Dimensionality Reduction and Model Selection for Click Prediction

Author: Aniket Mitra|2205941
Supervisor: dr. Auke Pot
Supervisor: dr. Evert Haasdijk
Reader: dr. Fetsje Bijma

Faculty of Sciences
1081 HV Amsterdam

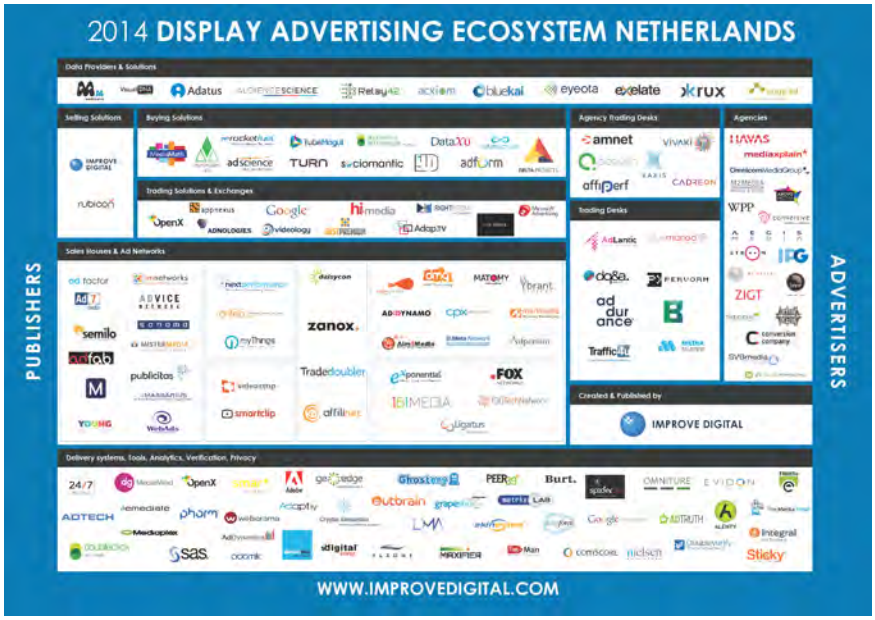
Basisweg 52D
1034 AP Amsterdam

Abstract: *As part of this research, an Insights System has been built, using an ensemble of Big Data Technologies and Machine Learning Algorithms. The Insights System can be used to process large data sets (impressions, bid requests), create viewer history and compute the values of various derived attributes. The system further facilitates feature and model selection using different statistical techniques and machine learning algorithms, thereby enabling us to identify, a set of relevant features and a model, that estimates the likelihood of whether an impression will result in a click or not.*

Keywords: Hadoop, MongoDB, HIVE, MapReduce, Feature Selection, Naïve Bayes, Logistic Regression, Support Vector Machines

Executive Summary

The advent of auction-based media buying is fast transforming the landscape of digital media, thereby allowing advertisers and publishers to obtain contextual relevance in advertisements, which a user sees on a website. Such customer marketing effectiveness, combined with process efficiency, promises better accountability for advertisers on their ad spends. As per the Internet Advertising Board report published in 2013, the Dutch online advertising market was €619m in H1 2013, a H/H growth of 5.8%.Moreover, the report expected the Dutch display advertising market to grow at 6% in 2014 mostly driven by high growth rates in the auction-based media buying. Auction-based media buying has led to the creation of an ecosystem comprising of Agencies, Trading Desks, Data Providers, Buying and Selling Solutions Providers as shown in the Display Advertising Ecosystem of Netherlands.



Adscience, which is a buy side solutions provider, connects to multiple sell side solutions providers (that run the auctions) and participates in auction-based media buying on behalf of media agencies /advertisers. Therefore, Adscience strives to combine media buying efficiency with display advertisement effectiveness in order to deliver greater accountability to advertisers on their ad spend. Moreover, a Winterberry Group White Paper published in 2013, revealed that the 260 executive-level marketers, technologists and media industry leaders that it interviewed, lay a high emphasis on applying programmatic approached to audience segmentation (91% of respondents) and actionable insight development (88% of respondents).This thesis will provide details of an Insights System that has been built for Adscience and using the system, different approaches for audience segmentation will be explored. Moreover, using the insights system, different attribute selection methods will be explored, that enhances the effectiveness of a display advertisement. This system will further streamline the media buying efficiency of Adscience by allowing them to identify the optimum bid price of an auction.

Preface

This thesis is part of my Master's degree program at VU University, where each student is required to perform a research, as part of the final semester of their study. I would like to extend my gratitude to dr. Auke Pot for his supervision, dr. Evert Haasdijk for his guidance and dr Fetsje Bijma for her advice throughout. I am grateful to all my colleagues at Adscience for being supportive during the period of the internship.

Aniket Mitra
Aniketmitra001@gmail.com
January, 2014

Contents

1. Introduction

Realtime Bidding (RTB) '*allows advertisers to bid on the opportunity to show an advertisement to a specific user on a specific site with a display advertisement slot'* [51]. To help advertisers participate in realtime bidding, intermediaries exist called ***demand side platforms*** (DSP), that connect to multiple ad exchanges (which run the auctions), and participate in realtime bidding on behalf of the advertiser. The demand side platform can access inventory from these ad exchanges and decide whether or not they want to bid and at what price. An ***inventory*** is defined as the place where the advertisement will be shown. Each won bid is termed as an ***impression***. The goal is to identify if showing a particular advertisement to a viewer will result in a click and to determine the bid price accordingly. The likelihood of a viewer clicking on an advertisement depends on a number of different factors (attributes), and some factors have a greater influence than others in the final outcome (click/no-click). The goal can therefore be restated as "*identification of a set of factors that increase the likelihood of a viewer clicking on a particular advertisement*"

Adscience provides a demand side platform that enables media agencies and advertisers to participate in realtime bidding for display ads. The core business of Adscience is Marketing Accountability: demonstrating the relationship between marketing efforts on the one hand, and unmistakable additional returns on the other. Adscience is also specialized in real-time campaign management. NOAX, Adscience's optimization platform, explores and discovers such relationships and makes real-time decisions, or highlights these relationships to marketing and sales specialists. Adscience is affiliated with Ortec, a global market leader in software and consultancy in the field of planning and optimization [1].

NOAX, the real-time bidding algorithm of Adscience uses a Naïve Bayes algorithm to determine the bid price of a banner to be displayed for an inventory. The algorithm uses a set of base attributes (received as part of the bid requests) as well as derived attributes to determine the click probability of each viewer. The details regarding each won-bid (impression) and the outcome of it (no action, click, conversion) is registered and stored in the database. This information is processed frequently in order to update the statistics regarding each attribute. A set of delivery rules can be set that restricts the algorithm from exploring uninteresting audiences [4]. Moreover, NOAX allows campaign managers to define constraints on the budget spent, bid price, number of impressions etc. NOAX also allows campaign managers to do audience retargeting mainly on two kinds of attributes: the current url and the browsing pattern of the user [4]. As shown in Figure [1], Adscience provides an interface that allows campaign managers to create and manage campaigns while a collection of bidding and database servers, process and store bids information.

Adscience receives approximately 100M bid requests and serves more than 1M impressions per day. This number is set to grow rapidly as Adscience connects to more Ad Exchanges and therefore, gains access to global inventory in the near future. This allows Adscience access to large volumes of data and therefore opens up the possibility of mining these massive datasets [6]. Therefore, Adscience requires a system that can store and process the bid requests and impressions. Moreover, the data stored in this new system needs to be processed to compute the values of various attributes. Furthermore, the system should enable attribute selection using various machine learning techniques and also serve as a test bed for estimating the click probability of an impression using alternate algorithms like Logistic Regression and Support Vector Machines.

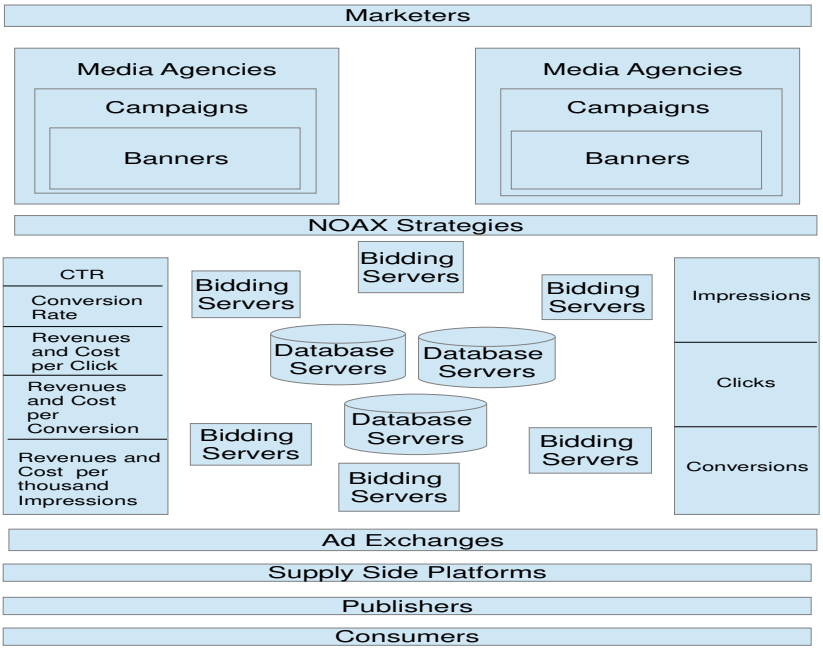


Figure 1: NOAX- An Overview

The aim of this research is to implement an Insights System using an ensemble of Big Data technologies and Machine Learning packages for feature selection and classifier comparison. The Insights System extracts the bids requests and impressions from the operational platform and stores it in a distributed file system (bid requests) and a NoSQL data store (impressions) [14][15]. This data is then transformed using a set of map-reduce jobs to compute the values of various attributes [16].The transformed data set (attribute values per impression) is then loaded into an environment suitable for statistical computing and graphics, wherein different machine learning packages are used to identify the attributes that best predict the likelihood of a viewer clicking on a display advertisement [17] [18].

A number of feature selection methods like Wrappers, Filters and Embedded methods have been evaluated for this purpose. Due to the high dimensionality and large number of impressions, it has been found that Wrappers have a high time complexity. Therefore, a unique method has been proposed, that combines the filter and wrapper methods wherein a filter method is used to reduce the dimensionality of the dataset by selecting attributes that have a high correlation with the outcome while a low correlation among themselves. The reduced feature set is then passed to a wrapper method that performs an iterative ROC analysis to identify the set of features that best predict the likelihood of a viewer clicking on a display advertisement. Moreover, the existence of class skew in the data set has led us to formulate a novel approach for ROC analysis by ranking impressions based on their estimated click probabilities. Finally, the click probability of each impression has been estimated with alternate algorithms like Logistic Regression and Support Vector Machines and performance of these algorithms have been compared against the Naïve Bayes algorithm [18][19][20]. The predictions of the models have been validated against the actual outcome of distinct validation sets (to assess robustness) and methods like ROC analysis and statistical significance tests have been used to draw conclusions on the performance of the models thereby allowing us to draw conclusions on the comparative analysis of the models. Moreover, the system is being used for other research purposes like viewer profiling and conversion attribution which are out of scope for this report.

2. The Insights System

2.1 Overview

The Insights System is an ensemble of Big Data Technologies and a Statistical Analysis Environment for processing and analyzing impressions data. As shown in Figure [2], the system processes approximately 100M+ bid requests, 1M+ impressions, their corresponding outcomes (no clicks, clicks, conversions) and computes the attributes values for each impression daily. Moreover, the system runs various machine learning algorithms on large volumes of impressions data to estimate the click probability for each impression.

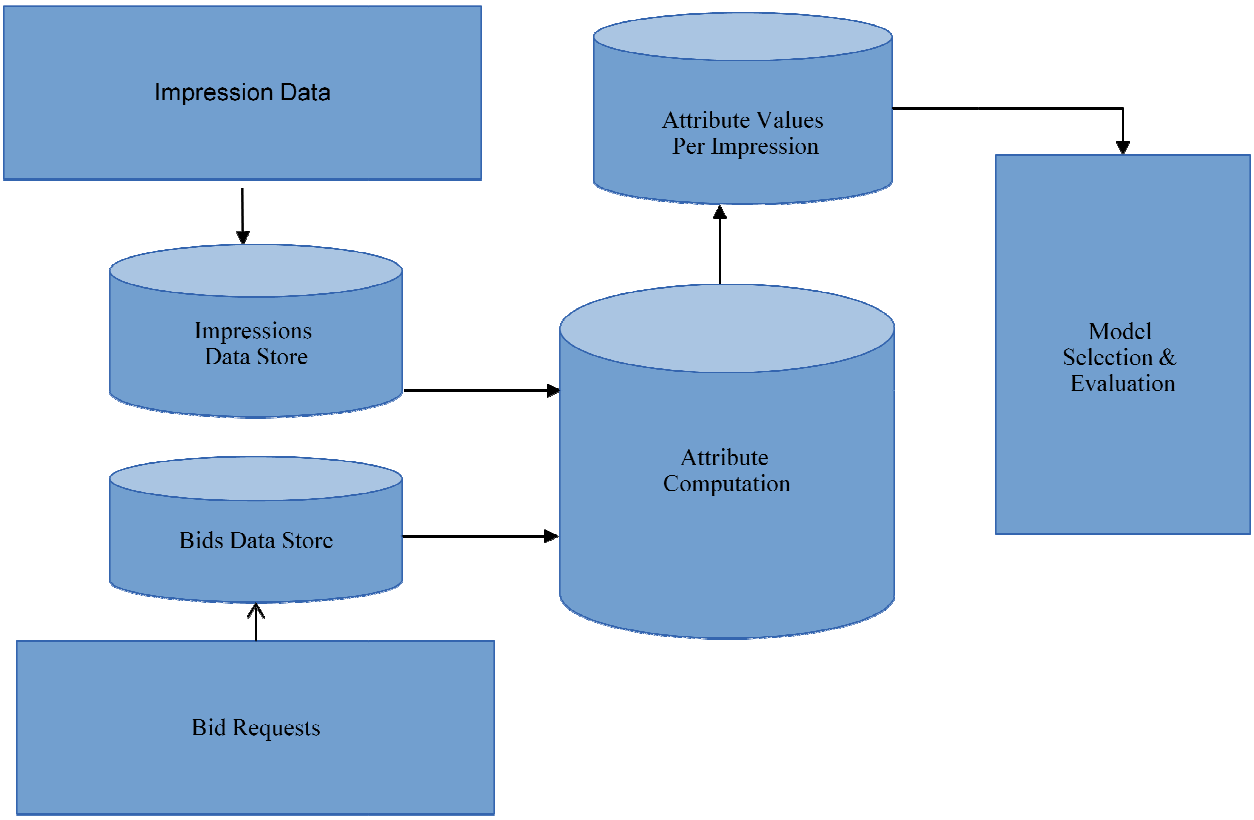


Figure 2: Insights System – An Overview

2.2 The ETL Process

2.2.1 Extraction

The extraction process, as shown in Appendix A, sources bid requests, impressions and events data (clicks, conversions and retargeting) from the operational system. Moreover, the extraction process also extracts domain categories and average time spent by a viewer on a webpage. Bid requests are extracted from each of the bidding server and contain the following information

- ✓ *Viewer Id*
- ✓ *Impression Id*
- ✓ *Banner Id*
- ✓ *Position of the Banner*
- ✓ *Size of the Banner*
- ✓ *Browser*
- ✓ *Domain Name*
- ✓ *IP Address*
- ✓ *Language*
- ✓ *Operating System*
- ✓ *SSP*
- ✓ *Timestamp*

The value of Banner Id indicates whether we bid for the requested impression or not. A Banner Id of -1 indicates that we did not bid for the impression. The Impression Id is a concatenation of the Timestamp and Viewer Id. Similarly, impressions data is extracted from the operational databases and contains the following information

- ✓ *Impression Id*
- ✓ *Viewer Id*
- ✓ *Timestamp*
- ✓ *Domain Name*
- ✓ *Estimated CTR* (Click Through Rate)
- ✓ *IP Address*
- ✓ *Operating System*
- ✓ *Banner Id*
- ✓ *Campaign Id*
- ✓ *Browser*

Moreover, events data (clicks, conversions & retargeting) is extracted which contains the following information

- ✓ Viewer Id
- ✓ Event Time

The extraction process extracts these data sets and stores them in the Insights System for further processing.

2.2.2 Transformation

The transform process, as shown in Appendix B, creates and stores viewer profile i.e impressions and its outcomes (click, no click, conversion, no conversion, retargeting) are aggregated per viewer. Creation of viewer profile transforms the attribute computation problem into an embarrassingly parallel problem such that the attributes for each viewer can be computed in parallel. Once the viewer profile (history) has been created, a large number of attribute can be computed for each viewer. Some of these attributes are mentioned below

- ✓ *Base Attributes:* These attributes are the default attributes present in the data extracted from the operational system. Examples of Base Attributes are:
  - ***BROWSER***
  - ***OS***
  - ***DOMAIN***
  - ***IP***
  - ***BANNER***
  - ***CAMPAIGN***
  - ***SSP***
  - ***DATE_TIME***
  - ***CTR***
  - ***DOMAIN_CATEGORY***

- ✓ *DateTime Attributes:* These attributes are derived from the timestamp at which the impression was served. Examples of such attributes are:
  - ***DATE***
  - ***DAY_OF_WEEK***
  - ***HOUR_OF_DAY***

- ✓ *GeoLocation Attributes:* These attributes are derived from the IP address of a viewer and provide information about the geographic location of a user. Examples of such attributes are :
  - ***COUNTRY***
  - ***LATITUDE***
  - ***LONGITUDE***
  - ***REGION***
  - ***CITY***
  - ***WEATHER:*** the geographic coordinates (Latitude, Longitude) are used to derive the weather information of the location in which the impression is served.

- ✓ *Count Attributes:* These attributes define the number of occurrences of a particular event per viewer, till the given impression was served. Examples of such attributes are:
  - ***NOF_IM_THIS_AD :*** Number of times the viewer has seen this banner

- ***NOF_IM_THIS_CAMPAIGN :*** Number of times the viewer has seen any banner belonging to this campaign
- ***NOF_IMPRESSIONS :*** Number of Impressions already served to the viewer
- ***NOF_CONVERSIONS :*** Number of conversions already registered for the viewer
- ***NOF_RETARGETING_REQUESTS :*** Number of times the viewer has been retargeted
- ***NOF_CLICKS :*** Number of clicks already registered for the viewer
- ***NOF_CONVERSIONS_THIS_AD :*** Number of conversions already registered for the viewer for this banner
- ***NOF_VISITS_THIS_DOMAIN :*** Number of times the viewer has visited this domain
- ***NOF_BID_REQUESTS :*** Number of bid requests received for the viewer
- ***NOF_VISITS_THIS_DOMAIN_CATEGORY :*** Number of times the viewer has visited this domain category
- ***NOF_CONVERSIONS_THIS_CAMPAIGN :*** Number of conversions already registered for the viewer for this campaign
- ***NOF_CL_THIS_CAMPAIGN :*** Number of clicks already registered for the viewer for this campaign
- ***NOF_CL_THIS_AD :*** Number of clicks already registered for the viewer for this banner
- ***NOF_BID_RESPONSES :*** Number of bid requests for the viewer that have been responded to

✓ *Flag Attributes:* These attributes indicate the occurrence of an event in the past for a viewer. Examples of such attributes are:

- ***HAS_CLICKED :*** A yes/no flags indicating whether a viewer has already clicked on a banner in the past
- ***HAS_CONVERTED :*** A yes/no flags indicating whether a conversion has been registered for the viewer in the past

2.2.3 Load

Since the main objective is to compute the click probability per impression, the load process maps the attribute values computed per viewer (in the transform step) to each impression served to the viewer thereby generating attribute values per impression. The attribute values per impression are then loaded into the statistical analysis environment wherein various machine learning algorithms are used to identify the attributes/algorithm pair that best predicts the likelihood of a viewer clicking on a display advertisement.

2.3 Setup of the Insights System

As described in Section 2.1, due to the large volume of data that has to be stored and processed by the Insights System, a set of parallel and distributed computing technologies is required. This section describes the different technologies that have been used in setting up the Insights System

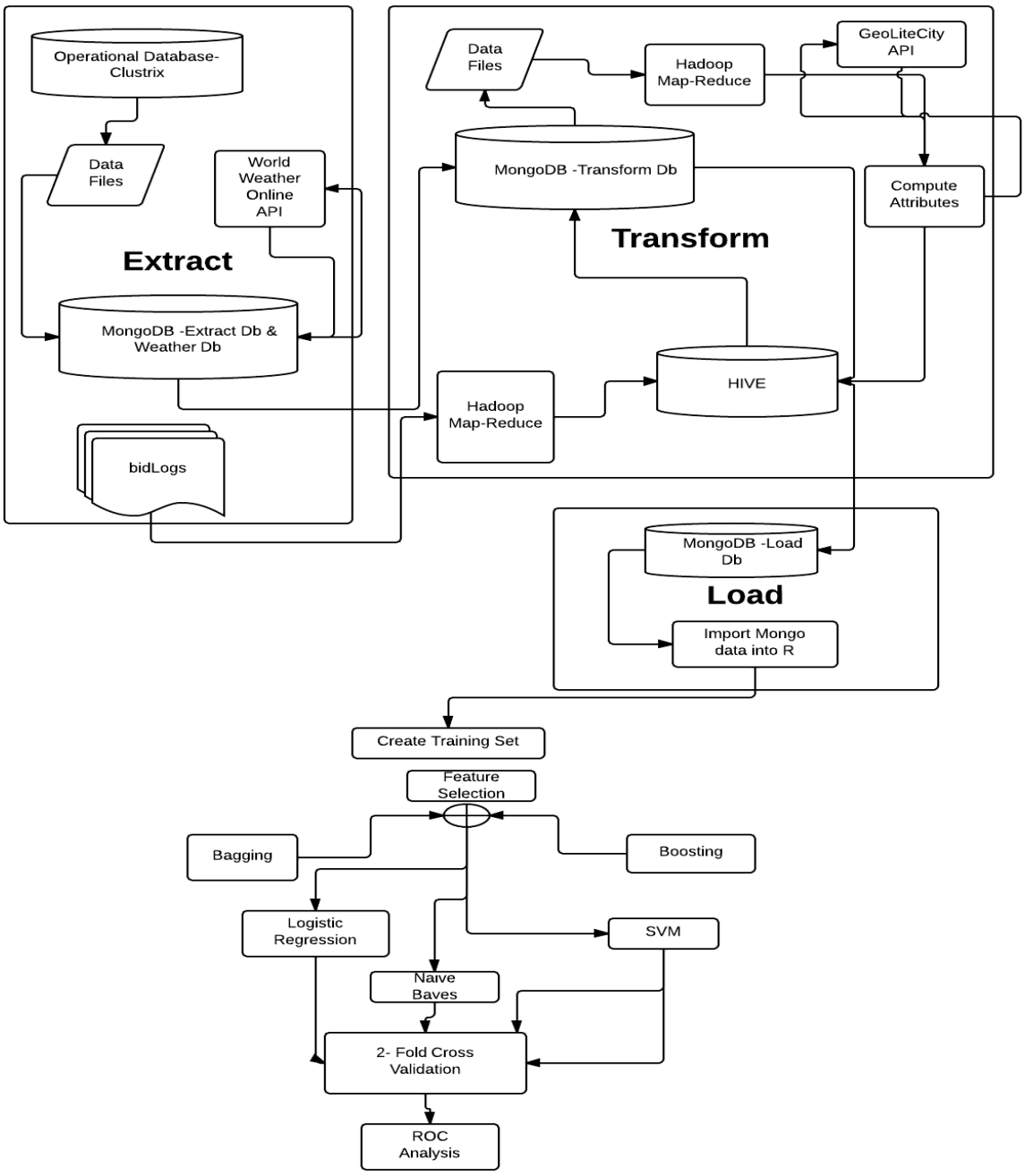


Figure 5: Architecture of the Insights System

2.3.1 Data Storage

The data model of the Insights System (see Appendix C) is implemented using MongoDB and HIVE which are described in this section. MongoDB is a NoSQL database which stores data in key-value format and offers a wide range of features [21]. Some of the features of MongoDB which make it appealing as the data store of the Insights System are

- ✓ *Document Storage:* Since most of the big data sets generated by the operational platform are in key-value format, the feature of MongoDB to store data as JSON documents makes it a good fit. The key-value structure of the data model is shown in Figure [8].

- ✓ *Indexing Support:* The attribute computation task requires accessing documents (records) from large collections of data. The indexing in MongoDB allows low latency access to documents and therefore is expected to speed up the data transformation process. Indexing in the data model is shown in Figure [8].

- ✓ *Horizontal Scaling:* Since the Insights System has to store large collections of data, this data needs to be split and stored as multiple smaller sets, in order to facilitate quick retrieval of data. Moreover, as the data size keeps growing, it should be feasible to scale horizontally by distributing the data over additional processing units. MongoDB supports this feature of horizontal scaling by allowing creation of shards and adding them to a cluster as shown in Figure [6] [22]. Once a shard has been added, the automatic sharding feature of MongoDB enables redistribution of data over this new (and larger) processing cluster. The use of shard key in the data model is demonstrated in Figure [8].

- ✓ *Querying:* MongoDB supports a large collection of data retrieval, aggregation and data insert/update queries which allows us to perform as lot of the ETL (extract, transform, load) tasks as described in Section 2.2.2, via Mongo queries.

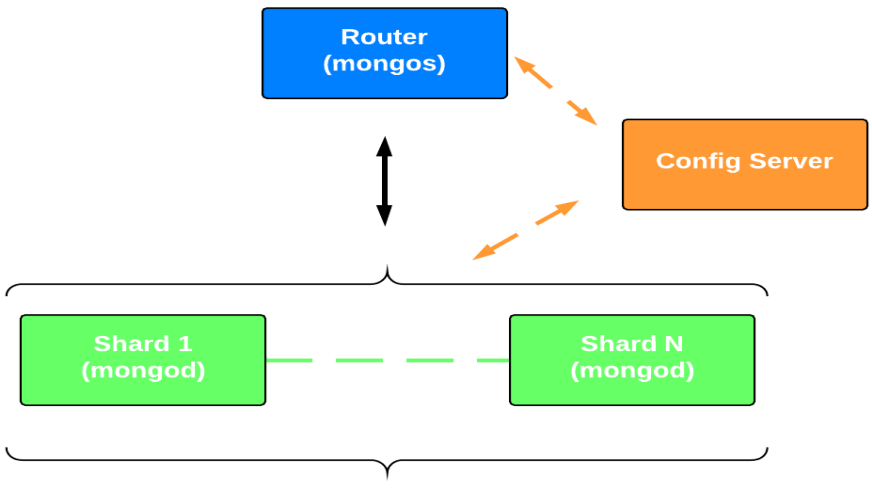


Figure 6: Architecture of MongoDB Sharded Cluster for the Insights System

While MongoDB provides a number of features which make it a de-facto choice for data store on the Insights System, it suffers from high latency when the number of inserts is high as it creates write locks on the collection in each shard. The performance of MongoDB worsens for updates as the collection size gets larger [25].Therefore, MongoDB ,though a good choice for storing and processing impressions data , it does not work for bid requests wherein the number of bid requests received daily is approximately 100 times more than the number impressions served daily. On the contrary, Apache Hive is a powerful data warehousing application built on top of Hadoop, which enables access to data using Hive QL, a language which is very similar to SQL[23][30]. As shown in Figure [5], Hive only requires the bid request files (in json format) to be processed (via map-reduce jobs as shown in Figure [5]) and copied to HDFS (Hadoop Distributed File System). Hive provides a database layer abstraction on this file (see Figure [7]) by allowing the user to create tables and map the columns of the table to each field in the file. Data retrieval/aggregation is done via Hive QL, which in turn invokes map-reduce jobs (see Figure [7]) to process the data, thereby allowing the computation of various attributes from bid requests [30].

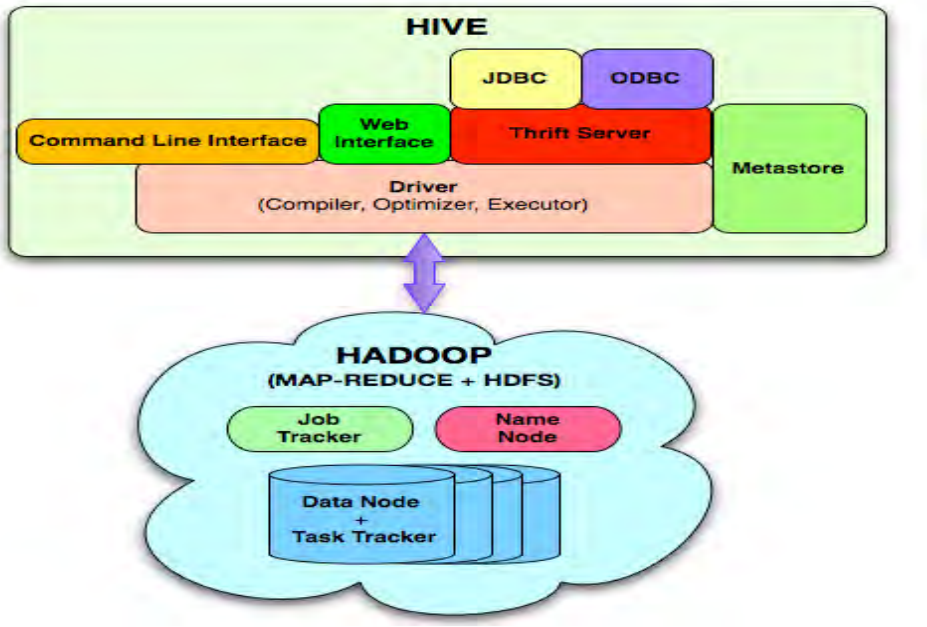


Figure 7: Architecture of Hive

(Source: Hive- A Warehousing Solution Over a Map-Reduce Framework)

2.3.2 Data Processing

The attribute computation task as outlined in Section 2.2.2 has a high computational complexity and therefore requires parallel and distributed applications for the same. Hadoop is an open source framework for writing and running distributed applications that process large amounts of data [14]. Of all the great features of Hadoop that has made it one of the most widely used Big Data technologies, we intend to leverage its *salability* and *simplicity* in the Insights System. Hadoop scales linearly to handle larger data by adding more processing units to the cluster [14]. Moreover, Hadoop provides a very simple abstraction for writing efficient parallel code, while the bookkeeping of splitting/managing input data and the assignment of computation to a processing unit is done under the hood [14].As shown in Figure [10], a Hadoop setup consists of the following components

- ✓ *Name Node*: Hadoop employs a master/slave architecture in which Name Node acts as the master of HDFS (Hadoop Distributed File System) that directs the slave Data Node to perform low level I/O tasks. The Name Node is the bookkeeper of HDFS; it keeps track of how files are broken down into file blocks, which nodes store which blocks and the overall health of the distributed file system. The Name Node is memory and I/O intensive [14].

- ✓ *Data Node:* The Data Node is responsible for reading or writing HDFS blocks to actual files on the local file system. When a user program wants to read or write a HDFS file, the file is broken into blocks and the Name Node tells the user program which Data Node does each block reside in. The user program then directly communicates with the Data Node daemons to process the local files corresponding to the blocks [14].The interaction of the Name Node and Data Node is demonstrated in Figure [10].

- ✓ *Secondary Name Node:* The Secondary Name Node is an assistant daemon for monitoring the state of the cluster HDFS. The SSN differs from the Name Node in that this process doesn't receive or record any realtime changes to HDFS. Instead, it communicates with Name Node to take snapshots of the HDFS metadata at intervals defined by the cluster configuration [14].

- ✓ *Job Tracker:* As shown in Figure [9], the Job Tracker acts as a liaison between the user's program and the Task Tracker. Once a code is submitted to the Hadoop cluster, the Job Tracker determines the execution plan by determining which files to process, assign Task Trackers to different tasks, and monitors all tasks that are running on the Task Trackers. Should a task fail, the Job Tracker will automatically relaunch the task, possibly on a different Task Tracker, upto a predefined limit of retries. There is one Job Tracker daemon per cluster [14].

- ✓ *Task Tracker:* The Task Tracker is responsible for managing the execution of each map-reduce job that the Job Tracker assigns to it, as shown in Figure [9]. The Task Tracker constantly communicates with the Job Tracker. If the Job Tracker fails to receive a heartbeat from a Task Tracker within a specified amount of time, it will assume that the

Task Tracker has crashed and will resubmit the corresponding task to another Task Tracker in the cluster [14].

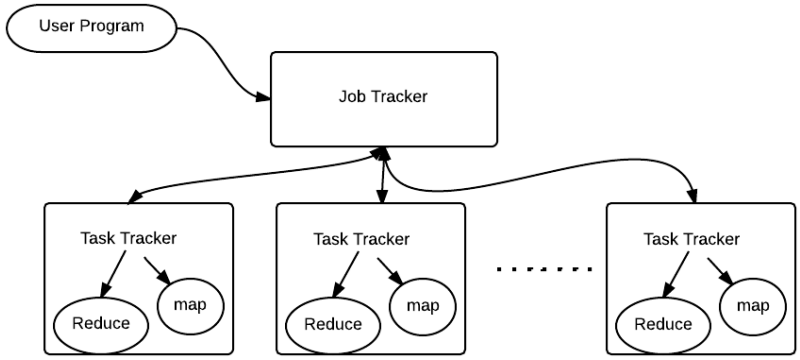


Figure 9: Interaction between Job Tracker and Task Tracker

(Source: Hadoop in Action by Chuck Lam)

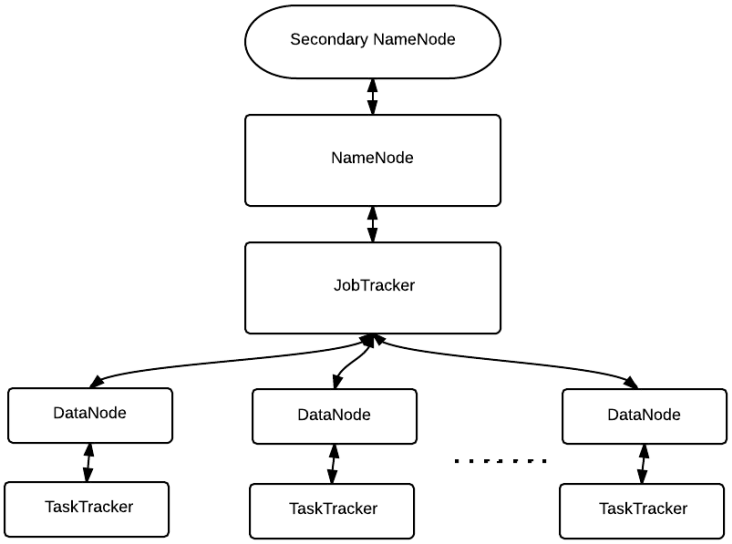


Figure 10: Architecture of Hadoop Cluster for the Insights System

(Source: Hadoop in Action by Chuck Lam)

The structure of a typical map-reduce job is shown in Figure [11] wherein the master is the Job Tracker which assigns map task to each Task Tracker in the cluster. The Task Trackers, then execute the map task on the corresponding input split and generate intermediate files onwhich reduce tasks are run to create the final output file. Since we are using both MongoDB and Hadoop on the Insights System, this allows us to leave the data aggregation (reduction) to MongoDB. Therefore, for the execution of a map-reduce program (data transformation / attribute computation) on the Insights Platform (see Figure [12]), each map task sends the output to be aggregate/stored to the Mongo router, which in turn distributes the data across multiple shards. This structure therefore allows us to further simplify the map-reduce user program, wherein the user only has to write map-jobs and embedd the Mongo insert/update query in the

map job and therefore eliminates the need to write reduce jobs . Moreover, the output of a map-reduce task is stored in MongoDB (and therfore available for further procesing by leveraging on MongoDB's rich querying features) rather than in flat files. Hadoop can be configured to run in *fully distributed mode* wherein the Data Node and Task Trackers run on different machines on a cluster or in *pseudo-distributed mode* wherein all Hadoop components run on a single machine. Since the Insights System is setup on a single server with multiple CPU cores, we are running Hadoop in *pseudo-distributed mode.*

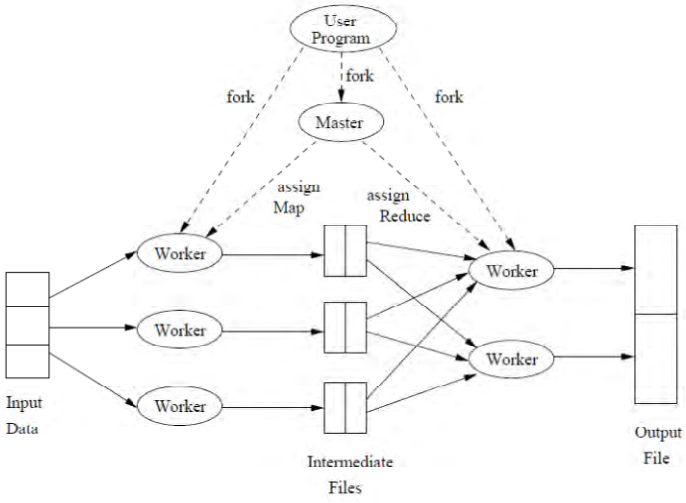


Figure 11: Overview of the execution of a map-reduce program

(Source: Mining of Massive Datasets by Rajaraman,Leskovec and Ullman)

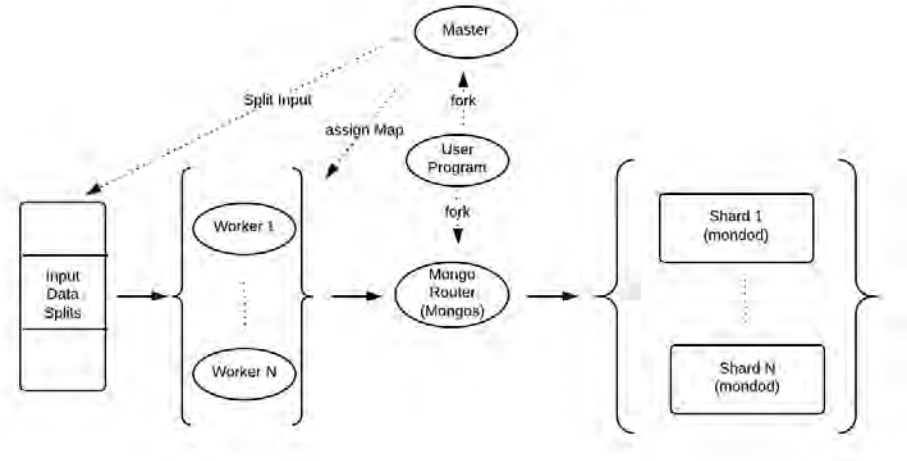


Figure 12: Overview of the execution of a map-reduce program on the Insights System

The data processing and attribute computation classes for the Insights System have been written in Python. The map-reduce jobs that have been written in python are executed on Hadoop by using the *Hadoop Streaming API ,* which is an utility that allows map-reduce jobs to be written in languages other than Java[16].Appendix D gives an outline of all the classes that have been written for the Insights System.

2.3.3 Data Analysis

Apart from being the statistical computing and graphics environment , with a large library of packages that support statistical computing and data visualization, R has a number of machine learning packages like *"glm"* (for Logistic Regression),"*klaR*" (for Naïve Bayes),*"e1071"* (for Support Vector Machines) and *"pracma"* (for ROC analysis) among others which are required for the analysis done in this research [17][18][26].Moreover, R provides packages like *"rmongodb"* and *"Rhive"* for connecting to and retrieving data from Mongo DB and Hive respectively [27][28].

2.3.4 External API's

As shown in Figure [5], the Insights system uses two external API'. The first API is the *GeoLite City* database of MaxMind which enables us to determine the country, region, city, latitude and longitude associated with IP addresses worldwide [31]. The second API is the *Local Weather API* of World Weather Online which returns the current weather for a given latitude/longitude coordinates [32]. Since the Insights System processes the impressions data of the previous day, the weather data has to be stored beforehand. Therefore a list of latitude/longitude coordinates is maintained along with the frequency of impressions from each coordinate which is updated when the Insights System processes impressions. Therefore the weather information is obtained daily from the Local Weather API based on the geographical coordinates sorted by frequency of impressions in descending order.

2.3.5 Interfaces

A set of interfaces have been setup for the Insights System, which allows for ease of access, monitoring and analysis of data. The first interface is a job scheduler called *Citrine*, which allows us to create, schedule and monitor all data processing tasks in Section 2.3.2. Please refer to [34] for details. Similarly, we have setup an interface for MongoDB called *RockMongo* which allows us to access, update and monitor the state of data in the Mongo cluster. Please refer to [35] for details. *Rstudio IDE* servers as a users interface for R on the Insights platform, thereby allowing data analysis to be done easily. Please refer to [36] for details. Moreover, the status of map-reduce jobs running on the Task Trackers can be monitored through [37] and the health of the Name Node and therefore the Hadoop Cluster can be monitored through [38].

3. Dimensionality Reduction & Model Selection

3.1 The Data Mining Process

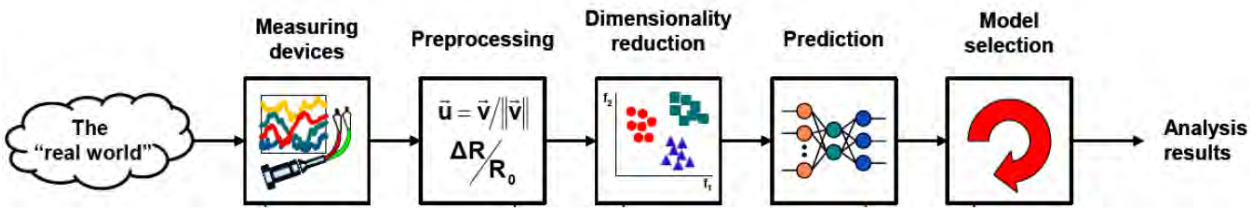


Figure 14: The DataMining Process

A standard Data Mining process consists of steps as shown in Figure [14].We briefly describe each step in this section

- ✓ *Data Collection/Storage :* The data collection/storage step includes extraction and storage of bid requests, impressions and events data as described in section [2.2.1]
- ✓ *Preprocessing:* The preprocessing step includes computation of various attributes as described in section [2.2.2].
- ✓ *Dimensionality Reduction:* In the dimensionality reduction stage, we identify irrelevant and redundant features that can be excluded from the model. The various feature selection methods have been explained in section [3.12]
- ✓ *Prediction:* During the prediction stage of the data mining process, we train different learning models (Naïve Bayes, Logistic Regression, Support Vector Machines) on the training set and then use the model to predict outcomes for a validation set. Training Set and Validation Set have been described in section [3.2]. The various learning models have been explained in section [3.5],[3.6],[3.7]
- ✓ *Model Selection & Analysis :* In the model selection & analysis stage, we evaluate the accuracy of each model by ROC Analysis and compare the performance of models against each other using methods described in section [3.10],[3.11]

3. 2 Notations & Definitions

We define a ***viewer history***, $V_k \in V : \bigcup_k V_k = V, \bigcap_k V_k = \phi, k \in \mathbb{Z}^+$ and '$n_k \in \mathbb{Z}^+$' as the number of impressions shown to viewer V_k. The set of attributes for viewer V_k is denoted by $A_j, \mathrm{j} = 1,..., J, A_j \in A$. Let the ***class label*** $C \in \{0,1\}$ denote ***no-click/click***.Therefore, each ***impression*** for V_k is denoted by $x_i : i = 1,..., n_k, \sum_k n_k = I$ where $x_i = (x_{i1,} x_{i2}, ..., x_{iJ}), \forall i = 1,..., I; j = 1,..., J$ is a vector of dimension $1 \times J$ and represents the attribute values for an impression and is therefore called an ***instance*** of the data set. Therefore, x_{ij} is the ***attribute value*** of attribute A_j for impression i. Moreover c_i denotes the oucome for impression 'i'. Let $X_{I \times J}$ be a $I \times J$ matrix denoting the dataset where $\sum_k n_k = I$ is the number of impressions and J is the number of attributes in the dataset. Let $\aleph = X \times C$ denote the space of ***labeled instances***. Therefore the vector $\aleph_i = (x_{i1,} x_{i2}, ..., x_{iJ}, c_i), \forall i = 1,..., I, x_{.j} \in A_j$ represents the attribute values for impression 'i' and its outcome (***click/no-click***). $\tilde{\aleph} \subset \aleph$ denotes the ***training set*** while $\aleph \setminus \tilde{\aleph}$ represents the ***validation set***. A ***classifier*** $\mathbb{C} : x_i \to c_i$ maps an impression to its outcome. ***Training the classifier*** $\mathbb{C}$ on the training set $\tilde{\aleph}$ implies identifying a scoring function

$$f : \tilde{X} \times \tilde{C} \to \mathbb{R} \tag{1}$$

While ***Cross Validation*** implies hypothesizing the class 'c_i' such

$$\mathbb{C}(x_i) = \underset{c_i}{\operatorname{argmax}}\, f(x_i, c_i) \forall x_i \in X \setminus \tilde{X}, c_i \in C \tag{2}$$

and then comparing the predicted class of x_i with the actual class. In a binary classification problem, the following scenarios can occur

Table [1]: Confusion Matrix for a Binary Classifier

	Actual Class = 1	Actual Class = 0
Predicted Class =1	True Positives	False Positives
Predicted Class =0	False Negatives	True Negatives

When the number of non-clicks (***negative instances***) >> number of clicks (***positive instances***) i.e $Click\ Through\ Rate\ (CTR) = \frac{\sum_{i=1}^{I} c_i}{I} \to 0$, the data set suffers from ***Class Skew.*** In such cases, the classifier $\mathbb{C}$ does not find sufficient number of positive examples to train on and as a result the scoring function f is inaccurate. Therefore the cross validation step results in large number of false negatives while the number true positives are close to zero. On the other hand, in-order to ensure that the classifier finds sufficient number of positive instances to train on without disturbing the class proportions (CTR), the number of impressions, I in the training set and the validation set have to be large which in turn increases the cost of training a classifier. Moreover, if the number of attributes J is large, the classifier further suffers from the ***Curse of Dimensionality.***

3.3 Click Prediction Process

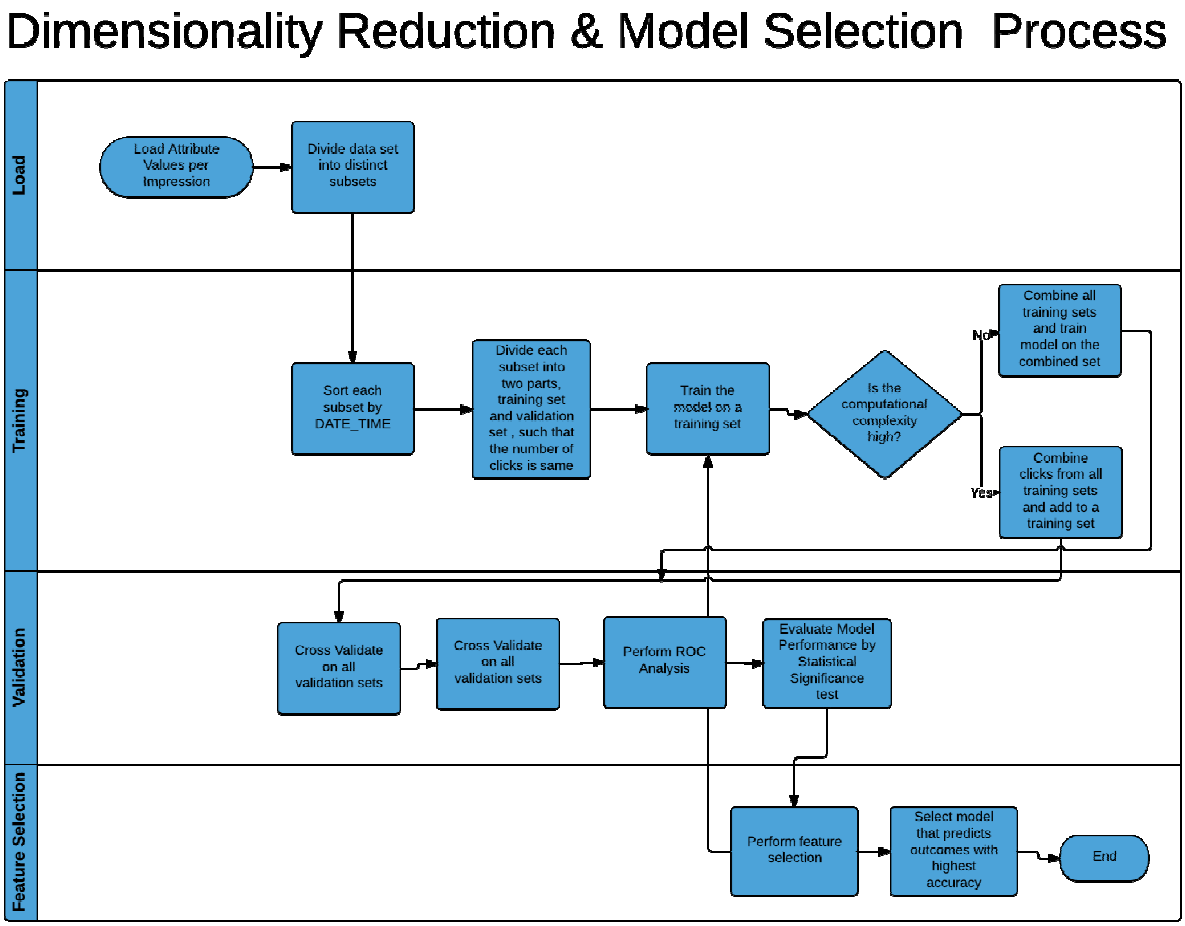
The click prediction process involves creating K distinct random samples from the data set, splitting the impressions ordered by date-time in each of these K samples into a training set and test set such that both the training set and the test set have equal number of clicks. We then train the classifier on each of the K training sets and cross validate on the K validation sets. Finally, we conduct statistical significance tests on the result of the K cross validation steps to hypothesize on the performance of the classifier [44]. A formal representation of the algorithm is given below.

- ✓ **Step 1:** Divide the space of labeled instances $\aleph = X \times C$ into K distinct subsets $\aleph_1 = X_1 \times C_1, \aleph_2 = X_2 \times C_2, ..., \aleph_k = X_k \times C_k$ such that $\bigcup_{k=1}^{K} \aleph_k = \aleph$ and $\bigcap_{k=1}^{K} \aleph_k = \phi$ where each $\aleph_k$ is a $I_k \times (J+1)$ matrix.
- ✓ **Step 2:** Sort $\aleph_k \forall k \in K$ by DATE_TIME
- ✓ **Step 3:** Find l_k such that $\forall \aleph_k \exists \tilde{\aleph}_k \subset \aleph_k : \sum_{i=1}^{l_k} c_i = \frac{\sum_{i=1}^{I_k} c_i}{2}$
- ✓ **Step 4:** Therefore $\tilde{\aleph}_k$ which is a $l_k \times (J+1)$ matrix forms the training set while $\aleph \setminus \tilde{\aleph}_k$ which is a $(I_k - l_k) \times (J+1)$ matrix forms the validation set.
- ✓ **Step 5:** Train classifier $\mathbb{C}$ on $\tilde{\aleph}_k$ and obtain scoring function f as shown in equation [1]
- ✓ **Step 6:** Cross-validate the classifier $\mathbb{C}$ on $\aleph_k \setminus \tilde{\aleph}_k$ and perform ROC analysis.
- ✓ **Step 7:** Construct confidence intervals on the performance of the classifier $\mathbb{C}$

For classifiers like Naïve Bayes where the training costs are low and frequency of clicks in the training set is a parameter of the model we train the model on $\bigcup_k \tilde{\aleph}_k$ and cross validate on $\aleph_k \setminus \tilde{\aleph}_k, \forall k \in K$ .However, for models that incur a high cost of training like Logistic Regression and Support Vector Machines, we select a training set from one of the K distinct training sets and add impressions that resulted in clicks from the rest of the $K-1$ training sets to the selected training set. This results in the classifier finding sufficient number of positive instances in the training set in order to identify a correct scoring function. A formal representation of the algorithm is given next.

- ✓ **Step 1:** Choose an $\tilde{\aleph}_k$ and add $\tilde{\aleph}_{mi} : c_i = 1, m \neq k \forall m \in K$
- ✓ **Step 2:** Train classifier $\mathbb{C}$ on the training set created in step 1
- ✓ **Step 3:** Cross-validate the classifier $\mathbb{C}$ on $\aleph_k \setminus \tilde{\aleph}_k$ and perform ROC analysis
- ✓ **Step 4:** Construct confidence intervals on the performance of the classifier $\mathbb{C}$

3.4 DRMS Flow Diagram



3. 5 Naïve Bayes Algorithm

The Naïve Bayes algorithm estimates the probability that an impression will result in a click, by assuming independence of the attributes conditioned on the outcome (click/no-click). Therefore the conditional probability of the attributes along with the prior is estimated from the training set, which is then used to estimate the posterior (click probability) for an impression in the validation set.

3. 5.1 The Model

Adscience uses a Naïve Bayes algorithm to compute the click probability of an impression.
Naïve Bayes can be defined as

$$p(c_i = 1 \mid x_i) = \frac{1}{Z} p(c_i = 1) \prod_{j=1}^{J} p(x_{ij} \mid c_i = 1) \tag{3}$$

$$Z = p(c_i = 0) \prod_{j=1}^{J} p(x_{ij} \mid c_i = 0) + p(c_i = 1) \prod_{j=1}^{J} p(x_{ij} \mid c_i = 1) \tag{4}$$

Where $p(c_i = 1 \mid x_i) : x_i \in X \setminus \tilde{X}$ is the likelihood of a click, given an impression 'i' having attributes $x_i = (x_{i1}, x_{i2}, ..., x_{iJ}), \forall i = 1, ..., I; j = 1, ..., J$ [2]. The prior $p(c_i = 1)$, which is estimated from the training set $\tilde{\aleph}_k$, is defined as [5],

$$p(c_i = 1) = eCTR = \frac{\#\text{Clicks}}{\#\text{Impressions}}; p(c_i = 0) = 1 - p(c_i = 1) \tag{5}$$

Furthermore, the *Weak Law of Large Numbers* implies

$$\lim_{I \to \infty} \mathrm{P}\left(\left| \frac{\sum_{i=1}^{I} c_i}{\sum_{i=1}^{I} 1} - E\left[\frac{\sum_{i=1}^{I} c_i}{\sum_{i=1}^{I} 1} \right] \right| > \varepsilon \right) = 0 \forall \varepsilon > 0, c_i \in \{0,1\} \tag{6}$$

Which suggests that as the number of impressions in the training set $\tilde{\aleph}_k$, gets large, the CTR i.e $p(c_i)$ converges in probability to the expected click through rate i.e eCTR.Moreover, the ***likelihood***, $p(x_{ij} \mid c_i)$ is defined as the conditional probability that impression 'i' will have attribute value x_{ij} given that it's a click or a non-click.

3.5.2 Estimation of Likelihood

The ***likelihood***, as defined in section [3.5.1] can be estimated in two ways. The first method as demonstrated in equation [7] computes the proportion of occurrence of an attribute value for a given event in the training set.

$$p(x_{ij} \mid c_i) = \frac{\sum_{i=1}^{I} 1_{x_{ij} \cap c_i}}{\sum_{i=1}^{I} 1_{c_i}} \forall x_{ij} \in A_j, c_i \in \{0,1\} \tag{7}$$

The second method, as shown in equation [8] is the ***kernel density estimation*** of attribute A_j wherein $h > 0$ is a smoothing parameter called bandwidth and $K(.)$ is the kernel, a symmetric but not necessarily positive function that integrates to one. The kernel function $K(.)$ determines the shape while the bandwidth h determines the width of $\hat{f}$ [44].

$$\hat{f}_h(A_j) = \frac{1}{Ih} \sum_{1}^{I} K\left(\frac{x - x_{ij}}{h} \right) : x_{ij} \in \tilde{\aleph}_k, A_j \in A \tag{8}$$

3.5.3 Estimation of Click Probability for New Attribute Values

The computation of the posterior probability $p(c_i = 1 | x_i) \forall x_i \in X \setminus \tilde{X}$ requires the computation of $p(x_{ij} | c_i) \forall x_{ij} \in \tilde{X}$. A Naïve Bayes Classifier when trained on the training set $\tilde{\aleph}_k$ results in J matrices each of dimensions $|C| \times \left| \bigcap_i x_{ij} \right|$ wherein each element represents the probability $p(x_{ij} | c_i)$. Therefore, in cases where there can be many possible values for an attribute A_j, there can be cases where in $x_i \in X \setminus \tilde{X}$ but $x_i \notin \tilde{X}$ and as a result, the classifier does not return the corresponding $p(x_{ij} | c_i) \forall x_{ij} \in \tilde{X}$ .Thus, for the computation of the posterior probability, the following algorithm has been proposed

- ✓ **Step 1:** $\forall x_{ij} \in X \setminus \tilde{X}$, check whether $x_{ij} \in |C| \times \left| \bigcap_i x_{ij} \right|$. If TRUE , then Goto Step 2 else Goto Step 3
- ✓ **Step 2 :** Return $p(x_{ij} | c_i)$; STOP
- ✓ **Step 3 :** Return $\frac{1}{\left| \bigcap_i x_{ij} \right|}$; STOP

3. 6 Logistic Regression

The Logistic Regression model estimates the click probability of an impression by first finding a separating hyper plane between the clicks and non-clicks and then by mapping the hyper plane to the interval (0, 1) by a logit function.

3. 6.1 The Model

The Logistic Regression model is a binary classifier $\mathbb{C}$ that identifies a scoring function f by

- ✓ Formulating a linear model from the training set, $\tilde{\aleph}$ and
- ✓ Map the linear model to the interval $(0,1)$ by a *logit* function which in turn becomes the probability of a positive outcome (click).

Let us denote by $p = P(c_i = 1)$, the probability that an impression results in a click out of I impressions. Therefore we define a Bernoulli random variable $M_i = \begin{cases} 1 \text{ if } c_i = 1 \\ 0 \text{ o.w} \end{cases}$. Therefore, the random variable M denotes the number of clicks in I impressions and has a binomial distribution as shown in equation [9].

$$M = \sum_{i=1}^{I} M_i \sim Bin(I, p) \qquad (9)$$

Now, we formulate the linear model as shown in equation [10] wherein $\tilde{X}$ is a $I \times (J+1)$ matrix of I impressions, J attributes and a vector of 1's. Moreover, the weights $\beta = (\beta_0, \beta_2, ..., \beta_J)$ is a $(J+1) \times 1$ vector where β_0 represents the intercept. The model results in a $I \times 1$ vector η.

$$\eta = \tilde{X}\beta \tag{10}$$

We map $\eta \in \mathbb{R}^+$ to $p \in (0,1)$ by the logit function given in equation [11]

$$\eta = g(p) = \log\left[\frac{p}{(1-p)}\right] \tag{11}$$

Therefore, the computational problem is estimation of the weights $\hat{\beta}$ from the training set $\tilde{\aleph}$ which is obtained by maximizing the log-likelihood of the canonical distribution function of M with respect to β .This is done by the ***Fisher Scoring Method.*** An alternate method for obtaining $\hat{\beta}$ is by the ***Batch Gradient Descent Method*** though; the Fisher scoring method is faster. We do not elaborate on these methods as they are built into most data mining packages [19] [20].

3. 6.2 Estimation of Click Probability

The click probability for an impression x_i is estimated from equation [12] where $x_i \in X \setminus \tilde{X}$ and $\hat{\beta}$ is estimated using the methods described in section [3.6.2.1].

$$p = p(c_i = 1 \mid x_i; \hat{\beta}) = \frac{1}{1 + e^{-x_i^T \hat{\beta}}} \tag{12}$$

3. 7 Support Vector Machine

A support vector machine is a binary classifier that builds a model by finding a separating hyper plane with maximal margins, that separates clicks from non clicks and then uses the model to classify whether an impression will result in a click or not.

3. 7.1 The Model

Support Vector Machine is a binary classifier $\mathbb{C}$ that identifies a scoring function f by

- ✓ Mapping an instance $x_i \in \tilde{\aleph}$ into a higher (maybe infinite) dimensional space [50]
- ✓ Find a separating hyper plane with the maximal margin in the higher dimensional space [50]

This follows from redefining the classification problem as the problem of finding a hyper plane (as shown in equation [13]) with suitable weights, w such that the hyper plane can separate instances of the two classes [44]. Therefore an instance x_i can be mapped to a target class depending on which side of the hyper plane it lies on.

$$(w^T x_i + w_0) \geq 0 \text{ for } c_i = 1$$
$$(w^T x_i + w_0) \leq 0 \text{ for } c_i = 0 \quad (13)$$

Separability of instance into two classes results from the following theorem

Theorem 3.1 (Cover's Theorem): **A complex pattern-classification problem, cast in a high-dimensional space nonlinearly, is more likely to be linearly separable than in a low dimensional space, provided that the space is not densely populated.**

Support Vector Machines, further improvise on this concept by finding a hyper plane that has the largest distance to the nearby training instance of any class (the ***margin***), since larger the margin, the better the classifier will generalize to unseen data. Therefore, we rewrite equation [13] as follows to incorporate margins into the classification problem

$$(w^T x_i + w_0) \geq +1 \text{ for } c_i = 1$$
$$(w^T x_i + w_0) \leq -1 \text{ for } c_i = 0 \quad (14)$$

Therefore, if we define $y_i = \begin{cases} -1 \; if \; c_i = 0 \\ 1 \; if \; c_i = 1 \end{cases}$, equation [14] can be re-written as

$$y_i (w^T x_i + w_0) \geq +1 \quad (15)$$

The ***distance from a separating hyper plane*** is defined as

$$\frac{| w^T x^t + w_0 |}{|| w ||} \quad (16)$$

Using equation [14], equation [16] can be rewritten as $\frac{y_i (w^T x_i + w_0)}{|| w ||}$. However, since the click prediction problem suffers from class-skew we define a ***soft-margin hyper plane*** by re-writing equation (15) as

$$y_i (w^T x_i + w_0) \geq 1 - \xi_i, \xi_i > 0 \quad (17)$$

Where $\xi_i > 0$ is a slack variable which stores the deviation from the margin. A deviation from the margin can be of two types (a) x_i lies on the wrong side of the hyper plane and is therefore misclassified or (b) x_i lies on the right side of the hyper plane but within the margin. If training an SVM on $\tilde{\aleph}$, results in

$\xi_i = 0$, then x_i has been correctly classified, if $0 < \xi_i < 1$, x_i has been misclassified while $\xi_i \geq 1$ implies that x_i has been misclassified. Moreover, we introduce a penalty parameter, $C > 0$ which penalizes $x_i : \xi_i > 0$ thereby leading to lower generalization error. The instances $x_i : \xi_i > 0$ are called ***support vectors*** and the number of support vectors is an upper bound estimate for the expected number of errors [44]. Finally the instance x_i is mapped to a higher dimensional space by a function ϕ. Equation [18] represents the optimization problem solved by support vector machines where w is the normal vector to the hyper plane. A number of popular methods like ***Sequential Minimization Optimization*** exist for solving equation [18]. We will however not elaborate on these methods since they are readily available with the LIBSVM library and other data mining packages. The time complexity of an SVM is $O(I^3)$ [50].

$$\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{I} \xi_i$$

$$subject\ to\ y_i (w^T \phi(x_i) + b) \geq 1 - \xi_i ; \xi_i \geq 0 \qquad (18)$$

3. 7.2 RBF Kernel

Equation [18], when represented in its dual form, requires the computation of $\phi(x_i)^T \phi(x_j)$. However, the inner product $\phi(x_i)^T \phi(x_j)$ can be replaced by a ***kernel function*** $K(x_i, x_j)$. Therefore, instead of having to map two instances x_i and x_j to a higher dimensional space and then having to do a dot product, the kernel function can be directly applied in the original space. The substitution of $\phi(x_i)^T \phi(x_j)$ with $K(x_i, x_j)$ is called ***kernel trick***. The symmetric and positive semi definite $I \times I$ matrix of kernel values is called a ***Gram matrix*** [44].

The radial basis function (RBF) kernel, given in equation [19] is one of the most widely used kernels that non-linearly maps instances into a higher dimensional space, so that, unlike the linear kernel, it can handle the case when the relation between class labels and attributes is non linear. An important property of the RBF kernel is $0 < K_{ij} \leq 1$ [50].

$$K(x_i, x_j) = \exp\left(-\gamma \left\| x_i - x_j \right\|^2\right), \gamma > 0 \qquad (19)$$

3. 7.3 Parameter Estimation

Support Vector Machines require the estimation of the penalty parameter C and the kernel parameter γ. Since these parameters are not known beforehand, the parameters need to be estimated from the training set, $\tilde{\aleph}$. The goal is to identify (C, γ) such that the classifier can predict class labels from the validation set $\aleph \setminus \tilde{\aleph}$ with minimum misclassification errors. A number of techniques have been suggested in literature to

estimate the parameters. One such technique is combining a ***grid-search method*** with cross validation which can be easily parallelized [50]. However, this technique is only feasible when the number of instances in the data set is in order of magnitude of thousands [50]. For very large data sets, we have to rely on heuristic techniques. One such method is to consider exponentially growing sequences of C and γ , for example $C = 2^{-5}, 2^{-3}, ..., 2^{15}$; $\gamma = 2^{-15}, 2^{-13}, ..., 2^{3}$ [50]. Therefore, the heuristic method identifies a starting position and then identifies the direction of the next sequence by cross validating with values on either side of the chosen parameter values. This enables us to specify a lower bound for the parameter. Since the click prediction problem suffers from class skew, we need to have a high penalty for misclassification. Therefore the lower bound on C can be given as $C > 1$. It is important to note that for very large value of C, the training time will also go up accordingly. Therefore, starting with $C = 2$, C can be increased exponentially till the misclassification errors on the validation set $\aleph \setminus \tilde{\aleph}$ have fallen to a desired level. The starting value of γ is generally chosen to be $\frac{1}{J}$ where J is number of features in the training set [18].

3.8 Dimensionality Reduction for Logistic Regression and SVMs

Logistic Regression and SVMs require each impression, x_i to be represented as a vector of real numbers. Attributes that are defined on the nominal scale, have to be converted to numeric data. This requires encoding the attribute values into a $1 \times \left| \bigcap_i x_{ij} \right|$ binary vector, where for an impression the attribute is represented a vector Y of dimension $1 \times \left| \bigcap_i x_{ij} \right|$ where

$$Y_k = \begin{cases} 1 \; if \; x_{ij} \in x_i \\ 0 \; o.w. \end{cases}, k \in \left| \bigcap_i x_{ij} \right| \tag{20}$$

The number of attribute values for attributes like DOMAIN, IP, COUNTRY and BANNER can be very large and as a result the binary vector Y would be large as well. Since training a Logistic Regression and SVMs require estimation of weights for each of the J features, increasing the size of J by introducing large Y requires the models to estimate a larger weight vector. As a result, the number of iterations required by the estimation method increases as well, thereby increasing the computational complexity of the estimation method. Therefore, to prevent the model from suffering from curse of dimensionality, such attributes need to be grouped into categories. However, a number of attributes like IP and DOMAIN cannot be easily categorized. We present the following algorithm in such cases which uses Naïve Bayes' posterior probabilities to group attribute values.

- ✓ **Step 1:** Compute the priors and the likelihood for $\bigcup_k \tilde{\aleph}_k$ using equations [5][7]and [8]
- ✓ **Step 2:** Estimate the click probability for attribute 'j' as

$$p(c_l = 1 \mid a_l) = \frac{1}{Z} p(c_l = 1) p(a_l \mid c_l = 1) \forall a_l \in \cap_i x_{ij}, l \in \left| \cap_i x_{ij} \right|, x_{ij} \in \bigcup_k \tilde{\aleph}_k \quad (21)$$

- ✓ **Step 3:** Divide $p(c_l)$ into 'm' groups and map $x_{ij} \in \bigcup_k \tilde{\aleph}_k$ to each of these groups

It is important to note that the choice of m will determine how well the target algorithm (logistic regression or support vector machine) performs. Therefore, search heuristic methods combined with cross validation as outlined in section [3.7.3] can be used to estimate m.

3.9 Transformation of Features

For attributes defined on the ordinal or interval scale, large numeric ranges cause numerical difficulties in the algorithms used to estimate the weights in logistic regression and support vector machines. Moreover attributes with large numeric ranges tend to dominate attributes in smaller numeric ranges and therefore would lead to incorrect feature selection. It is important to note that both training and testing data have to be scaled using the same method [50]. A number of packages for SVM scale the numeric attribute values to zero mean and unit variance. However, for the logistic regression model, the input data has to be scaled and then given to the algorithm to train or predict. A number of transformations like the ***Box-Cox Transform*** and the ***Inverse Hyperbolic Sine Transform*** have been proposed. We present a simplified version of the Box-Cox transform in equation [22] that scales numeric attributes for which the bounds are unknown, to the interval $[0,1]$.

$$g(x_{.j}) = \log(x_{.j} + 1) \quad (22)$$

On the other hand, non-negative attributes for which the upper bounds are known, we scale the data to the interval $[0,1]$ by computing the ratio of the attribute value to the upper bound.

3.10 Receiver Operating Characteristics (ROC) Analysis

A receiver operating characteristics graph (ROC) is a *'technique for visualizing, organizing and selecting classifiers based on their performance'* [13]. In a ROC graph , the true positive rate $\left(tp\ rate \approx \frac{True\ Positives}{Total\ Positives} \right)$ is plotted against the false positive rate $\left(fp\ rate \approx \frac{False\ Positives}{Total\ \text{Negatives}} \right)$ .We can only compute the true positive rate and the false positive rate in case of discrete classifiers for which the output is a class label.

As defined in section [3.2], the cross-validation step results in a class $c \in \{0,1\}$ such that $\mathbb{C}(x_i) = \underset{c}{\text{argmax}}\, f(x_i, c_i) \forall x_i \in X \setminus \tilde{X}, c_i \in C$ where in the scoring function $f : \tilde{X} \times \tilde{C} \to \mathbb{R}$ is the probability that an impression will result in a click. However, as the CTR, $\frac{\sum_{i=1}^{I} c_i}{I} \to 0$, the estimated click probability $f(x_i, c_i = 1)$ is small while the probability of not clicking an impression, $f(x_i, c_i = 0) = 1 - f(x_i, c_i = 1)$ is large. This results in the classifier, misclassifying clicks as no-clicks

and as a result $fn >> tp$ .Therefore , instead of classifying an impression in the validation set as click/no-click , we rank the impressions by their estimated click probability i.e $R: f \to \mathbb{Z}^+$ such that $R_i > R_j \forall f(x_i, c=1) > f(x_j, c=1)$ .This implies that the impressions with the highest rank have the highest probability (≈ 1) of being clicked, while the impression with the lowest rank have the lowest probability (≈ 0) of being clicked. Let $c_i \in C \setminus \tilde{C}$ be the actual click for impression $'i'$. In case of perfect estimation of click probabilities, we can partition R into $R^{'}$ and $R \setminus R^{'}$ such that $R_i < R_j \forall R_i \in R^{'}$ and $R_j \in R \setminus R^{'}$ and $c_i = 0$, $c_j = 1 \forall i \in R^{'}, j \in R \setminus R^{'}$. However, such a scenario only happens when the model ***over fits*** the data. The cross validation step will therefore result in cases there $\exists i \in R^{'} : c_i = 1$ (false negative) and $\exists j \in R \setminus R^{'} : c_j = 0$ (false positive).It is therefore important to note that number of false positives will depend on the partitioning of R as decreasing the number of false positives will increase the number of false negatives and vice versa. This leads to the optimization problem of finding the optimal partition of R such that misclassification costs are minimized with different costs assigned to misclassification of click as no-click and misclassification of no-click as click. However, this is out of scope of this research and is a topic that should be investigated in future work. To construct the ROC, we take the running sum of the actual clicks ordered by the rank of the impressions i.e

$$\sum_{i=1}^{k} c_i, k = 1,..,I : R_i \leq R_{i+1} \tag{23}$$

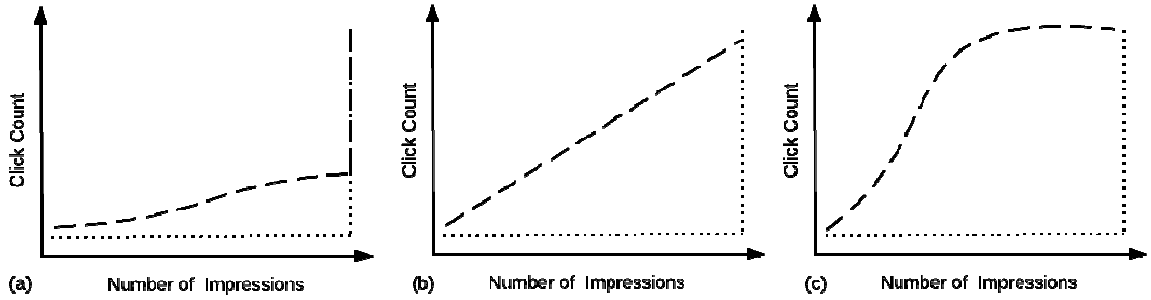


Figure 15: ROC Graphs. The dotted lines represent the estimated click probabilities sorted in ascending order. The dashed lines represent the actual outcomes sorted by the click probabilities.

Figure [15] shows three different schematics of ROC where the dotted line represents the estimated click probabilities for each impression while the dashed line represents the running sum of the clicks ordered by the rank of each impression. In a perfect prediction or in cases of over fitting, the dashed line will super impose the dotted line. In (a), the ROC shows that there exists some false negatives wherein a click probability, $f(x_j, c=1) \approx 0$ was estimated even for a click. However there exist a substantial proportion of clicks which have been predicted correctly. This is represented by the part of the curve where the dotted line and the dashed line overlap. On the other hand in (b) the classifier performs worse than (a) as

the false negatives are large i.e the scoring function has estimated a click probability, $f(x_j, c=1) \approx 0$ for all clicks. In (c), the classifier performs even worse wherein an even lower click probability is estimated for each click as is evident from the steep rise in click count for impression with low rank.

We now turn our attention to the ***Area under an ROC Curve (AUC)*** which is defined as '*probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance*' and can be used as a measure of classifier performance [13]. From Figure [15], the total area of the ROC space can be defined as the area of the rectangle, where the length is the number of impressions and breadth in the total number of clicks in the validation set. The ROC curve occupies a certain proportion of this area and therefore the area occupied by the ROC curve can be expressed as a percentage of the total area. Now, as described in the properties of the ROC curve, lesser the proportion of area occupied by the ROC curve, lower is the number of misclassification by the algorithm and as a result better the performance. Therefore, the proportion of space occupied by the ROC curve in Figure [15.a] is lesser than the proportion of space occupied in Figure [15.b] and Figure [15.c] and therefore represents better performance. However, standard formulations of ROC curves and AUC suggest that larger the AUC, better is the performance of the model. Therefore, in order to standardize our analysis, we take the complement of the proportion of space occupied by the ROC curve.

Formally, The AUC is estimated as

$$AUC = 1 - \frac{\int_0^I \sum_{i=1}^k c_i}{I \times \sum_{i=1}^I c_i}, k = 1,..,I : R_i \leq R_{i+1} \tag{24}$$

The definite integral $\int_0^I \sum_{i=1}^k c_i$ is approximated using the trapezoidal rule [42]. Moreover, as described in section [3.3], when there are K distinct folds on which we train and validate the model, we will have K ROC curves.

3.11 Estimation of Model Performance

For training set $\tilde{X}$ and validation set $X \setminus \tilde{X}$, let us denote by p, the probability that the classifier classifies a click correctly $\mathbb{C}(x_i) = 1 \forall x_i \in X \setminus \tilde{X}, c_i = 1$. Therefore we define a Bernoulli random variable $M_i = \begin{cases} 1 \text{ if } \mathbb{C}(x_i) = 1 \forall x_i \in X \setminus \tilde{X}, c_i = 1 \\ 0 \text{ o.w} \end{cases}$. We define $N = \sum_i c_i$ as the total number of clicks in the validation set. Therefore, $M = \sum_{i=1}^N M_i \sim Bin(N, p)$ and we test whether p is less than or equal to a predefined probability p_0 .We test the following hypothesis

$$H_0 : p \le p_0 \text{ vs } H_1 : p > p_0 \tag{25}$$

Thus the binomial test rejects the null hypothesis if $P(M \ge k) = \sum_{m=k}^{N} \binom{N}{m} {p_0}^m (1-p_0)^{N-m} < \alpha$ where α is the significance level.

Since, M is the sum of independent random variables from the same distribution, by the *Central Limit Theorem,* for large N, $\frac{M}{N}$ is approximately normal with mean p_0 and variance $p_0(1-p_0)$. Then

$\frac{\frac{M}{N} - p_0}{\sqrt{p_0(1-p_0)}} \sim \mathrm{N}(0,1)$. Furthermore, if we train $\mathbb{C}$ on $\tilde{X}$ and validate on $X \setminus \tilde{X}$, K times then we obtain the probability of correctly classifying clicks from each of the K validation sets i.e $p_i, i = 1,...,K$ where $p_i = \frac{\sum_{i=1}^{N} M_i}{N}$ and $\bar{p} = \frac{\sum_{i=1}^{K} p_i}{K}; S^2 = \frac{\sum_{i=1}^{K} (p_i - \bar{p})^2}{K-1}$. Therefore,

$$\frac{\sqrt{K}(\bar{p} - p_0)}{S} \sim t_{k-1} \tag{26}$$

We reject the alternate hypothesis that the classification algorithm can predict clicks with probability more than p_0 when [26] is less than $t_{\alpha,K-1}$ [44].

Similarly, we can compare the performance of two classifiers, validated on K validation sets, using a paired t-test [44]. We define $p_i = p_i^1 - p_i^2$ as the difference in the proportion of clicks classified by two classifiers. Therefore $\bar{p} = \frac{\sum_{i=1}^{K} p_i}{K}; S^2 = \frac{\sum_{i=1}^{K} (p_i - \bar{p})^2}{K-1}$ and we test the null hypothesis $H_0 : \mu = 0$ that the distribution of the difference of proportion of correct classifications across two classifiers has mean 0. We reject the null hypothesis if $\frac{\sqrt{K}\,\bar{p}}{S}$ lies outside the interval $(-t_{\alpha/2,K-1}, t_{\alpha/2,K-1})$.

3.12 Feature Selection

The feature selection problem is defined as '*the process of selecting a subset* $\tilde{A} \subset A$ *of original features according to certain criteria'* [7] [9]. Therefore, the feature selection process results in a $\aleph = I \times (K+1) : K \subset J$ where J is the number of features in the original feature space. On the other hand feature extraction is defined as '*the process of constructing a new set of K dimensional features space out of the original J features*'. Some of the widely used feature extraction techniques are ***Principal Component Analysis*** (PCA) and ***Linear Discriminant Analysis*** (LDA) [44].However feature extraction is out of scope for this research and we only focus on the problem of feature selection.

Feature selection algorithms are often used as a dimensionality reduction technique. The central assumption when using a feature selection algorithm is that the training data contains many ***redundant*** and ***irrelevant features*** [9].

A *redundant feature* is defined as

$$\exists A_j \subset A \setminus \tilde{A} \forall \tilde{A} \subset A : P(C=1 \mid \tilde{A}) = P(C=1 \mid (A_j \cup \tilde{A})) \tag{27}$$

An *irrelevant* feature is defined as

$$\begin{aligned} &P(C=1 \mid \tilde{A}) = P(C=1 \mid (A_j \cup \tilde{A})) : A_j \subset A \setminus \tilde{A} \\ &\exists A_j \subseteq A \setminus \hat{A} : P(C=1 \mid \hat{A}) \neq P(C=1 \mid (A_j \cup \hat{A})) \end{aligned} \tag{28}$$

Irrelevant features are those that provide no useful information in any context whereas *Redundant features* are those which provide no more information than the currently selected features [7].

A feature space of J features can have 2^J possible subsets of feature J but for large J, training and validating a classifier on each of the 2^J possible combinations wherein the number of instances in both the training and validation sets is large, is computationally intractable. We therefore resort to heuristic methods to get a reasonable (but not optimal) solution in reasonable (polynomial) time [44].

3.12.1 Feature Selection Methods

3. 12.1.1 Wrapper Methods

Wrappers utilize the learning model of interest as a black box to score subset of variables according to their predictive power [45]. Two commonly used wrapper techniques are

3. 12.1.1.1 Forward Selection

In forward selection, we start with no features and add them one by one, at each step adding the feature that increases the AUC the most, until any further addition does not increase the AUC [44]. Therefore starting with $F = \phi$ we iterate over the following steps

- ✓ **Step 1:** Find j such that $\arg\max_j AUC(F \cup A_j)$
- ✓ **Step 2:** Add A_j to F if $AUC(F \cup A_j) > AUC(F)$

As is evident, the computational complexity of this algorithm is $O(J^2)$ and again for large J the computation time increases quadratically. For the Naïve Bayes model, the prior [equation 5] probabilities and likelihood [equation 8] need not be re-estimated from the training set for each step of the forward selection algorithm as addition or removal of feature only involves multiplying or not multiplying the conditional feature probability in computation of the posterior. This saves a considerable amount of computational overhead when the prediction model is Naïve Bayes. However, for Logistic Regression and Support Vector Machines, each addition of feature requires the model to be re-trained on the training

set as well. Furthermore, forward selection is a local search procedure and does not guarantee finding the optimal subset, namely the minimal subset that maximizes the AUC [44].

3. 12.1.1.2 Backward Selection

In backward selection, we start with all features and remove them one by one, at each step removing the one that increases the AUC the most until any further removal does not increase the AUC any further [44].Therefore, starting with the full feature set $F = A$, we iterate over the following steps

- ✓ **Step 1:** Find j such that $\arg\max_j AUC(F - A_j)$
- ✓ **Step 2:** Remove A_j from F if $AUC(F - A_j) > AUC(F)$

The computational complexity computational overhead of training the model at each backward selection step is same as that of forward selection.

It is important to note that in step 1 of both forward and backward selection the feature selection algorithm tries to identify whether a feature is irrelevant [equation 36].However irrelevant features can be efficiently identified and excluded in bulk from the feature set which are described in the next section.

3. 12.1.2 Filters Methods

Filters select subset of variables as a pre-processing step, independently of the chosen predictor and are faster when compared to wrappers [45]. The commonly used filter techniques measure the association among features using an association matrix. The ***Measure of Association Method*** evaluates subsets of features on the basis of the following hypothesis "*Good Feature subsets contain features highly associated with the classification , yet unassociated with each other* ".This approach is useful in identifying and removing ***multicollinearity*** in generalized regression models and for reinforcing the ***independence assumption*** of Naïve Bayes. Generally the association between two features is measured by computing the ***correlation*** between them. However, it is only possible to measure correlation between two features if they are on the ordinal, interval or ratio scale. For features measured on the nominal scale we compute the measure of association using ***Cramer's V***.

3. 12.1.2.1 Cramer's V

Cramer's V measures the strength of association between the features and the outcome (click/no-click) and the association among features for training set $\tilde{\aleph}_k$ [47]. Cramer's V for feature v/s outcome results in a $\left|\cap_i x_{ij}\right| \times |C|$ contingency table for each attribute A_j .Where $\left|\cap_i x_{ij}\right|$ is the number of attribute values for attribute A_j .Since $C \in \{0,1\}, |C| = 2$ .Therefore; the contingency table has the following general structure.

Table [2]: Contingency Table for attribute Aj and Clicks

		Outcome		
		C=0	C=1	Total
Attribute A_j	1	n_{10}	n_{11}	$n_{1.}$
	2	n_{20}	n_{21}	$n_{2.}$
	…	…	…	…

	r=$\left\|\bigcap_i x_{ij}\right\|$	n_{r0}	n_{r1}	$n_{r.}$
Total		$n_{.0}$	$n_{.1}$	$n_{..}$

The Crammer's V given in equation [37] produces a value in the range $[-1,1]$ where 0 is no association, +1 is strong positive association and -1 is strong negative association [47]. Moreover $N = n_{..}$ and χ^2 is computed from the contingency table shown in Table [2].

$$V = \sqrt{\frac{1}{N}\frac{\chi^2}{\min\{(r-1),(c-1)\}}} = \sqrt{\frac{1}{N}\frac{\chi^2}{\min\{(\left|\bigcap_i x_{ij}\right|-1),1\}}} \tag{29}$$

On the other hand, in order to measure the association between two features, we need to construct a $\left|\bigcap_i x_{ij}\right| \times \left|\bigcap_i x_{ik}\right| \forall i \neq k$ contingency table as shown in Table [3]

Table [3]: Contingency Table for attribute A_j and A_k

		Attribute A_k				
		1	2	…	c=$\left\|\bigcap_i x_{ik}\right\|$	Total
Attribute A_j	1	n_{11}	n_{12}		n_{1c}	$n_{1.}$
	2	n_{21}	n_{22}		n_{2c}	$n_{2.}$
	…	…	…		…	…
	r=$\left\|\bigcap_i x_{ij}\right\|$	n_{r1}	n_{r2}		n_{rc}	$n_{r.}$
Total		$n_{.1}$	$n_{.2}$		$n_{.c}$	$n_{..}$

The computation of V is then done using the same formula as in ().Moreover, we compute the Cramer's V for each of the K training sets, $\tilde{\aleph}_k$ and test its statistical significance for the population. We therefore define the following hypothesis

$$H_0 : \textit{there exists no association b/w } A_j \textit{ and } A_l, \forall j \neq l$$

$$H_1 : \textit{there exists an association b/w } A_j \textit{ and } A_l, \forall j \neq l \tag{30}$$

The statistical significance test for Cramer's V is same as that of chi-square test of independence i.e $V \sim \chi^2_{(r-1)(c-1),1-\alpha}$ [46]. Therefore we reject the null hypothesis H_0 if the p-value $P(>\chi^2)$ is less than

the significance level α / K where α / K is the ***Bonferroni correction*** for significance testing with K samples [44].

However, it is important to note that for large r, m (for attributes like Domain and IP address) the computation of Cramers'V becomes computationally intractable. In such cases, a better approach would be to group the attribute into values using the techniques described in Section [3.8], rank them in ascending order of the estimated click probabilities, produced by training a Naïve Bayes algorithm only on the feature under consideration and the corresponding outcome. Rank ties are assigned a rank equal to the average of their positions in the ascending order of the values. We then compute the ***Spearman's Rank Correlation Coefficient*** which has the same formulation as that outlined in section [3.14.1.2]

3. 12.1.2.2 Pearson's Correlation Coefficient

When the attribute values are defined on the interval or ratio scale, the irrelevant attributes can be identified by testing whether the Pearson correlation coefficient $r = 0$ .The Pearson correlation coefficient for measure of correlation between attributes A_j and the outcome (click/no click) is given by the formula

$$r = \frac{\sum_{i=1}^{I}(x_{ij} - \bar{x}_{.j})(c_i - \bar{c})}{\sqrt{\sum_{i=1}^{I}(x_{ij} - \bar{x}_{.j})^2}\sqrt{\sum_{i=1}^{I}(c_i - \bar{c})^2}} \tag{31}$$

Where I is the number of impressions in the training set $\aleph \setminus \tilde{\aleph}$ .The measure of correlation between two attributes A_j and A_k is given by the formula

$$r = \frac{\sum_{i=1}^{I}(x_{ij} - \bar{x}_{.j})(x_{ik} - \bar{x}_{.k})}{\sqrt{\sum_{i=1}^{I}(x_{ij} - \bar{x}_{.j})^2}\sqrt{\sum_{i=1}^{I}(x_{ik} - \bar{x}_{.k})^2}} \tag{32}$$

When the sample size is large i.e $I > 25$, the sample correlation coefficient $r \sim N\left(0, \frac{1}{I-2}\right)$. Therefore an approximate confidence interval for K samples can be defined to assess whether $'r'$ is statistically different from zero [48].

$$P\left(\frac{-Z_{\alpha/2K}}{\sqrt{I-2}} < r < \frac{Z_{\alpha/2K}}{\sqrt{I-2}}\right) = 1 - \frac{\alpha}{K} \tag{33}$$

Finally, the test hypothesis for the population correlation coefficient ρ is defined as

$$H_0 : \rho = 0 \text{ vs } H_1 : \rho \neq 0 \tag{34}$$

Therefore we reject the null hypothesis if the computed sample correlation coefficient is lies outside the confidence interval defined in equation [33].

The feature selection algorithm now created as an ensemble of a filter and a wrapper is defined next

- ✓ **Step 1:** Set $F = \phi$. Construct an upper (lower) triangular matrix A of dimension $J \times J$,containing the measure of association between two attributes
- ✓ **Step 2:** Construct a $J \times 1$ matrix named B wherein B_j indicates the degree of association between attribute A_j and the outcome (click/no-click)
- ✓ **Step 3:** Select $\arg\min_{ij} A_{ij}$ *and* $\arg\max_{j} B_j$ and add to F
- ✓ **Step 4:** Call the *Backward Selection Algortihm* described in Section [3.13.1.1.2]

The obvious advantage of combining a filter with a wrapper is that it reduces the computational complexity of the feature selection algorithm.

3. 12.1.3 Embedded Methods

Embedded methods perform feature selection in the process of training [45].This is a useful feature selection technique for logistic regression and support vector machines when the training costs are high. Feature selection in logistic regression models can be done by computing the t-ratio which is the estimate of β_j computed by the logistic regression model divided by the estimated standard deviation. Therefore we formulate the null hypothesis

$$H_0 : \beta_j = 0 \forall j = 1, ..., J \tag{35}$$

And accept it if the t-ratio is significant at $t_{1-\alpha/2,\mathrm{I-J-1}}$ .Accepting the null hypothesis results in the removal of features that are irrelevant.

The reduced model can be further assessed by computing the Akaike Information Criterion (AIC) given in equation [36]

$$AIC(J) = 2J + D(J) \tag{36}$$

Where J is the number of features in the model and $D(J)$ is the measure of goodness of fit of the model with J features to the data and is measured as the difference between the fitted model and the residual model. Alternately, Bayes Information Criterion (BIC), can be computed in which $2J$ in equation [36] is replaced with $J \log I$ .The set of features which minimize the AIC or BIC are considered relevant. Once the relevant attributes have been identified with retrain the model with these attributes and then cross validate on the validation sets.

4. Results

5. Conclusions and Recommendations

The Insights System was evaluated for its ability to process large data sets and computation of various attributes. The system could successfully process 22 M impressions and 100 M bid requests. It could further create a viewer profile of 10M viewers and computed the values of attributes for each these viewers. A total of twenty-five derived attributes were computed on the Insights System. After the data processing stage, a subset of the data was used to train different machine learning and feature selection algorithms in order to build a model that can predict the likelihood of an impression being clicked or not. In this regard, a number of classification algorithms and feature selection algorithms were evaluated. For the task of predicting the outcome of an impression, algorithms like Naïve Bayes, Logistic Regression and Support Vector Machines were used. For the task of identifying the relevant attributes from the full attribute set, Filter, Wrapper and Embedded methods were used. The predictions of the models were validated against the actual outcome of seven distinct validation sets and methods like ROC analysis and statistical significance tests were used to draw conclusions on the performance of the models.

5.1 Conclusions

During the data processing stage, it was observed that the use of MongoDB as a central data store seems to lead to a performance bottleneck due to its high latency for write transactions. In order to boost performance, new shards were added to the Mongo cluster. However, addition of Mongo shards during the data processing stage further slowed down the process as all read and write transactions were locked until the mongo router had redistributed all data to the new shards.

During the data analysis stage, it was observed that exploratory data analysis reveals to a good extent how different features are related to the outcome (click/no click) and the shape of the conditional density function of each attribute. Due to the high dimensionality and large number of impressions, it was found that wrappers have a high time complexity .Therefore, a combination of filter and wrapper methods were found to work best wherein a filter method was used to reduce the dimensionality of the dataset by selecting attributes that have a high correlation with the outcome while a low correlation among themselves. To measure existence of multicollinearity among attributes and correlation with the outcome, association matrices can be constructed. The reduced feature set was then passed to a wrapper method to identify the set of relevant features. ***It was observed that the Naïve Bayes algorithm when trained on the data set containing only the relevant features, has the best accuracy in predicting, whether an impression will result in a click or not***. However, it has been observed that Naïve Bayes estimates very low click probabilities for impressions, due to the existence of class skew and this leads us to consider impressions that have an estimated click probability greater than 0.

On the other hand, Logistic Regression when trained on the full feature set estimates higher click probabilities for impressions than Naïve Bayes but the estimated click probabilities are either ≈ 0 or ≈ 1 which in turn leads to a large number of false negatives. However, using an embedded feature selection algorithm to identify relevant features and then training a Logistic Regression algorithm on the relevant feature set results in a more continuous probability estimate but the probability estimates are lower as compared to the ones from the full feature set. While Naïve Bayes could be trained on a very large training set owing to its low computational complexity whereas the Logistic Regression algorithm had to be trained on smaller training sets as it has a higher computational complexity than Naïve Bayes. Moreover, it was observed that altering the distribution of %of clicks in a training set from 0.1% to 0.9% does not really improve the performance of the model, nor does scaling down of numeric feature to the

interval [0,1]. Finally, it was concluded that Logistic Regression with feature selection does not perform as well as Naïve Bayes with feature selection mainly because of the suboptimal grouping of features like DOMAIN into categories based on their estimated click probabilities. To verify this claim, the Naïve Bayes model was used to estimate the click probabilities without the DOMAIN feature and it was observed that the AUC dropped significantly, implying poor model performance.

Finally, training a Support Vector Machine showed that it has a higher computational complexity than both Naïve Bayes and Logistic Regression and therefore estimating the penalty parameter, C and kernel parameter, γ turned out to be infeasible for large training sets. We therefore had to train an SVM on the full feature set, with $C=1$ and $\gamma=1/36$ which resulted in a model that could classify clicks from non-clicks unlike Naïve Bayes and Logistic Regression models which did not perform well as classifiers and we therefore had to rely on the probability estimates returned by them. However, the inability to train an SVM with large penalty parameter, C because of its high computational complexity prevented us from improving the performance of SVMs. Moreover, using an embedded feature selection method to improve the performance of SVM reduced the computational complexity of training an SVM but also reduced the accuracy of the model, thereby leading us to conclude that embedded feature selection methods don't work well with SVMs.

5.2 Recommendations

It is recommended to segregate the data processing and attribute computation tasks on the Insights System into two parts namely (a) batch writes and (b) aggregated writes The batch writes should include tasks that require a large number of writes to the disk and should use an alternate database to MongoDB. This means that all the Extract and Transform tasks of the ETL process as explained in this report, should use a database that offers low write latency.

The Naïve Bayes model used in this research used a normal kernel function to estimate the conditional probability densities of various attributes. It would be interesting to evaluate the performance of the model using alternate kernel functions. Moreover, alternate feature selection algorithms like entropy based methods (Kullback-Leibler measure) and unsupervised learning methods (Relief) could be investigated. Similarly, statistical properties of AUC should be further investigated so that confidence intervals can be constructed for the AUC which would enable us to draw statistically relevant conclusions can be drawn from the AUC. Moreover, ANOVA and non-parametric tests can be used to compare the performance of multiple classifiers. The Logistic Regression model falls behind Naïve Bayes in terms of performance because of the suboptimal grouping of features with large number of values into groups. Therefore alternate methods need to be investigated into for categorizing features with large number of values. Similarly, parallelizing the Fisher Scoring/Batch Gradient Descent method of Logistic Regression could improve its training time. Moreover, alternate transformation methods need to be investigated for transformation of features with large numeric ranges and their impact on model performance should be evaluated. Similarly, if alternate methods are found that can estimate the penalty and the kernel parameters of SVMs, at a lower computational complexity, it would then be interesting to see the prediction accuracy of these models.

6. Appendix

6.1 Appendix A-Data Flow Diagram (Extraction Process)

6.2 Appendix B-Data Flow Diagram (Transform Process)

6.3 Appendix C-Entity Relationship Diagram

6.4 Appendix D-Class Diagram

References

[1] *http://www.adscience.nl*

[2] Zhang Harry, Su Jiang *Naïve Bayesian Classifiers for Ranking*

[3] Ng Andrew, Jordan Michael *On discriminative vs. Generative Classifiers*

[4] Hristov Asparuh *Retargeting and Audience Targeting on a Real-Time Bidding Platform*

[5] Smith M J *Statistical Analysis Handbook*

[6] Ullman et all *Mining of Massive Datasets*

[7] http://en.wikipedia.org/wiki/Feature_selection

[8] http://en.wikipedia.org/wiki/McNemar%27s_test

[9] Zhao et all *Advancing Feature Selection Research*

[10] Alelyani Salem, *On Feature Selection Stability: A Data Perspective*

[11] Kalousis et all *Stability of Feature Selection Algorithms: a study of high dimensional spaces*

[12]Yu Ying, *SVM-RFE Algorithm for Gene Feature Selection*

[13] Fawcett Tom, *An introduction to ROC analysis*

[14] Chuck Lam, *Hadoop in Action*

[15] http://docs.mongodb.org/manual/

[16] http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/

[17] http://www.r-project.org/

[18] http://cran.r-project.org/web/packages/e1071/index.html

[19] M.C.M de Gunst, *Statistical Models*

[20] M.C.M de Gunst and M.P. McAssey, *R Maual for Statistical Models*

[21] http://www.mongodb.org/

[22] http://docs.mongodb.org/manual/core/sharded-cluster-architectures/

[23] http://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH4/4.3.0/CDH4-Installation-Guide/cdh4ig_topic_18.html

[24] http://www.slideshare.net/spf13/mongodb-and-hadoop

[25] http://sergei.clustrix.com/2011/01/mongodb-vs-clustrix-comparison-part-1.html

[26] http://cran.r-project.org/web/packages/ROCR/index.html

[27] http://cran.r-project.org/web/packages/rmongodb/index.html

[28] https://github.com/nexr/RHive

[29] http://www.slideshare.net/spf13/mongodb-and-hadoop

[30] Thusso et all, *Hive – A Warehousing Solution Over a Map-Reduce Framework*

[31] http://dev.maxmind.com/geoip/legacy/geolite/

[32] http://www.worldweatheronline.com/free-weather-feed.aspx

[33] http://adsrv10.prolocation.net/mapred/html/mapredClasses/

[34] http://adsrv10.prolocation.net:8080/citrine/tasks.do?selectedGroupName=-

[35] http://adsrv10.prolocation.net/rockmongo/index.php?action=admin.index

[36] http://adsrv10.prolocation.net:8787/

[37] http://adsrv10.prolocation.net:50030/jobtracker.jsp

[38] http://adsrv10.prolocation.net:50070/dfshealth.jsp

[39] http://www.slideshare.net/mongodb/r-statistics-with-mongo-db-28536016

[40] http://cran.r-project.org/web/packages/foreach/vignettes/nested.pdf

[41] http://www.exegetic.biz/blog/2013/08/the-wonders-of-foreach/

[42] http://cran.r-project.org/web/packages/pracma/pracma.pdf

[43] Graham N.E.,*Areas beneath the relative operating characteristics (ROC) and relative operating levels (ROC) curves: Statistical Significance and interpretation.*

[44] E Alpaydin, *Introduction to machine learning*

[45] Guyon et al, *An Introduction to Variable and Feature Selection*

[46] http://uregina.ca/~gingrich/ch11a.pdf

[47] Smith MJ, *Statistical Analysis Handbook*

[48] http://www.ltrr.arizona.edu/~dmeko/notes_9.pdf

[49] http://robjhyndman.com/hyndsight/transformations/

[50] Hsu et al, *A Practical Guide to Support Vector Classification*

[51] Perlich Claudia et all , *Bid Optimization and Inventory Scoring in Targeted Online Advertising*