the predictors. In particular, from Figure 5.10, the LSTM-EDDM approach contributes more by achieving higher *kpi* measurements than $BBB - DDM$; $LSTM - Naive$ are a close second choice. In addition, the cuboids which most represent the scenarios used by the GBR regressor in selecting the base

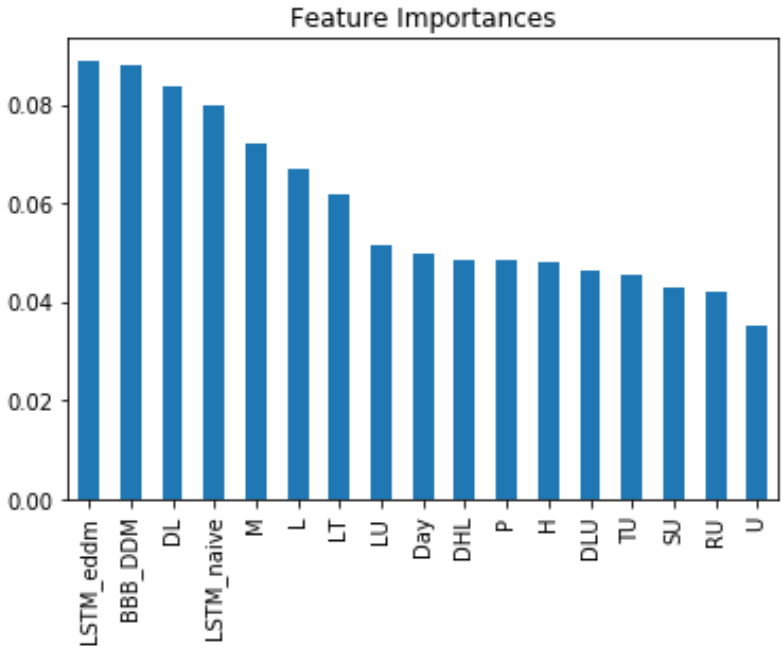


Figure 5.10: The importance assigned to the features by the Gradient Boosting Regressor modelled per configuration A.10 outlined in Section 5.3; a subset of that used in A.9. The y-axis represents the Gini importance measure where all values sum to 1. It is a normalized measure of the total reduction in the GBR model's criterion attributed to each feature.

learner are shown on the horizontal axis of the figure referenced. Critically, the diagram also highlights that the day of the week is of little importance; this is a symptom of the concept drifts within the time series of cuboids in demand, and it follows that the progression in weeks was not important since there was not a strong cyclical pattern of cuboid demand at this time scale. The final interpretation of the stacking ensemble results is that $LSTM - EDDM$ is preferred over all other approaches in terms of *kpi* measure given the many concept drifts under investigation, and that $BBB - DDM$ is a close second choice, particularly if a simpler model in terms of complexity and runtime is required by the user.

5.4 Conclusion

In this chapter, the efficacy of the 12 approaches, three predictive methods and four change detection-adaption methods, were contrasted on three time series. Altogether, the Gap-Bide and BBB prediction methods operated on the order of seconds whereas LSTM RNN was on the order of a few minutes for any of the change detection-adaption procedures. In Figure 5.7, the conclusive roles of approaches in the ensemble are depicted and supported by the ranked test of statistical significance in Section 5.2. Furthermore, the choice between $LSTM - EDDM$ and $BBB - DDM$ becomes clearer through the analysis of feature importance resulting from implementing a stacked ensemble of regression trees using the GBR algorithm. In the final chapter, a summary is presented of the contributions of this work and directions for future work.

Chapter 6

Conclusion

This thesis concerned the intelligent adaptation to changes in user query patterns over time. We introduced the LP-CDA framework that employed ML algorithms in order to predict such changes across multiple time scales. Our experimental results indicated the value of our methodology. Our results show that an ensemble of approaches, each a combination of a prediction and change detection-adaption methods, can achieve a robust positive prediction rate $\geq 75\%$ amidst various types of concept drift often asynchronous and overlapping (with respect to time) of each over.

6.1 Future Work

There are several extensions of the framework developed in this thesis which would lend to better predictions in more complex modelling scenarios. First , many approaches surveyed in Chapter 2 are based on user profiling and assume the profiles can be used to group interests in data for querying to build materialized views of some form. However, the assumptions of number of groups, members of the group, and the mutual exclusive membership to a single group are biases which would contradict the data-driven approach employed in this thesis. Instead, using unsupervised clustering methods to specify that which is previously assumed will allow any benefits of grouping observations (often proved to be beneficial in mixed effects modelling as

in [FEH15b]) to be incurred by the framework. The clustering method must also account for concept drift.

Second, there is evidence of the minority class problem defined in the data mining field [Han15] and carried over into modern ML, in which certain cuboids will be considered in low demand given their relative frequency to all other cuboids in a time period. However, some cuboids of low frequency may be related to complex (runtime wise) queries, thus these cuboids will not be selected with priority to join the smart view. An element of the resolution for this minority class problem may be clustering users, given a correlation between complex and less frequent queries and the group of users is true.

Third, to predict along the row subset in addition to the dimensions, as in [AV16], the authors of that article provide an example of the smart materialization algorithm segmenting data row wise in addition to columns. The row wise materialization is avoided in the initial work of this thesis since the consequence of an incorrect prediction on the rows renders a materialized cuboid unusable by most DBMS in resolving a query requiring any rows not in the materialized set (cause for errors). A focus on exploring approaches to predict the row space demand per cuboid is a valuable progression of the framework developed in this thesis.

Fourth, improvements to the quality of the LSTM RNN can be achieved by seeking a stable variant of the model. Stability is not usually inherent to the LSTM model, although some authors in recent work [MH19] have shown empirical solutions to either replace RNNs with feed-forward models or *impose stability* by modifying parameters of the RNN to achieve *soft-isometry* (a condition of stability for RNN). In addition, it would be beneficial to build a *Markov Chain* analysis then estimate through Monte Carlo simulations [SM09] the future demand of cuboids and to compare the results with the current LP-CDA framework.

Finally, an addition to the ensemble built in this thesis may be to use mixed-effects modelling given identified clusters of cuboid patterns. This may be a competitive alternative to the LSTM-RNN method in terms of complexity to capture dependencies between events aside from time progression.

References

[AE20] Luk Arbuckle and Khaled El Emam. *Building an Anonymization Pipeline: Creating Safe Data*. O'Reilly Media, 2020.

[AGM+11] Julien Aligon, Matteo Golfarelli, Patrick Marcel, Stefano Rizzi, and Elisa Turricchia. Mining preferences from olap query logs for proactive personalization. pages 84–97, 01 2011.

[Air15] AirBNB. Airbnb new user bookings. Kaggle Online Data Sharing Portal, 2015.

[AS15] Saida Aissi and Mohamed S. Gouider Schleicher. Towards the next generation of data warehouse personalization system, a survey and a comparative study. 2015.

[AV16] Daniel K. Antwi and Herna L. Viktor. Dynamic materialization for building personalized smart cubes. *Transactions on Large-Scale Data- and Knowledge-Centered Systems XXVI: Special Issue on Data Warehousing and Knowledge Discovery*, pages 61–88, 2016.

[BF11] Daniel Berrar and Peter Flach. Caveats and pitfalls of ROC analysis in clinical microarray research (and how to avoid them). *Briefings in Bioinformatics*, 13(1):83–97, 03 2011.

[BGCAF+06] Manuel Baena-Garcia, Jose Del Campo-Avila, Raul Fidalgo, Albert Bifet, Ricard Gavalda, and Rafael Morales-Bueno. Early drift detection method. *Fourth International Workshop on Knowledge Discovery from Data Streams*, 2006.

[BGR11] P. Biondi, M. Golfarelli, and S. Rizzi. Preference-based datacube analysis with myolap. *ICDE*, 3:1328–1331, 2011.

[BH17] Anne-Laure Boulesteix and Myriam Hatz. Benchmarking for clustering methods based on real data: A statistical view. *Data Science, Studies in Classification, Data Analysis, and Knowledge Organization*, pages 73–82, 2017.

[BS13] Denni D. Boos and L. A. Stefanski. *Essential Statistical Inference: Theory and Methods*, chapter Bayesian Inference, pages 163–203. Springer New York, New York, NY, 2013.

[DB16] E.B. Dagum and S. Bianconcini. *Seasonal Adjustment Methods and Real Time Trend-Cycle Estimation*. Statistics for Social and Behavioral Sciences. Springer International Publishing, 2016.

[Ema13] Khaled El Emam. *Guide to the De-Identification of Personal Health Information*. Auerbach Publications, 2013.

[FEH15a] Jr. Frank E. Harrell. *Regression Modeling Strategies With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis*, chapter Describing, Resampling, Validating, and Simplifying the Model, pages 103–126. Springer, Cham, 2015.

[FEH15b] Jr. Frank E. Harrell. *Regression Modeling Strategies With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis*, chapter Multivariable Modeling Strategies, pages 63–102. Springer, Cham, 2015.

[Fla12] Peter Flach. *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*. Cambridge University Press, USA, 2012.

[Fos15] Robert C. Foster. Beta-binomial empirical bayes. `http://www.probabilaball.com/2015/05/beta-binomial-empirical-bayes.html`, 2015.

[Fri99] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. 1999.

[FV06] Frédéric Ferraty and Philippe Vieu. *Nonparametric Functional Data Analysis*. Springer Series in Statistics. Springer-Verlag New York, 2006.

[GH06] Liqiang Geng and Howard J. Hamilton. Interestingness measures for data mining: A survey. *Computing Surveys*, 2006.

[GMCR04a] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Pereira Rodrigues. Learning with drift detection. *SBIA*, pages 286–295, 2004.

[GMCR04b] João Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. volume 8, pages 286–295, 09 2004.

[gpbed] Gap-Bide Python package. `https://github.com/socrateslee/pygapbide`, 2019 accessed.

[GPMT09] Irene Garrigós, Jesús Pardillo, Jose-Norberto Mazón, and Juan Trujillo. A conceptual modeling approach for olap personalization. volume 5829, pages 401–414, 11 2009.

[GR09] M. Golfarelli and S. Rizzi. Expressing olap preferences. *SSDBM*, 3:83–91, 2009.

[GZB+13] Joao Gama, Indre Zliobaite, ALbert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaption. *Computing Surveys*, 2013.

[Han15] Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Haufmann Publishers, 2 edition, 2015.

[Her99] Michael C. Herron. Postestimation uncertainty in limited dependent variable models. *Political Analysis*, 8, 1999.

[HS97] Sepp Hochreiter and Jorgen Schmidhuber. Long short-term memory. 1997.

[HTF01] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., 2001.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer-Verlag New York, 2009.

[Hun07] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[Jud16] Thomas W. Judson. *Abstract Algebra: theory and applications*, chapter Lattices and Boolean Algebras, pages 299–317. Orthogonal Publishing L3C, Ann Arbor, Michigan, 2016.

[JWHT13] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R*. Springer, New York, NY, 2013.

[KB12] Rym Khemiri and Fadila Bentayeb. User profile-driven data warehouse summary for adaptive olap queries. *International Journal of Database Management Systems*, 4, December 2012.

[kered] Keras: high-level neural networks API python package. `https://github.com/keras-team/keras`, 2019 accessed.

[KR13] Ralph Kimball and Margy Ross. *The data warehouse toolkit: The definitive guide to dimensional modeling*. John Wiley & Sons, 2013.

[LBE15] Zachary C. Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. June 2015.

[LW08] Chun Li and Jianyong Wang. Efficiently mining closed subsequences with gap constraints. volume 1, pages 313–322, 04 2008.

[Mau12] Vihinen Mauno. How to evaluate performance of prediction methods? measures and their interpretation in variation effect analysis. *BMC Genomics*, 13, 2012.

[MB12] Imene Mami and Zohra Bellahsene. A survey of view selection methods. *Acm Sigmod Record*, 41(1):20–29, 2012.

[MBBF99] Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent. 1999.

[McK10] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[MH19] John Miller and Moritz Hardt. Stable recurrent models. 2019.

[MQJH15] Peter Müller, Fernando Andrés Quintana, Alejandro Jara, and Tim Hanson. *Bayesian Nonparametric Data Analysis*. Springer Series in Statistics. Springer International Publishing, 2015.

[MRBA18] Jacob Montiel, Jesse Read, Albert Bifet, and Talel Abdessalem. Scikit-multiflow: A multi-output streaming framework. *Journal of Machine Learning Research*, 19(72):1–5, 2018.

[NM17] Shu Kay Ng and Geoffrey J. McLachlan. On the identification of correlated differential features for supervised classification of high-dimensional data. *Data Science, Studies in Classification, Data Analysis, and Knowledge Organization*, pages 43–57, 2017.

[Ola15] Christopher Olah. Understanding lstm networks. Technical report, 2015.

[Oli] Travis Oliphant. NumPy: A guide to NumPy. USA: Trelgol Publishing, 2006–. [Online; accessed May 2019].

[Pag54] E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1):100 – 115, 1954.

[Pie13] Granjon Pierre. The cusum algorithm - a small review. Technical report, June 2013.

[PMV17] Francesco Palumbo, Angela Montanari, and Maurizio Vichi, editors. *Data Science: Innovative Developments in Data Analysis and Clustering*, Studies in Classification, Data Analysis, and Knowledge Organization. Springer International Publishing, 2017.

[Pow15] David M.W. Powers. What the f-measure does not measure: Features, flaws, fallacies, and fixes. Technical Report KIT-14-001 Computer Science, Engineering and Mathematics, Flinders University, 2015.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[Ram02] James B. Ramsey. *Wavelets in Economics and Finance: Past and Future*, volume 6 of *International Conference on Machine Learning and Applications*. The Berkeley Electronic Press, 2002.

[RJ09] Howard L. Reskinoff and Raymond O. Wells Jr. *Wavelet Analysis The Scalable Structure of Information*. Springer Series in Statistics. Springer-Verlag New York, 2009.

[RS05a] James Ramsay and B. W. Silverman. *Functional Data Analysis*, chapter Introduction. Springer Series in Statistics. Springer-Verlag New York, second edition, 2005.

[RS05b] James Ramsay and B. W. Silverman. *Functional Data Analysis*, chapter From functional data to smooth functions. Springer Series in Statistics. Springer-Verlag New York, second edition, 2005.

[RT08] F. Ravat and O. Teste. New trends in data warehousing and data analysis. *Personalization and OLAP Databases*, 3:71–92, 2008. 1)Decision Support Systems for Logistics and Supply Chain Management 2)Business Intelligence and the Web.

[Sch02] Christoph Schleicher. An introduction to wavelets for economists. 2002.

[She09a] Simon J. Sheather. *A Modern Approach to Regression with R*, chapter Simple Linear Regression, pages 15–43. Springer New York, New York, NY, 2009.

[She09b] Simon J. Sheather. *A Modern Approach to Regression with R*, chapter Logistic Regression, pages 263–303. Springer New York, New York, NY, 2009.

[She09c] Simon J. Sheather. *A Modern Approach to Regression with R*, chapter Variable Selection, pages 227–262. Springer New York, New York, NY, 2009.

[She09d] Simon J. Sheather. *A Modern Approach to Regression with R*, chapter Mixed Models, pages 331–369. Springer New York, New York, NY, 2009.

[SM09] Ronald W. Shonkwiler and Franklin Mendivil. *Explorations in Monte Carlo Methods*. Undergraduate Texts in Mathematics. Springer-Verlag New York, 2009.

[SPH19] Tomas Sabata, Juraj Eduard Pall, and Martin Holena. Deep bayesian semi-supervised active learning for sequence labelling. 2019.

[SY17] Mukund Sundararajan and Qiqi Yan. A simple and efficient mapreduce algorithm for data cube materialization. *arXiv preprint arXiv:1709.10072*, 2017.

[Wal13] R.E. Walpole. *Essentials of Probability & Statistics for Engineers & Scientists*. Pearson, 2013.

[Wel98] Howard L. ResnikoffRaymond O. WellsJr. *Wavelet Analysis The Scalable Structure of Information*. Springer, New York, NY, 1998.

[WH04] J. Wang and J. Han. Bide: efficient mining of frequent closed sequences. In *Proceedings. 20th International Conference on Data Engineering*, pages 79–90, 2004.

Appendix A

Queries

A.1 Query Profiles

A number of queries from the first user profile follows below; where the dimensions denoted by letters refer to those in the four data marts depicted in figures 4.4, 4.3, 4.1 and 4.2.

Listing A.1: User profile 1 query requiring a DLU cuboid

```
1  SELECT U.GENDER, D.YEAR,D.MONTH, D.WEEK_DAY, AVG(NIGHTS),AVG(TOTAL_USD_AMOUNT)
2  FROM RESERVATIONS AS F
3  INNER JOIN DATE AS D ON D.DATE_ID=F.DATE_ID
4  INNER JOIN DWELLING AS temp ON temp.DWELLING_ID=F.DWELLING_ID
5  INNER JOIN LOCATION AS L ON L.LOCATION_ID=temp.LOCATION_ID
6  INNER JOIN USER AS U ON U.USER_ID=F.USER_ID
7  WHERE D.MONTH IN ('SEPTEMBER','OCTOBER','NOVEMBER') AND L.COUNTRY_CODE IN
       ('GB','NL','CA','US','DE') and D.YEAR >=2015
8  GROUP BY
9     GENDER,
10    CUBE (D.YEAR,D.MONTH, D.WEEK_DAY)
11 ORDER BY
12    D.YEAR, D.MONTH, D.WEEK_DAY
```

In the above query, a user seeks to compute the he average number of nights and total dollar amount spent in the Fall season by users in Great Britain, Netherlands, Canada, U.S.A, and Germany; per years after 2014 per each Fall month, and per day of the week to facilitate analysis of dwelling demand in the fall against expectation. Of interest to the user is: would there be weekly patterns that differ from the usual behaviour of frequent weekend trips in the Summer? The result of the query above enables analytical process to be carried out to answer questions like the prior.

Listing A.2: User profile 1 query benefiting from a DL cuboid

```
1  SELECT D.YEAR,D.QUARTER, D.MONTH, D.WEEK_DAY,L.COUNTRY,
       L.CITY_TOWNSHIP,SUM(NIGHTS), SUM(TOTAL_USD_AMOUNT)
2  FROM RESERVATIONS AS F
3  INNER JOIN DATE AS D ON D.DATE_ID=F.DATE_ID
4  INNER JOIN DWELLING AS temp ON temp.DWELLING_ID=F.DWELLING_ID
5  INNER JOIN LOCATION AS L ON L.LOCATION_ID=temp.LOCATION_ID
6  WHERE D.QUARTER IN (2,3) AND L.COUNTRY_CODE IN ('GB','NL','ES','AUS','DE') and
       D.YEAR IN (2015,2016)
7  GROUP BY
8     L.COUNTRY, L.CITY_TOWNSHIP,
9     ROLLUP (D.YEAR,D.QUARTER, D.MONTH, D.WEEK_DAY)
10    -- order of parameters yields a drill down result
11 ORDER BY
12    D.YEAR, D.QUARTER, L.COUNTRY, L.CITY_TOWNSHIP
```

Listing A.3: User profile 1 query benefiting from a L cuboid

```
1  SELECT L.COUNTRY, L.STATE_PROVINCE, L.CITY_TOWNSHIP, SUM(NIGHTS),
       SUM(TOTAL_USD_AMOUNT)
2  FROM RESERVATIONS AS F
3  INNER JOIN DWELLING_BRIDGE AS B ON B.DWELLING_ID=F.DWELLING_ID
4  INNER JOIN LOCATION AS L ON L.LOCATION_ID=B.LOCATION_ID
5  GROUP BY
6     ROLLUP ( L.CITY_TOWNSHIP, L.STATE_PROVINCE, L.COUNTRY)
7     -- order of parameters yields a roll up result
8  ORDER BY
9     L.COUNTRY
```

Listing A.4: User profile 1 query benefiting from a UW cuboid

```
1  SELECT U.GENDER, W.TYPE, W.NUMBER_BEDROOMS, DATEDIFF(YEAR, U.BIRTHDATE,
       CURRENT_DATE) AS AGE , COUNT(F.USER_ID), AVG(NIGHTS),
       AVG(TOTAL_USD_AMOUNT)
2  FROM RESERVATIONS AS F
3  INNER JOIN DWELLING AS W ON W.DWELLING_ID=F.DWELLING_ID
4  INNER JOIN USER AS U ON U.USER_ID=F.USER_ID
5  WHERE
6  GROUP BY
7     GENDER, DATEDIFF(YEAR, U.BIRTHDATE, CURRENT_DATE),
8     CUBE (W.TYPE, W.NUMBER_BEDROOMS)
9  ORDER BY AGE DESC
10 -- Historical dwelling preferences across demographics.
```

The second user group profile:

Listing A.5: User profile 2 query benefiting from a W cuboid

```
1  SELECT W.TYPE, W.NUMBER_BEDROOMS, W.CAPACITY, COUNT(W.DWELLING_ID),
       SUM(NUMBER_OF_PEOPLE) SUM(NIGHTS), SUM(TOTAL_USD_AMOUNT)
2  FROM RESERVATIONS AS F
3  INNER JOIN DWELLING AS W ON W.DWELLING_ID=F.DWELLING_ID
4  GROUP BY
5     ROLLUP (W.TYPE, W.NUMBER_BEDROOMS, W.CAPACITY)
6  ORDER BY W.TYPE
```

Listing A.6: User profile 2 query benefiting from a P cuboid

```
1  SELECT P.TYPE,AVG(DURATION_IN_MINUTES), MAX(DURATION_IN_MINUTES),
2  FROM SESSIONS AS F
3  INNER JOIN ACTION AS P ON P.ACTION_ID=F.ACTION_ID
4  WHERE CLIENT_FACING='Y' AND P.CLASS IS NOT IN ('click','submit')
5  GROUP BY
6     P.TYPE
7  ORDER BY P.TYPE
8  LIMIT 10
```

Listing A.7: User profile 2 query benefiting from a SU cuboid

```
1  SELECT GENDER, DATEDIFF(YEAR, U.BIRTHDATE, CURRENT_DATE) AS AGE, S.PROVIDER,
       COUNT(USER_ID)
2  FROM SIGNUP AS F
3  INNER JOIN AFFILIATE AS S ON S.AFFILIATE_ID=F.AFFILIATE_ID
4  INNER JOIN USER AS U ON U.USER_ID=F.USER_ID
5  WHERE S.CHANNEL != 'direct'
6  GROUP BY
7     GENDER,
8     CUBE(AGE, PROVIDER)
9  /*To measure the way users are directed to signup on the app, in order to make
       reservations. A provider takes values: google, other, craigslist,
       instagram, and some others. Is the target audience for the app's marketing
       on different providers consistent with the marketing hypothesis of which
       platform to target certain demographics?*/
```

Listing A.8: User profile 2 query benefiting from a UW cuboid

```
1  SELECT CLIENT_FACING,P.TYPE,T.PART_OF_DAY, AM_PM, AVG(DURATION_IN_MINUTES),
       MAX(DURATION_IN_MINUTES),
2  FROM SESSIONS AS F
3  INNER JOIN ACTION AS P ON P.ACTION_ID=F.ACTION_ID
4  INNER JOIN USER AS U ON U.USER_ID=F.USER_ID
5  INNER JOIN TIME_OF_DAY AS T ON T.TIME_ID=F.TIME_ID
6  WHERE P.CLASS IN ('unavailabilities','ajax_country_options',
       'similar_listings')
7  GROUP BY
8     CLIENT_FACING,
9     ROLLUP(AM_PM,T.PART_OF_DAY,P.TYPE)
10 /*Here the duration between app users and company analytics' users at
       different parts of the day (morning, afternoon, evening, night) is
       aggregated as an average and the upper bound. This prepares a data set to
       contrast traffic of clients and company analysts' querying similar data
       marts.*/
```

Table A.1: Legend for referring to dimensions or views.

Letter Symbol	**Data Referenced**
U	User Dimension
T	Time Dimension
L	Location Dimension
D	Date Dimension
W	Dwelling Dimension
H	Promotion Dimension
P	Action Dimension
S	Affiliate Dimension
M	End Date role playing view of date dimension for reservations.
R	End Time role playing view of date dimension for reservations.
C	Visitor Contacts Bridge View of non-registered users.

Table A.2: Query distribution and cuboid association per profile.

Profile	**Query ID**	**Cuboid**	**Mean**	**Variance**	**Probability**
1^{st}	38	DLU	143	56	0.03
1^{st}	20	DLU	5	4	0.07
1^{st}	37	L	5	3	0.08
1^{st}	2	UW	16	3	0.09
1^{st}	9	DL	13	5	0.1
1^{st}	50	DL	4	2	0.13
1^{st}	12	UW	15	7	0.2
1^{st}	32	L	8	4	0.3
2^{nd}	1	PTU	28	33	0.01
2^{nd}	14	SU	29	14	0.02
2^{nd}	39	PTU	46	16	0.03
2^{nd}	3	SU	5	4	0.04
2^{nd}	34	W	7	3	0.04

Continued on next page

Table A.2 – continued from previous page

Profile	Query ID	Cuboid	Mean	Variance	Probability
2^{nd}	23	W	6	3	0.05
2^{nd}	43	PTU	14	3	0.11
2^{nd}	17	P	7	2	0.13
2^{nd}	11	SU	8	9	0.19
2^{nd}	19	P	8	4	0.19
2^{nd}	4	W	8	5	0.19
3^{rd}	33	M	6	3	0.01
3^{rd}	1	PTU	28	8	0.01
3^{rd}	5	LT	12	4	0.01
3^{rd}	34	W	7	2	0.02
3^{rd}	37	L	5	3	0.02
3^{rd}	9	DL	13	7	0.03
3^{rd}	36	LT	3	5	0.05
3^{rd}	50	DL	4	4	0.07
3^{rd}	11	SU	8	4	0.13
3^{rd}	10	M	8	5	0.13
3^{rd}	12	UW	15	12	0.2
3^{rd}	45	DHL	8	13	0.32
4^{th}	26	MU	6	2	0.01
4^{th}	41	L	8	2	0.01
4^{th}	14	SU	29	14	0.01
4^{th}	23	W	6	3	0.01
4^{th}	37	L	5	3	0.01
4^{th}	21	U	1	2	0.02
4^{th}	18	CDL	5	3	0.03
4^{th}	17	P	7	2	0.04
4^{th}	13	U	7	3	0.05
4^{th}	30	LR	23	13	0.06
4^{th}	6	CDLU	38	16	0.07
4^{th}	46	CDLU	2	1	0.08
4^{th}	42	LT	5	5	0.14

Continued on next page

Table A.2 – continued from previous page

Profile	Query ID	Cuboid	Mean	Variance	Probability
4^{th}	15	LT	10	5	0.21
4^{th}	28	TU	8	9	0.25

Appendix B

Analysis Data

B.1 Stacked Ensemble: Meta-Classifier Configurations

Here we show hyper-parameters and settings used during experimentation.

Listing B.1: configuration 1 of GBR

```
import pandas as pd
from sklearn.ensemble import GradientBoostingRegressor
from verstack.stratified_continuous_split import scsplit
#test size is 30% of data, using the full feature data set; next example shows a reduced set eliminating redundant variables improved the R^2 score (regression fit score).
X_train, X_test, y_train, y_test = scsplit(X, y, stratify = y, test_size=0.3)
reg = GradientBoostingRegressor(learning_rate=0.13, max_depth=8, loss='lad', n_estimators=250, criterion='friedman_mse')
#r^2 score: 0.6405120302906603
```

Listing B.2: configuration 2 of GBR

```
#Using a reduced set of features informed by a similar graphic for the first configuration, this new set appears in Figure \ref{fig: feature
```

```
    importance}.
import pandas as pd
from sklearn.ensemble import GradientBoostingRegressor
from verstack.stratified_continuous_split import scsplit
X_train, X_test, y_train, y_test = scsplit(X, y, stratify = y, test_size=0.3)
reg = GradientBoostingRegressor(learning_rate=0.1345, max_depth=8, loss='lad',
    n_estimators=300, criterion='friedman_mse')
#r^2 score: 0.7036641906025125
#using a 20% test set size, r^2 score: 0.7260587896054838
```

B.2 Measurements for Analysis

Remaining pages are the five data sheets which served as inputs into the analysis detailed in Chapter 5 and shown in the critical distance graphs for comparing significant differences among the approaches on the three measures for performance. The first iteration for set 1 began on day 4 of the time series, the first four days of data were for the initial training set. For set 2 the first iteration marks day 5 of the time series of total 39 days. Similarly in set 3, of the 99 day total time series the first iteration begins on day 4 leaving the 95 days of data to make up the test set.

Comparison of the KPI across all approaches on data set 1

Iteration	BBB w/ noCD	BBB w/ naive	BBB w/ ddm	BBB w/ eddm	GB w/ noCD	GB w/ naive	GB w/ ddm	GB w/ eddm	LSTM w/ noCD	LSTM w/ naive	LSTM w/ ddm	LSTM w/ eddm
1	0.85	0.85	0.85	0.85	0.86	0.9	0.86	0.86	0.92	0.92	0.92	0.92
2	0.97	0.97	0.97	0.97	0.9	0.97	0.9	0.9	0.8	0.8	0.8	0.8
3	0.91	0.91	0.91	0.91	0.94	1	0.94	0.94	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	0.97	1	1	0.87	0.87	0.87	0.87
6	0.96	0.96	0.96	0.96	0.88	0.89	0.88	0.88	0.62	0.62	0.62	0.62
7	0.72	0.72	0.72	0.72	0.79	0.9	0.79	0.79	0.82	0.82	0.82	0.82
8	0.71	0.81	0.81	0.71	0.8	0.88	0.8	0.8	0.81	0.92	0.92	0.88
9	0.68	0.78	0.78	0.68	0.78	0.88	0.78	0.78	1	0.96	1	1
10	0.77	0.77	0.77	0.77	0.77	0.88	0.77	0.77	1	0.96	1	1
11	0.77	0.77	0.77	0.77	0.77	0.87	0.77	0.77	1	1	1	1
12	0.75	1	1	0.75	0.75	0.86	0.75	0.75	0.96	0.9	0.96	0.96
13	0.74	0.96	0.96	0.74	0.74	0.85	0.74	0.74	0.92	0.87	0.92	0.91
14	0.72	0.91	0.91	0.72	0.71	0.9	0.91	0.71	0.9	0.89	0.91	0.91
15	0.68	0.89	0.89	0.68	0.88	1	0.88	0.88	0.87	0.87	0.87	1
16	0.75	1	1	0.75	0.87	0.8	1	0.87	0.8	0.8	0.81	0.81
17	0.75	0.8	0.8	0.75	0.76	0.74	0.8	0.76	0.74	0.74	0.74	0.74
18	0.75	0.74	0.74	0.74	0.74	1	0.74	0.74	1	1	1	1
19	1	1	1	1	1	0.2	1	1	0.98	0.22	0	0.98
20	0.98	0.2	0.2	0.2	0.1	0.99	0.1	0.1	0.99	0.99	0.01	0.99
21	0.99	0.99	0.99	0	0	0.98	0.99	0	0.9	0.69	0.79	0.9
22	0.9	0.92	0.92	0.44	0.21	1	0.98	0.21	0.98	1	1	0.28
23	0.98	0.99	0.99	0.52	0.98	0.99	1	0.98	0.67	0.99	0.99	0.99
24	0.99	0.99	0.99	0.65	0.99	1	0.99	0.99	1	1	1	1
25	1	1	1	1	1	1	1	1	1	1	1	1
26	1	1	1	1	1	1	1	1	1	1	1	1

Comparison of the KPI across all approaches on data set 2

Iteration	BBB w/ noCD	BBB w/ naive	BBB w/ ddm	BBB w/ eddm	GB w/ noCD	GB w/ naive	GB w/ ddm	GB w/ eddm	LSTM w/ noCD	LSTM w/ naive	LSTM w/ ddm	LSTM w/ eddm
1	0.92	0.87	0.92	0.92	0.77	0.77	0.77	0.77	0.89	0.89	0.89	0.89
2	0.85	0.81	0.85	0.85	0.83	0.83	0.83	0.83	0.78	0.78	0.78	0.83
3	0.88	0.85	0.88	0.88	0.8	0.8	0.8	0.8	0.72	0.72	0.72	0.72
4	0.77	0.71	0.77	0.77	0.71	0.71	0.71	0.71	0.6	0.6	0.16	0.6
5	0.63	0.56	0.63	0.63	0.69	0.63	0.63	0.69	0.56	0.67	0.86	0.56
6	0.81	0.78	0.72	0.81	0.83	0.76	0.77	0.83	0.85	0.81	0.83	0.85
7	0.78	0.74	0.82	0.78	0.85	0.74	0.79	0.85	0.71	0.75	0.76	0.71
8	0.65	0.6	0.71	0.65	0.78	0.74	0.76	0.78	0.76	0.65	0.72	0.76
9	0.67	0.65	0.81	0.67	0.78	0.7	0.69	0.73	0.76	0.77	0.71	0.84
10	0.6	0.72	0.86	0.6	0.72	0.68	0.68	0.72	0.77	0.79	0.75	0.52
11	0.53	0.81	0.82	0.53	0.66	0.64	0.66	0.66	0.52	0.51	0.62	0.59
12	0.66	0.67	0.68	0.66	0.76	0.73	0.66	0.76	0.66	0.73	0.55	0.67
13	0.72	0.75	0.79	0.72	0.77	0.66	0.66	0.77	0.68	0.76	0.73	0.65
14	0.69	0.82	0.76	0.69	0.73	0.69	0.71	0.75	0.74	0.75	0.73	0.75
15	0.75	0.78	0.73	0.75	0.69	0.59	0.73	0.65	0.76	0.71	0.74	0.65
16	0.73	0.79	0.63	0.73	0.73	0.78	0.78	0.69	0.84	0.79	0.53	0.81
17	0.62	0.71	0.78	0.62	0.75	0.72	0.72	0.75	0.82	0.75	0.52	0.74
18	0.77	0.78	0.82	0.77	0.79	0.7	0.7	0.79	0.73	0.82	0.53	0.73
19	0.81	0.71	0.8	0.81	0.81	0.75	0.76	0.76	0.79	0.8	0.31	0.74
20	0.85	0.75	0.77	0.85	0.78	0.77	0.78	0.71	0.67	0.59	0.27	0.48
21	0.67	0.68	0.87	0.61	0.8	0.8	0.8	0.72	0.85	0.88	0.88	0.86
22	0.77	0.73	0.83	0.84	0.85	0.85	0.64	0.73	0.89	0.89	0.86	0.8
23	0.77	0.81	0.88	0.9	0.78	0.78	0.69	0.75	0.67	0.74	0.74	0.74
24	0.69	0.67	0.73	0.73	0.68	0.65	0.65	0.68	0.65	0.65	0.57	0.77
25	0.74	0.58	0.64	0.64	0.71	0.67	0.67	0.71	0.7	0.81	0.69	0.74
26	0.67	0.6	0.69	0.73	0.74	0.69	0.69	0.74	0.73	0.84	0.6	0.83
27	0.79	0.74	0.65	0.79	0.85	0.77	0.77	0.85	0.82	0.84	0.26	0.84
28	0.83	0.64	0.8	0.79	0.83	0.72	0.72	0.83	0.78	0.77	0.77	0.77
29	0.69	0.53	0.71	0.74	0.78	0.64	0.78	0.64	0.76	0.77	0.81	0.8

Iteration	BBB w/ noCD	BBB w/ naive	BBB w/ ddm	BBB w/ eddm	GB w/ noCD	GB w/ naive	GB w/ ddm	GB w/ eddm	LSTM w/ noCD	LSTM w/ naive	LSTM w/ ddm	LSTM w/ eddm
30	0.77	0.73	0.77	0.79	0.79	0.63	0.63	0.63	0.8	0.81	0.81	0.93
31	0.91	0.81	0.8	0.8	0.82	0.6	0.7	0.82	0.79	0.81	0.81	0.91
32	0.91	0.77	0.79	0.79	0.8	0.67	0.8	0.8	0.83	0.81	0.81	0.8
33	0.82	0.79	0.79	0.77	0.73	0.71	0.75	0.67	0.83	0.76	0.77	0.78
34	0.73	0.73	0.78	0.83	0.73	0.73	0.7	0.69	0.83	0.83	0.83	0.83

Comparison of the KPI across all approaches on data set 3

Iteration	BBB w/ noCD	BBB w/ naive	BBB w/ ddm	BBB w/ eddm	GB w/ noCD	GB w/ naive	GB w/ ddm	GB w/ eddm	LSTM w/ noCD	LSTM w/ naive	LSTM w/ ddm	LSTM w/ eddm
1	0.92	0.92	0.92	0.92	0.77	0.77	0.77	0.77	0.89	0.89	0.89	0.89
2	0.85	0.85	0.85	0.85	0.83	0.83	0.83	0.83	0.78	0.78	0.78	0.78
3	0.88	0.88	0.88	0.88	0.8	0.8	0.8	0.8	0.72	0.72	0.72	0.72
4	0.77	0.77	0.77	0.77	0.71	0.71	0.71	0.71	0.6	0.6	0.6	0.6
5	0.63	0.63	0.63	0.63	0.69	0.63	0.63	0.69	0.56	0.67	0.67	0.56
6	0.81	0.72	0.72	0.81	0.83	0.76	0.77	0.83	0.85	0.81	0.81	0.85
7	0.78	0.86	0.82	0.78	0.85	0.74	0.79	0.85	0.69	0.75	0.75	0.71
8	0.65	0.73	0.71	0.65	0.78	0.74	0.76	0.78	0.76	0.65	0.65	0.76
9	0.67	0.8	0.81	0.67	0.78	0.7	0.69	0.73	0.87	0.79	0.49	0.87
10	0.6	0.79	0.86	0.6	0.72	0.68	0.68	0.72	0.82	0.78	0.74	0.82
11	0.53	0.82	0.82	0.53	0.66	0.64	0.66	0.66	0.63	0.58	0.55	0.67
12	0.66	0.68	0.68	0.66	0.76	0.73	0.66	0.76	0.81	0.82	0.66	0.8
13	0.72	0.8	0.79	0.72	0.77	0.66	0.66	0.77	0.81	0.81	0.3	0.76
14	0.69	0.81	0.76	0.69	0.73	0.69	0.71	0.75	0.77	0.76	0.65	0.76
15	0.75	0.77	0.73	0.75	0.69	0.59	0.73	0.65	0.81	0.77	0.65	0.74
16	0.73	0.81	0.63	0.73	0.73	0.78	0.78	0.69	0.81	0.81	0.61	0.69
17	0.62	0.85	0.78	0.62	0.75	0.72	0.72	0.75	0.63	0.81	0.42	0.76
18	0.77	0.81	0.82	0.77	0.79	0.7	0.7	0.79	0.81	0.74	0.25	0.74
19	0.81	0.74	0.8	0.81	0.81	0.75	0.76	0.76	0.79	0.85	0.55	0.79
20	0.85	0.81	0.77	0.85	0.78	0.77	0.78	0.71	0.76	0.76	0.7	0.52
21	0.67	0.83	0.87	0.61	0.8	0.8	0.8	0.72	0.84	0.86	0.52	0.86
22	0.77	0.85	0.83	0.84	0.85	0.85	0.64	0.73	0.89	0.89	0.8	0.9
23	0.77	0.9	0.88	0.9	0.78	0.78	0.69	0.75	0.74	0.74	0.74	0.77
24	0.69	0.74	0.73	0.73	0.68	0.65	0.65	0.68	0.77	0.71	0.63	0.67
25	0.74	0.7	0.64	0.64	0.71	0.67	0.67	0.71	0.67	0.7	0.53	0.7
26	0.67	0.74	0.69	0.73	0.74	0.69	0.69	0.74	0.79	0.75	0.79	0.73
27	0.79	0.79	0.65	0.79	0.85	0.77	0.77	0.85	0.84	0.85	0.84	0.82
28	0.83	0.75	0.8	0.79	0.83	0.72	0.72	0.83	0.69	0.77	0.77	0.77
29	0.69	0.67	0.71	0.74	0.78	0.64	0.78	0.64	0.79	0.8	0.77	0.76

Iteration	BBB w/ noCD	BBB w/ naive	BBB w/ ddm	BBB w/ eddm	GB w/ noCD	GB w/ naive	GB w/ ddm	GB w/ eddm	LSTM w/ noCD	LSTM w/ naive	LSTM w/ ddm	LSTM w/ eddm
30	0.77	0.76	0.77	0.79	0.79	0.63	0.63	0.63	0.61	0.83	0.8	0.8
31	0.91	0.81	0.8	0.8	0.82	0.6	0.7	0.82	0.81	0.81	0.81	0.81
32	0.91	0.8	0.79	0.79	0.8	0.67	0.8	0.8	0.77	0.76	0.8	0.77
33	0.82	0.83	0.79	0.77	0.73	0.71	0.75	0.67	0.79	0.83	0.79	0.79
34	0.73	0.83	0.78	0.83	0.73	0.73	0.7	0.69	0.82	0.83	0.82	0.82
35	0.82	0.83	0.82	0.83	0.73	0.75	0.68	0.68	0.85	0.84	0.85	0.85
36	0.87	0.84	0.84	0.82	0.82	0.82	0.69	0.69	0.92	0.91	0.92	0.92
37	0.85	0.92	0.91	0.91	0.89	0.88	0.88	0.74	0.9	0.66	0.9	0.89
38	0.92	0.9	0.9	0.89	0.86	0.7	0.7	0.86	0.89	0.9	0.9	0.89
39	0.85	0.89	0.89	0.88	0.85	0.84	0.84	0.85	0.91	0.91	0.91	0.91
40	0.8	0.91	0.91	0.9	0.83	0.83	0.83	0.83	0.94	0.96	0.96	0.96
41	0.74	0.95	0.95	0.95	0.85	0.84	0.84	0.84	0.85	0.88	0.87	0.86
42	0.62	0.85	0.85	0.87	0.76	0.75	0.75	0.76	0.69	0.76	0.7	0.69
43	0.67	0.69	0.69	0.71	0.74	0.74	0.74	0.74	0.67	0.79	0.77	0.72
44	0.62	0.87	0.86	0.81	0.81	0.8	0.8	0.81	0.62	0.71	0.24	0.68
45	0.56	0.89	0.9	0.85	0.78	0.76	0.76	0.74	0.85	0.86	0.12	0.81
46	0.79	0.85	0.86	0.79	0.82	0.8	0.8	0.78	0.84	0.88	0.9	0.79
47	0.76	0.87	0.88	0.82	0.85	0.83	0.83	0.83	0.87	0.86	0.86	0.8
48	0.75	0.83	0.86	0.79	0.83	0.77	0.77	0.77	0.77	0.8	0.8	0.59
49	0.71	0.79	0.81	0.72	0.77	0.67	0.67	0.67	0.72	0.73	0.72	0.68
50	0.69	0.74	0.78	0.68	0.73	0.6	0.6	0.61	0.74	0.75	0.18	0.59
51	0.63	0.74	0.82	0.74	0.76	0.7	0.7	0.63	0.84	0.79	0.63	0.7
52	0.67	0.82	0.82	0.75	0.79	0.62	0.62	0.62	0.92	0.76	0.69	0.61
53	0.79	0.92	0.85	0.81	0.84	0.75	0.75	0.64	0.93	0.74	0.88	0.86
54	0.8	0.92	0.86	0.81	0.85	0.75	0.75	0.64	0.97	0.91	0.95	0.91
55	0.85	0.97	0.9	0.87	0.9	0.79	0.8	0.68	0.94	0.92	0.89	0.91
56	0.82	0.94	0.91	0.88	0.88	0.78	0.78	0.67	0.93	0.94	0.88	0.93
57	0.92	0.93	0.93	0.92	0.91	0.8	0.8	0.67	0.88	0.87	0.84	0.86
58	0.77	0.81	0.84	0.83	0.82	0.72	0.71	0.61	0.65	0.66	0.63	0.71
59	0.61	0.67	0.71	0.73	0.68	0.64	0.64	0.61	0.67	0.69	0.8	0.66
60	0.66	0.81	0.69	0.87	0.79	0.78	0.78	0.76	0.76	0.72	0.82	0.76
61	0.72	0.78	0.75	0.79	0.8	0.8	0.8	0.76	0.78	0.7	0.79	0.78
62	0.77	0.78	0.78	0.8	0.8	0.77	0.77	0.47	0.85	0.73	0.88	0.85

Iteration	BBB w/ noCD	BBB w/ naive	BBB w/ ddm	BBB w/ eddm	GB w/ noCD	GB w/ naive	GB w/ ddm	GB w/ eddm	LSTM w/ noCD	LSTM w/ naive	LSTM w/ ddm	LSTM w/ eddm
63	0.79	0.87	0.85	0.83	0.86	0.81	0.82	0.45	0.85	0.83	0.92	0.89
64	0.7	0.9	0.9	0.87	0.79	0.68	0.65	0.17	0.83	0.68	0.87	0.83
65	0.56	0.84	0.84	0.82	0.67	0.66	0.68	0.22	0.69	0.7	0.69	0.7
66	0.72	0.71	0.71	0.74	0.78	0.7	0.7	0.36	0.8	0.75	0.79	0.68
67	0.74	0.81	0.81	0.78	0.79	0.73	0.69	0.39	0.84	0.73	0.84	0.86
68	0.71	0.85	0.85	0.85	0.74	0.63	0.66	0.38	0.76	0.76	0.76	0.76
69	0.73	0.77	0.77	0.76	0.7	0.58	0.67	0.27	0.76	0.68	0.69	0.81
70	0.76	0.81	0.81	0.77	0.74	0.79	0.76	0.24	0.64	0.69	0.63	0.89
71	0.64	0.87	0.87	0.8	0.76	0.82	0.73	0.19	0.85	0.82	0.31	0.79
72	0.79	0.82	0.82	0.84	0.8	0.81	0.62	0.59	0.46	0.78	0.63	0.7
73	0.74	0.72	0.72	0.79	0.75	0.73	0.73	0.65	0.77	0.84	0.7	0.82
74	0.84	0.79	0.79	0.83	0.74	0.72	0.75	0.67	0.86	0.85	0.66	0.89
75	0.85	0.84	0.84	0.84	0.75	0.8	0.8	0.77	0.82	0.8	0.77	0.86
76	0.8	0.86	0.86	0.82	0.69	0.71	0.71	0.67	0.84	0.84	0.72	0.78
77	0.81	0.86	0.86	0.83	0.72	0.73	0.73	0.68	0.83	0.85	0.69	0.81
78	0.81	0.86	0.86	0.82	0.71	0.78	0.69	0.68	0.81	0.82	0.46	0.74
79	0.8	0.85	0.85	0.83	0.71	0.7	0.69	0.55	0.74	0.67	0.52	0.67
80	0.73	0.73	0.73	0.74	0.65	0.64	0.64	0.54	0.79	0.67	0.61	0.72
81	0.75	0.79	0.79	0.75	0.71	0.72	0.71	0.64	0.8	0.76	0.29	0.77
82	0.81	0.85	0.85	0.83	0.76	0.72	0.74	0.67	0.79	0.76	0.83	0.77
83	0.8	0.83	0.83	0.81	0.76	0.74	0.74	0.7	0.86	0.81	0.84	0.83
84	0.82	0.85	0.85	0.82	0.75	0.75	0.77	0.7	0.84	0.78	0.84	0.78
85	0.79	0.83	0.83	0.8	0.7	0.7	0.72	0.64	0.83	0.76	0.83	0.75
86	0.78	0.83	0.83	0.73	0.67	0.68	0.68	0.64	0.69	0.7	0.67	0.81
87	0.72	0.79	0.79	0.73	0.67	0.69	0.69	0.57	0.66	0.7	0.66	0.77
88	0.66	0.73	0.73	0.7	0.67	0.65	0.65	0.57	0.6	0.6	0.66	0.73
89	0.56	0.73	0.73	0.66	0.65	0.63	0.63	0.54	0.61	0.75	0.37	0.74
90	0.56	0.74	0.74	0.69	0.67	0.63	0.6	0.51	0.6	0.72	0.73	0.76
91	0.6	0.76	0.76	0.7	0.68	0.67	0.65	0.59	0.79	0.67	0.77	0.9
92	0.77	0.9	0.9	0.77	0.7	0.7	0.7	0.62	0.82	0.9	0.9	0.89
93	0.77	0.89	0.89	0.77	0.73	0.72	0.79	0.67	0.88	0.89	0.89	0.87
94	0.73	0.86	0.86	0.83	0.77	0.74	0.74	0.7	0.89	0.89	0.89	0.89
95	0.74	0.88	0.88	0.87	0.78	0.77	0.77	0.74	0.93	0.95	0.95	0.95

Set 3 comparison of daily change detection flag

Iteration	noCD	BBB w/ naive	BBB w/ ddm	BBB w/ eddm	GB w/ naive	GB w/ ddm	GB w/ eddm	LSTM w/ naive	LSTM w/ ddm	LSTM w/ eddm
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	1	1	0
4	0	0	0	0	1	1	0	1	1	0
5	0	1	1	0	1	1	0	1	1	0
6	0	1	1	0	0	0	0	0	0	0
7	0	0	0	0	1	0	0	0	0	0
8	0	1	1	0	1	0	1	1	1	0
9	0	0	0	0	1	1	0	0	1	1
10	0	0	0	0	1	1	0	0	1	0
11	0	0	0	0	1	1	0	1	1	0
12	0	1	1	0	1	1	0	0	1	0
13	0	0	0	0	1	1	1	0	1	0
14	0	0	0	0	1	1	0	0	1	1
15	0	0	1	0	1	1	0	0	1	0
16	0	0	1	0	0	0	0	0	1	0
17	0	0	0	0	1	1	0	0	1	0
18	0	0	0	0	1	1	1	1	1	0
19	0	1	0	0	0	0	0	0	1	0
20	0	0	0	1	0	0	0	0	1	0
21	0	0	0	0	0	0	0	0	1	0
22	0	0	0	0	0	1	0	0	0	1
23	0	0	0	1	0	1	0	1	1	0
24	0	1	1	0	1	1	0	1	1	0
25	0	1	1	0	1	1	0	1	1	0
26	0	1	1	0	1	1	0	0	0	0
27	0	0	1	0	0	0	0	0	0	0
28	0	0	0	1	1	1	1	0	0	1
29	0	1	1	0	1	0	0	0	0	0
30	0	0	0	0	1	1	0	0	0	0
31	0	0	0	1	1	1	0	0	0	0

Iteration	noCD	BBB w/ naive	BBB w/ ddm	BBB w/ eddm	GB w/ naive	GB w/ ddm	GB w/ eddm	LSTM w/ naive	LSTM w/ ddm	LSTM w/ eddm
32	0	0	0	0	1	0	1	0	0	0
33	0	0	0	0	1	0	0	0	0	0
34	0	0	0	0	1	1	0	0	0	0
35	0	0	0	0	0	1	0	0	0	0
36	0	0	0	0	0	1	0	0	0	0
37	0	0	0	0	0	0	0	1	0	0
38	0	0	0	0	1	1	0	0	0	0
39	0	0	0	0	0	0	1	0	0	0
40	0	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	1	0
43	0	1	1	0	1	1	0	0	0	0
44	0	0	0	0	0	0	0	1	1	0
45	0	0	0	0	0	0	0	0	1	0
46	0	0	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0
48	0	0	0	1	0	0	1	0	0	0
49	0	0	0	0	1	1	0	1	1	0
50	0	1	0	0	1	1	0	0	1	0
51	0	1	0	0	1	1	0	0	1	0
52	0	0	0	0	1	1	0	0	1	0
53	0	0	0	1	0	0	0	1	0	0
54	0	0	0	0	0	0	0	0	0	0
55	0	0	0	0	0	0	0	0	0	1
56	0	0	0	0	0	0	0	0	0	0
57	0	0	0	0	0	0	0	0	0	0
58	0	0	0	0	1	1	0	1	1	0
59	0	1	1	0	1	1	0	1	0	0
60	0	0	1	0	0	0	0	1	0	0
61	0	0	0	0	0	0	1	1	0	1
62	0	0	0	1	0	0	0	1	0	0
63	0	0	0	0	0	0	0	0	0	0

Iteration	noCD	BBB w/ naive	BBB w/ ddm	BBB w/ eddm	GB w/ naive	GB w/ ddm	GB w/ eddm	LSTM w/ naive	LSTM w/ ddm	LSTM w/ eddm
64	0	0	0	0	1	1	0	1	0	0
65	0	0	0	0	1	1	0	1	1	0
66	0	1	1	0	1	1	0	0	0	0
67	0	0	0	0	1	1	0	1	0	0
68	0	0	0	0	1	1	0	0	0	0
69	0	0	0	1	1	1	0	1	1	1
70	0	0	0	0	0	0	0	1	1	0
71	0	0	0	0	0	1	0	0	1	0
72	0	0	0	0	0	1	0	0	1	0
73	0	1	1	0	1	1	0	0	1	0
74	0	0	0	0	1	0	0	0	1	0
75	0	0	0	0	0	0	0	0	0	1
76	0	0	0	0	1	1	0	0	1	0
77	0	0	0	0	1	1	0	0	1	0
78	0	0	0	0	0	1	0	0	1	0
79	0	0	0	0	1	1	0	1	1	0
80	0	1	1	0	1	1	0	1	1	0
81	0	0	0	0	1	1	0	0	1	0
82	0	0	0	0	1	1	0	0	0	1
83	0	0	0	0	1	1	0	0	0	0
84	0	0	0	0	0	0	0	0	0	0
85	0	0	0	1	1	1	0	0	0	0
86	0	0	0	0	1	1	0	1	1	0
87	0	0	0	0	1	1	0	1	1	0
88	0	1	1	0	1	1	0	1	1	0
89	0	1	1	0	1	1	0	0	1	0
90	0	1	1	0	1	1	0	1	1	0
91	0	0	0	0	1	1	0	1	0	0
92	0	0	0	0	1	1	0	0	0	1
93	0	0	0	1	1	0	0	0	0	0
94	0	0	0	0	1	1	0	0	0	0
95	0	0	0	0	0	0	0	0	0	0

Set 3 comparison of daily scale allocated memory

Iteration	noCD	BBB w/ naive	BBB w/ ddm	BBB w/ eddm	GB w/ naive	GB w/ ddm	GB w/ eddm	LSTM w/ naive	LSTM w/ ddm	LSTM w/ eddm
1	1140	1140	1140	1140	1140	1140	1140	1140	1140	1140
2	1140	1140	1140	1140	1140	1140	1140	1140	1140	1140
3	1140	1140	1140	1140	1140	1140	1140	1140	1140	1140
4	1140	1140	1140	1140	1140	1140	1140	1174	1174	1140
5	1140	1140	1140	1140	1186	1186	1140	1350	1350	1140
6	1140	1277	1277	1140	1328	1328	1140	1458	1458	1140
7	1140	1315	1315	1140	1328	1328	1140	1458	1458	1140
8	1140	1315	1315	1140	1341	1328	1140	1458	1458	1140
9	1140	1341	1368	1140	1354	1328	1174	1140	1140	1140
10	1140	1140	1140	1140	1140	1140	1140	1140	1436	1277
11	1140	1140	1140	1140	1220	1220	1140	1140	1450	1277
12	1140	1140	1140	1140	1354	1330	1140	1334	1710	1277
13	1140	1220	1220	1140	1381	1450	1140	1334	1710	1277
14	1140	1220	1220	1140	1505	1580	1163	1334	1710	1277
15	1140	1220	1220	1140	1595	1643	1163	1334	1710	1290
16	1140	1220	1244	1140	1710	1676	1163	1140	1140	1140
17	1140	1140	1140	1140	1140	1140	1140	1140	1300	1140
18	1140	1140	1140	1140	1174	1174	1140	1140	1710	1140
19	1140	1140	1140	1140	1233	1233	1186	1151	1710	1140
20	1140	1151	1140	1140	1233	1233	1186	1151	1710	1140
21	1140	1151	1140	1254	1233	1233	1186	1151	1710	1140
22	1140	1151	1140	1254	1233	1233	1186	1151	1710	1140
23	1140	1151	1140	1254	1233	1369	1186	1140	1140	1140
24	1140	1140	1140	1140	1140	1140	1140	1151	1151	1140
25	1140	1151	1163	1140	1254	1254	1140	1197	1289	1140
26	1140	1209	1291	1140	1354	1354	1140	1257	1573	1140
27	1140	1221	1368	1140	1435	1435	1140	1257	1573	1140
28	1140	1221	1505	1140	1435	1435	1140	1257	1573	1140
29	1140	1221	1505	1186	1478	1478	1231	1257	1573	1163

Iteration	noCD	BBB w/ naive	BBB w/ ddm	BBB w/ eddm	GB w/ naive	GB w/ ddm	GB w/ eddm	LSTM w/ naive	LSTM w/ ddm	LSTM w/ eddm
30	1140	1319	1565	1186	1641	1478	1231	1140	1140	1140
31	1140	1140	1140	1140	1140	1140	1140	1140	1140	1140
32	1140	1140	1140	1197	1311	1140	1140	1140	1140	1140
33	1140	1140	1140	1197	1416	1197	1197	1140	1140	1140
34	1140	1140	1140	1197	1473	1197	1197	1140	1140	1140
35	1140	1140	1140	1197	1502	1257	1197	1140	1140	1140
36	1140	1140	1140	1197	1502	1345	1197	1140	1140	1140
37	1140	1140	1140	1197	1502	1426	1197	1140	1140	1140
38	1140	1140	1140	1140	1140	1140	1140	1243	1140	1140
39	1140	1140	1140	1140	1197	1197	1140	1243	1140	1140
40	1140	1140	1140	1140	1197	1197	1254	1243	1140	1140
41	1140	1140	1140	1140	1197	1197	1254	1243	1140	1140
42	1140	1140	1140	1140	1197	1197	1254	1243	1140	1140
43	1140	1140	1140	1140	1197	1197	1254	1243	1197	1140
44	1140	1208	1208	1140	1209	1209	1254	1140	1140	1140
45	1140	1140	1140	1140	1140	1140	1140	1186	1710	1140
46	1140	1140	1140	1140	1140	1140	1140	1186	1710	1140
47	1140	1140	1140	1140	1140	1140	1140	1186	1710	1140
48	1140	1140	1140	1140	1140	1140	1140	1186	1710	1140
49	1140	1140	1140	1186	1140	1140	1163	1186	1710	1140
50	1140	1140	1140	1186	1231	1231	1163	1210	1710	1140
51	1140	1151	1140	1186	1416	1416	1163	1140	1140	1140
52	1140	1140	1140	1140	1140	1140	1140	1140	1277	1140
53	1140	1140	1140	1140	1288	1288	1140	1140	1354	1140
54	1140	1140	1140	1208	1288	1288	1140	1151	1354	1140
55	1140	1140	1140	1208	1288	1288	1140	1151	1354	1140
56	1140	1140	1140	1208	1288	1288	1140	1151	1354	1322
57	1140	1140	1140	1208	1288	1288	1140	1151	1354	1322
58	1140	1140	1140	1208	1288	1288	1140	1140	1140	1140
59	1140	1140	1140	1140	1140	1140	1140	1243	1277	1140
60	1140	1231	1186	1140	1265	1265	1140	1318	1277	1140
61	1140	1231	1257	1140	1265	1265	1140	1358	1277	1140
62	1140	1231	1257	1140	1265	1265	1151	1426	1277	1174

Iteration	noCD	BBB w/ naive	BBB w/ ddm	BBB w/ eddm	GB w/ naive	GB w/ ddm	GB w/ eddm	LSTM w/ naive	LSTM w/ ddm	LSTM w/ eddm
63	1140	1231	1257	1197	1265	1265	1151	1455	1277	1174
64	1140	1231	1257	1197	1265	1265	1151	1455	1277	1174
65	1140	1231	1257	1197	1354	1392	1151	1140	1140	1140
66	1140	1140	1140	1140	1140	1140	1140	1197	1208	1140
67	1140	1186	1186	1140	1197	1197	1140	1197	1208	1140
68	1140	1186	1186	1140	1221	1269	1140	1221	1208	1140
69	1140	1186	1186	1140	1368	1383	1140	1221	1208	1140
70	1140	1186	1186	1151	1601	1494	1140	1306	1280	1208
71	1140	1186	1186	1151	1601	1494	1140	1384	1434	1208
72	1140	1186	1186	1151	1601	1524	1140	1140	1140	1140
73	1140	1140	1140	1140	1140	1140	1140	1140	1277	1140
74	1140	1174	1174	1140	1163	1163	1140	1140	1341	1140
75	1140	1174	1174	1140	1198	1163	1140	1140	1462	1140
76	1140	1174	1174	1140	1198	1163	1140	1140	1462	1265
77	1140	1174	1174	1140	1246	1210	1140	1140	1506	1265
78	1140	1174	1174	1140	1271	1234	1140	1140	1596	1265
79	1140	1174	1174	1140	1271	1308	1140	1140	1140	1140
80	1140	1140	1140	1140	1140	1140	1140	1231	1402	1140
81	1140	1163	1163	1140	1265	1265	1140	1329	1598	1140
82	1140	1163	1163	1140	1303	1316	1140	1329	1710	1140
83	1140	1163	1163	1140	1342	1329	1140	1329	1710	1163
84	1140	1163	1163	1140	1355	1342	1140	1329	1710	1163
85	1140	1163	1163	1140	1355	1342	1140	1329	1710	1163
86	1140	1163	1163	1197	1423	1382	1140	1140	1140	1140
87	1140	1140	1140	1140	1140	1140	1140	1197	1231	1140
88	1140	1140	1140	1140	1208	1208	1140	1257	1342	1140
89	1140	1163	1163	1140	1329	1329	1140	1446	1463	1140
90	1140	1186	1186	1140	1488	1488	1140	1446	1710	1140
91	1140	1198	1198	1140	1667	1710	1140	1489	1710	1140
92	1140	1198	1198	1140	1710	1710	1140	1608	1710	1140
93	1140	1198	1198	1140	1710	1710	1140	1140	1140	1140
94	1140	1140	1140	1140	1140	1140	1140	1140	1140	1140
95	1140	1140	1140	1140	1151	1151	1140	1140	1140	1140

Feature table for Meta-Classification/Regression, where "MAX" (response variable) denotes the maximum expected KPI per iteration/day in the simulation

Day	week	Iteration	BBB_DDM	LSTM_naive	LSTM_edm	Max	UW	H	SU	L	HL	DU	LT	LU	M	DHL	DL	PTU	P	U	W	TU	CDL	RU	DLU	CDLU	LR	LS	MU
6	1	1	0.92	0.89	0.89	0.92	2.17	0.54	0.33	8.21	0.42	0	17	2.42	4.75	1.75	10.13	0	1.17	0.75	1.58	13	0	0.25	0	2	0	0	0
7	1	2	0.85	0.78	0.78	0.89	6.46	3.83	0.46	11.08	0	0	17	1.92	4.21	2.71	3.67	0	0	0.67	3.38	13	0	0.33	0	2	0	0	0
1	2	3	0.88	0.72	0.72	0.88	3.33	2.5	0.5	15.33	0	0	6.17	0.92	7.21	5.21	3.92	0	0	0.42	1.75	4	0	0	0	2	0	0	0
2	2	4	0.77	0.6	0.6	0.77	0	0	0.88	14.63	0.83	7.42	8.79	0.58	10.88	2.96	10.58	3	0	0	0	4.92	0	1.21	1.17	3.17	0	0	0
3	2	5	0.63	0.67	0.56	0.81	0	0	0.88	9.54	0.42	3.17	15.17	0.75	4.38	1.13	14.88	3.79	0.83	0	0	11.17	0	0	0.83	4	0	0	0
4	2	6	0.72	0.81	0.85	0.87	0	0	0	11.04	0	5.96	16	0.67	1.17	4	11.71	1.88	0.08	0	0	12	0	1.21	0.67	4	0	0	0
5	2	7	0.82	0.75	0.71	0.82	3	0.42	3.33	7	0.58	2.17	6.08	0	1.54	1.25	12.13	1	0	0.75	0	4.17	1.25	0.83	3.29	4	0.42	0	0.58
6	2	8	0.71	0.65	0.76	0.81	4.21	0.25	3.88	4.96	0	0	5.25	1.04	1.58	1.92	13.21	0	0	0.33	0	3.25	2.33	0	0.83	4	0.25	0	0.42
7	2	9	0.81	0.79	0.87	0.87	5.38	1.83	0.63	1.67	0	0	6	2.33	1.08	0.33	5.63	0	2.54	0.42	1.67	4	0	0	0	4	0	1.17	0
1	3	10	0.86	0.78	0.82	0.86	7.92	1	0.46	4.33	0.58	0	2.54	1.21	1.92	0	2.96	0	0.92	0.08	0.67	2.46	0	0	0	4.5	0	2	0
2	3	11	0.82	0.58	0.67	0.82	0	0	0.75	11.54	0	4.67	7.67	1.42	3.13	1.5	7.88	1.33	0.79	0.17	0	4.08	0	2.38	3.21	7.17	0	0.58	0
3	3	12	0.68	0.82	0.8	0.82	3.83	0	1.04	14.54	0.5	2.04	11.38	0.08	4.13	3.79	16.88	1.83	1.58	0	0	7.92	1.71	0.25	3.04	8	0	0	0.5
4	3	13	0.79	0.81	0.76	0.85	10.33	1.79	0.67	21.04	0	0	9.17	4.67	6.92	4.33	13.92	0	1.04	0.58	6.58	9.17	0	0.17	0	8	0	0	0
5	3	14	0.76	0.76	0.76	0.76	4.83	0.75	1.33	12.54	0	0	10	1.88	10.21	1.96	5.46	0	0.67	0	2.08	10	0	0	0	8	0	0	0
6	3	15	0.73	0.77	0.74	0.78	4.29	1.75	0.42	9.25	0.33	0	11.96	2.33	6.29	3.54	5.96	0	0.42	0.58	3.08	11.63	0	0	0	8	0	1.17	0
7	3	16	0.63	0.81	0.69	0.88	12.46	4.5	0	12.83	0.25	0	15	2.08	0.17	7.75	0.21	0	0.5	2.13	3.96	14	0	0	0	8	0	2	0
1	4	17	0.78	0.81	0.76	0.82	2.96	0.5	0.63	5.71	0	0	8.25	1.92	3.58	1.13	4.08	0	0	0.42	0.25	8	0	0.5	0	2.67	0	1.17	0
2	4	18	0.82	0.74	0.74	0.83	0	0	0.54	15.5	0	5.17	5.79	1.33	5.38	0	10.33	0.42	0	0.17	0	5.17	0	1.75	1	1	0	0	0
3	4	19	0.8	0.85	0.79	0.86	0	0	1.17	10.54	1	3.08	7	0.33	7.71	0	13.88	1	0.42	0.04	0	6	0	0.25	0	2	0	0	0
4	4	20	0.77	0.76	0.52	0.89	7.13	1.08	0	5.67	0	1.58	9.5	1.75	1.5	0	5.83	0.33	0.5	0.75	0.83	6	0	0.96	1.83	2	0	0	0
5	4	21	0.87	0.86	0.86	0.87	2.13	0.17	0	3.83	0	0.67	12.46	0	1.17	1	10.13	0.83	0.5	0.75	0	8.46	2.46	0.63	2.5	2	0.17	0	0
6	4	22	0.83	0.89	0.9	0.9	6.67	1.83	0.46	6.33	0	0	12.67	2.75	2.29	1.83	14.83	0	1.42	0.5	4.75	9	0	0	0.83	2	0	1.79	0
7	4	23	0.88	0.74	0.77	0.88	4.58	1.67	0.88	6.33	0	0	2.75	1.38	6.04	0.42	5.38	0	0.42	0	0	2.75	0	0	0	2	0	3	0
1	5	24	0.73	0.71	0.67	0.73	0	0	0.08	7.67	0	4.5	4.25	1	6.75	0.33	10.21	0.83	0.67	0	0	4	0	1.71	2.13	2.29	0	1.58	0
2	5	25	0.64	0.7	0.7	0.7	6.58	1.79	0.75	12.83	0	3.96	5.42	4.58	7.46	0.58	17.63	2.75	0	1.79	3.04	3.58	0	1.25	2.92	4	0	0	0
3	5	26	0.69	0.75	0.73	0.82	0.17	0.08	0.96	13.54	0	1.38	5.63	1.5	2.58	2	18.75	2	0	0.08	0	3	0	0.42	2.75	4	0	0	0
4	5	27	0.65	0.85	0.82	0.85	0	0	0.29	5.83	0.67	0	4.29	2.5	2.83	0.33	10.17	0	0	0.08	0	4.13	0	0.75	0	4	0	0	0
5	5	28	0.8	0.77	0.77	0.84	7	1.13	1	6.75	0	0	7.5	0.25	4	0.33	8.17	0	0	1.71	0	7.5	3.13	0.17	0.25	4	1.13	0	0
6	5	29	0.71	0.8	0.76	0.8	2.71	0.17	0.42	9.92	0	3	12.46	2.25	7.13	0.42	10.88	2.33	0	0.33	0	11	0	1.46	1.58	4	0.17	0	0
7	5	30	0.77	0.83	0.8	0.83	0	0	0.58	7	0	0	14.25	0.92	7.58	1.75	12.33	0.33	0	0	0	12.25	0	0	0.08	3.58	0	0	0
1	6	31	0.8	0.81	0.81	0.81	0	0	0.46	11.33	0	0	17.17	0.96	7.88	5.17	3.79	0	0.5	0.08	0	15.17	0	0	0	1.17	0	0	0
2	6	32	0.79	0.76	0.77	0.8	6.04	2.08	0	16.63	0	0	18	1.67	10.54	5.33	5.38	0	3.13	0.67	3.17	16	0	0	0	0	0	0.79	0
3	6	33	0.79	0.83	0.79	0.83	8.67	5.71	0.33	10.46	0	0	13.29	6.08	10.17	2.75	4.67	0	0.88	0	3.46	12.83	0	0	0	0	0	3	0
4	6	34	0.78	0.83	0.82	0.83	5.71	2.54	0.5	9.21	0	0	15	0.75	8.21	2.63	5.58	0	0	0	1.33	15	0	0	0	0	0	2.75	0
5	6	35	0.82	0.84	0.85	0.85	3.79	0.33	1.13	5.21	0	0	18.33	0	6.75	0.58	6.33	0	0	0.33	0	18.33	1.79	0.33	1	0	0.33	0	0
6	6	36	0.84	0.91	0.92	0.93	6.25	0	0.83	5.92	0	0	22.17	1.46	2.88	1.17	8.96	0	0	0.17	0	22.17	0.96	0	1.92	0	0	0	0
7	6	37	0.91	0.66	0.89	0.91	4.5	0	0	1.5	0	0	25.17	0.5	6.92	0	10.71	0	0.58	0	0	25.17	1.08	0	0.67	0	0	0	0
1	7	38	0.9	0.9	0.89	0.9	9.33	0.33	0	5.83	0	0	26.92	0.33	7	0	5.21	0	0	0.33	0	26.92	1.58	1	1.08	0	0.33	0	0
2	7	39	0.89	0.91	0.91	0.92	18.21	0	0.5	5.83	0	0	30.79	0	7	0	6.71	0	0	0	0	30.79	2.04	0.25	0.08	0	0	0	0.5
3	7	40	0.91	0.96	0.96	0.96	26.58	0.83	0	6.67	0	0	33	1.33	3.71	0.33	10.46	0	0.5	1.38	0	33	0.58	0	0	0	0.83	0	0
4	7	41	0.95	0.88	0.86	0.95	27.42	1.25	1.58	8.04	0	3.08	27.83	0.5	7.33	0.42	9.38	1	0	1.25	0	15.13	1.46	1.13	4.08	0.33	1.25	0	0
5	7	42	0.85	0.76	0.69	0.85	2.38	0.08	1.42	9.29	0	7.96	7.96	0	6.88	0	21.5	4.92	0	0.08	0	0	0.46	0.58	7.63	2	0.08	0	0.33
6	7	43	0.69	0.79	0.72	0.86	10.42	0	0.08	11.17	0	11.17	11.17	0	0	0	34.25	3.25	0	0	0	0	2.04	1.17	11.17	3.67	0	0	0
7	7	44	0.86	0.71	0.68	0.86	22	0.5	2.46	8.54	0	8.54	8.54	0	0	0	43.46	7.42	0	0.5	0	0	1.25	0	5.13	7.33	0.5	0	0
1	8	45	0.9	0.86	0.81	0.9	0.92	0	1.08	14.79	0	10.13	10.79	0	3.54	1.08	50.17	6.25	0.42	0.08	0	0	0.96	1.17	2.63	11.17	0	0	0
2	8	46	0.86	0.88	0.79	0.89	1.29	0.42	0.25	20.25	0.58	10.04	12.5	0	2.08	2.92	62.38	3.5	0	0.42	0	1.17	1	1.25	4.88	12.92	0.42	0	0
3	8	47	0.88	0.86	0.8	0.88	0	0	0.08	23.96	0	8.08	13.79	0.58	3.17	4.17	34.33	4.08	0	0	0	3.5	0	1.04	1.83	14	0	0	0
4	8	48	0.86	0.8	0.59	0.86	0	0	0.25	12.96	0	10.83	21.5	0	6.92	1	17.79	3.58	0	0	0	7.17	0	1.25	1.75	15.25	0	0	0
5	8	49	0.81	0.73	0.68	0.81	0	0	0.08	13.5	0.58	12.33	26.08	0.96	12.67	0	11.42	7.08	0.42	0	0	9.79	0	0	3.58	16	0	0	0

Day	week	Iteration	BBB_DDM	LSTM_naive	LSTM_eddm	Max	UW	H	SU	L	HL	DU	LT	LU	M	DHL	DL	PTU	P	U	W	TU	CDL	RU	DLU	CDLU	LR	LS	MU
6	8	50	0.78	0.75	0.59	0.78	5.54	0.25	0.67	18.88	0	12.67	29	0.92	7.08	1.17	15.33	14.08	0	0.33	0	12.08	1.25	0.67	5.79	17.17	0.25	0	0
7	8	51	0.82	0.79	0.7	0.82	2.92	0	1.04	12.46	0.5	4.54	23.46	0	2.83	1.08	17.08	9.63	0.92	0	0	14.17	1.42	0	4.5	18	0	0	0.5
1	9	52	0.82	0.76	0.61	0.93	3.58	0	0.92	5.33	0	0	22.92	0.5	3.33	0.58	25.63	0	0	0.33	0	17.92	2.04	0	0	18	0	0	0.58
2	9	53	0.85	0.74	0.86	0.93	4.54	0.33	0.75	3.13	0	0	27	0	2.63	1.46	28.75	0	0	1.46	0	22	0.58	0.17	1	18	0.33	0	0
3	9	54	0.86	0.91	0.91	0.97	2.75	0.42	0.25	4.83	0	0	29.58	0.58	0.58	0	37.58	0	0.42	0.42	0	24.58	0.33	0.25	0	18	0.42	0	0
4	9	55	0.9	0.92	0.91	0.94	12.04	0.75	0	6.67	0	0	35.33	0.58	2.29	0	45.54	0	0	0.75	0	30.33	1.5	0	0	18	0.75	0	0
5	9	56	0.91	0.94	0.93	0.95	8.75	0.25	0.33	3.54	0	0	37.58	0.83	4.5	0.96	51.63	0	0	0.25	0	32.58	1.38	0	0	6	0.25	0	0
6	9	57	0.93	0.87	0.86	0.93	5.54	0.79	0	1.42	0	0	10.79	1.33	4.92	0	10.38	0	0	1	0	9.54	1.96	0	0.75	0	0.79	0	0
7	9	58	0.84	0.66	0.71	0.84	8.33	0.88	0.46	5.83	1.58	0	3.58	0.33	9.63	0	12.29	0	0.42	1.88	0	3.58	4.13	0.17	0.25	0	0.88	0	0
1	10	59	0.71	0.69	0.66	0.92	7.33	0.33	0	4.92	0	0	1.33	0.75	1.29	1	11.63	0	0	0.33	0	0.33	2.29	0.25	0.92	0	0.33	0	0
2	10	60	0.69	0.72	0.76	0.82	2.29	0.17	0	12.46	0	5.96	8.63	0	3.33	2.08	20.42	1.88	0.5	0.75	0	1.92	3.71	1.88	0.92	1.17	0.17	0	0
3	10	61	0.75	0.7	0.78	0.78	3	0.42	3.33	7.96	0.58	2.17	9.63	0	5.5	1.25	27.13	1	0	0.75	0	5.17	1.25	0.83	3.67	2	0.42	0	0.58
4	10	62	0.78	0.73	0.85	0.91	4.21	0.25	3.88	4.96	0	0	11.25	1.04	1.58	1.92	23.83	0	0	0.33	0	7.25	2.33	0	0.83	2	0.25	0	0.42
5	10	63	0.85	0.83	0.89	0.91	5.38	1.83	0.63	1.67	0	0	12	2.33	1.08	0.33	5.63	0	2.54	0.42	1.67	8	0	0	0	2	0	1.17	0
6	10	64	0.9	0.68	0.83	0.9	7.92	1	0.46	4.33	0.58	0	2.79	1.21	1.92	0	2.96	0	0.92	0.08	0.67	2.63	0	0	0	2	0	2	0
7	10	65	0.84	0.7	0.7	0.84	0	0	0.75	11.54	0	4.67	7.67	1.42	3.13	1.5	7.88	1.33	0.79	0.17	0	4.08	0	2.38	3.21	3.17	0	0.58	0
1	11	66	0.71	0.75	0.68	0.83	3.83	0	1.04	14.54	0.5	2.04	11.38	0.08	4.13	3.79	16.88	1.83	1.58	0	0	7.92	1.71	0.25	3.04	4	0	0	0.5
2	11	67	0.81	0.73	0.86	0.86	10.33	1.79	0.67	21.04	0	0	9.17	4.67	6.92	4.33	13.92	0	1.04	0.58	6.58	9.17	0	0.17	0	4	0	0	0
3	11	68	0.85	0.76	0.76	0.85	4.83	0.75	1.33	12.54	0	0	10	1.88	10.21	1.96	5.46	0	0.67	0	2.08	10	0	0	0	4	0	0	0
4	11	69	0.77	0.68	0.81	0.83	4.29	1.75	0.42	9.25	0.33	0	11.96	2.33	6.29	3.54	5.96	0	0.42	0.58	3.08	11.63	0	0	0	4	0	1.17	0
5	11	70	0.81	0.69	0.89	0.89	12.46	4.5	0	12.83	0.25	0	15	2.08	0.17	7.75	0.21	0	0.5	2.13	3.96	14	0	0	0	4	0	2	0
6	11	71	0.87	0.82	0.79	0.87	2.96	0.5	0.63	5.71	0	0	8.25	1.92	3.58	1.13	4.08	0	0	0.42	0.25	8	0	0.5	0	1.33	0	1.17	0
7	11	72	0.82	0.78	0.7	0.82	6.75	3.92	1.71	15.5	0	8	5.79	4.46	5.38	0.42	13.46	3.29	0.33	1.71	1.71	5.17	1.67	2.46	2.29	1	1	1.25	0.42
1	12	73	0.72	0.84	0.82	0.87	15.63	3.71	0	13.38	0	11.63	7	1.75	2.04	0.42	19.08	4.42	0	0	3.17	6	0.58	0.83	5.08	2	0	3.17	0
2	12	74	0.79	0.85	0.89	0.89	17.29	4.13	0	12.79	0	6.88	5.88	2.67	0	1.42	27.33	0.33	0	0.5	6.13	1	1.46	0.75	10	2	0	4.92	0
3	12	75	0.84	0.8	0.86	0.86	38.88	3.88	0	23.38	0	5.04	9.08	2.71	0	6.42	16.96	3.92	0	0.67	5.58	0	1.29	0.96	17.88	2	0.33	7.17	0
4	12	76	0.86	0.84	0.78	0.86	27.17	4.33	0.5	32	0	4.75	11.83	4.08	0	9	13.08	2.13	0.83	0.67	5.5	0	1.33	0.33	8.92	2	0.67	9.17	0.5
5	12	77	0.86	0.85	0.81	0.87	26.71	6.46	0.33	40.75	0	7.5	13.83	0.92	0	13.63	9.17	3.33	1.33	2.79	7.04	0	0.38	0.83	6.21	2	0.75	10	0.33
6	12	78	0.86	0.82	0.74	0.86	42.04	13.08	0.46	45.13	0	6.58	15	2.88	5.21	17.33	9.08	3.5	0.58	2.63	2.96	0	0.33	0.92	7.08	3.63	0.5	2.92	0
7	12	79	0.85	0.67	0.67	0.85	26.63	5.67	0.25	21	0	7.96	7.13	2.71	12.67	3.63	19.08	4.92	3.25	0	5.46	1.83	0	0.58	5.71	12	0	1.08	0
1	13	80	0.73	0.67	0.72	0.73	11.83	2.58	0.96	27.54	0.33	11.17	13.75	2.67	11.17	5.04	34.83	3.25	0.33	0.5	4.79	5.75	0	1.17	8.17	13.67	0	3.17	0
2	13	81	0.79	0.76	0.77	0.79	12.63	5.5	0.38	17.71	0	8.54	19.21	0.5	2.38	1.46	45.88	7.42	0	1.17	4.29	10.08	0	0	4.63	17.33	0	4	0
3	13	82	0.85	0.76	0.77	0.85	6.25	4.5	0.42	25.17	0	10.13	25.08	2.79	4.38	5.08	50.58	6.25	0.92	0.08	3.17	12	0	0.83	0.96	21.17	0	2.83	0
4	13	83	0.83	0.81	0.83	0.85	14.67	2.83	0.5	31.42	0.58	10.04	28.08	2.25	2.08	7.83	61.63	3.5	1.33	0	3.42	3.67	0	0.75	3.13	22.92	0	4	0
5	13	84	0.85	0.78	0.78	0.85	18.71	3.08	0.08	36.5	0	8.08	31.5	2.71	3.17	10.79	32.38	4.08	1.04	0.67	1.88	3.5	0	1.04	1.75	24	0	2.83	0
6	13	85	0.83	0.76	0.75	0.83	13.08	3.17	0	24	0	10.83	34	2.58	1.08	1.25	16.42	3.58	3.46	0.42	2.08	6	0	1.25	1.75	25.25	0	1.17	0
7	13	86	0.83	0.7	0.81	0.83	20.29	4.33	0	14.5	0	12.33	12.25	3.58	0	1.17	11.13	7.08	1.5	0.67	6.04	0.75	0	0	3.58	26	0	2.67	0
1	14	87	0.79	0.7	0.77	0.79	14.38	5.13	0	18.17	0	12.67	9.83	2.17	0	1.83	9.33	14.08	1	0	6.67	0	0	0.67	5.79	27.17	0	4	0
2	14	88	0.73	0.6	0.73	0.73	10.88	2.92	0	10.79	0	8.58	8.58	1.83	0	0.5	8.75	21	0.5	0	3.08	0	0	0.58	6.83	28	0	4.67	0
3	14	89	0.73	0.75	0.74	0.75	16.63	4.21	0	14.38	0	7.63	7.63	4.75	0	1.13	9.46	27.75	0.92	0.58	5.5	0	0	1.42	4.17	30.13	0	6	0
4	14	90	0.74	0.72	0.76	0.79	14.75	2.54	0	13.33	0	9.67	9.96	3.92	0	2.25	21.58	20.46	1	2.92	2.17	0	0	0.58	4.5	32	0	3.25	0
5	14	91	0.76	0.67	0.9	0.9	23.17	2.83	0	23.54	0	4.67	19.75	3.83	0	8.54	32.88	0	0.17	0.58	2.92	0	0	0.71	3.58	33.25	0	1.17	0
6	14	92	0.9	0.9	0.89	0.9	15.46	2.75	0	32.33	0	3.63	21.92	1.5	0	14.33	46.54	0.42	0.5	1.67	3.92	0	0	0.17	5.88	34.92	0	2	0
7	14	93	0.89	0.89	0.87	0.89	10.75	2.25	0	12.58	0	6.54	5.92	3.33	0	2.33	58.33	3.54	0	0.92	6.21	0	0	0.33	11.54	36.33	0	1.71	0
1	15	94	0.86	0.89	0.89	0.9	7.83	1.92	0	18.96	0	6.75	7.67	1.38	0	4.67	71.25	2.92	2.58	2.96	5.83	0	0	1.25	3.5	39	0	3	0
2	15	95	0.88	0.95	0.95	0.95	6.25	1.67	0	6.21	0	2.58	2.58	0.71	0	0	40	1	0.25	0.25	1.96	0	0	0.21	0	40	0	1.83	0