Prediction and Visualization of User Tendency Towards Purchases using Clickstream Data

Submitted in partial fulfillment of the requirements
of the degree of Bachelor of Engineering in Computer Engineering

B. E. Computer Engineering

By

Dakshil Shah	**60004120090**
Samkeet Shah	**60004120103**
Jamshed Shapoorjee	**60004120109**

Guide:

Kiran Bhowmick
Assistant Professor

Department of Computer Engineering
D. J. Sanghvi College of Engineering
Mumbai – 400 056

University of Mumbai
2015-2016

CERTIFICATE

This is to certify that the project entitled **Prediction and Visualization of User Tendency Towards Purchases using Clickstream Data** is a bonafide work of **Dakshil Shah(60004120090), Samkeet Shah(60004120103), Jamshed Shapoorjee(60004120109)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of B.E. in Computer Engineering

Prof. Kiran Bhowmick
Internal Guide

Dr. Narendra Shekokar
Head of Department

Dr. Hari Vasudevan
Principal

Project Report Approval for B.E.

This project report entitled ***Prediction and Visualization of User Tendency Towards Purchases using Clickstream Data*** by ***Dakshil Shah, Samkeet Shah, Jamshed Shapoorjee*** is approved for the degree of ***B.E. in Computer Engineering.***

Examiners

1.----------

2.----------

Date:

Place:

Declaration

We declare that this written submission represents my/our ideas in my/our own words and where others' ideas or words have been included, I/We have adequately cited and referenced the original sources. I/We also declare that I/We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my/our submission. I/We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Samkeet Shah
(60004120103)

Date:

Abstract

As of today, online shopping has captured a fair share of the retail market. Customers prefer to shop online as they can compare prices of a product across various stores, buy products while at home or traveling and pay using e-payment methods. Due to the simple usability, e-commerce websites are gaining more customers by the day. As the number of e-commerce websites is increasing, companies need to discover trends in user patterns so as to boost sales. One such method is the prediction of user purchases. A company can predict that a certain customer is going to purchase a certain product. It is necessary to have a good idea about the users' buying and browsing pattern on Ecommerce websites, so that online retailers can improve upon their system to attract and benefit more users. Traditional methods of gathering user data include analyzing their browsing patterns which primarily indicate their interests. Hence Clickstream data, that indicate the browsing paths and visiting frequency along with time spent per category or product is a largely unexplored area for predicting user behavior is a largely unexplored domain for gathering user data. We propose a model to analyze clickstream data and to predict which sessions lead to possible buys and of which items.

.

Contents

List of Figures

List of Tables

List of Abbreviations

Sr. No.	Abbreviation	Expanded form
1	SVM	Support Vector Machine
2	RF	Random Forest
3	DT	Decision Tree
4	PCA	Principal Component Analysis
5	CFS	Correlation-based feature selection
6	ID3	Iterative Dichotomiser 3
7	CART	Classification and Regression Tree
8	RVM	Relevance Vector Machine
9	RAM	Random Access Memory
10	DFD	Data Flow Diagram
11	MLP	Multilayer Perceptron

CHAPTER 1: INTRODUCTION

1.1. DESCRIPTION

The aim of this project is to use the available and collected clickstream data i.e., the sequence of click events for a particular session use on the online retailers' website to analyze the clickstream data and predict whether the use will buy the product or not and if so what item will he buy. Secondly we analyze this available clickstream data of all users, to generate an analytical model for the online retailers such as the buying behavior of the users during the day/week/months and which in turn can be used to generate offers and promotions to attract more users.

1.2. PROBLEM FOUNDATION

The problem provides a collection of sequences of click events; click sessions. For some of the sessions, there are also buying events. The goal is hence to predict whether the user (a session) is going to buy something or not, and if he is buying, what would be the items he is going to buy. Such information is of high value to an e-business as it can indicate not only what items to suggest to the user but also how it can encourage the user to become a buyer. For instance, to provide the user some dedicated promotions, discounts etc. The data represents six months of activities of an e-commerce business in Europe selling all kinds of stuff such as garden tools, toys, clothes, electronics and much more.

1.3. MOTIVATION

As of today, online shopping has captured a fair share of the retail market. Customers prefer to shop online as they can compare prices of a product across various stores, buy products while at home or traveling and pay using e-payment methods. Due to the simple usability, e-commerce websites are gaining more customers by the day. As the number of e-commerce websites is increasing, companies need to discover trends in user patterns so as to boost sales. One such method is the prediction of user purchases. If a company can predict that a certain customer is going to purchase a certain product. It is necessary to have a good idea about the users' buying and browsing pattern on E-commerce websites, so that online retailers can improve upon their system to attract and benefit more users. Traditional methods of gathering user data include analyzing

their browsing patterns which primarily indicate their interests. Hence Clickstream data, that indicate the browsing paths and visiting frequency along with time spent per category or product is a largely unexplored area for predicting user behavior is a largely unexplored domain for gathering user data. Hence we suggest this novel method of analyzing user buying behavior to predict the probability of a purchase as well as visualize the overall user pattern, based upon the collected clickstream data.

1.4. PROPOSED SOLUTION

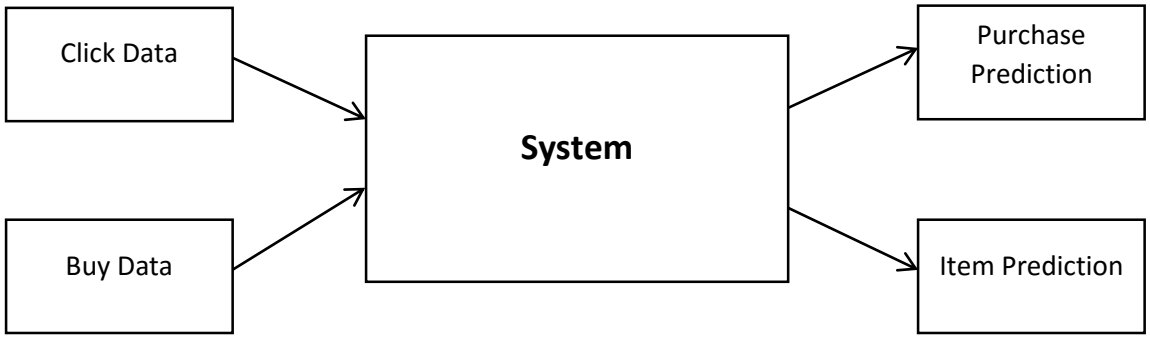

Figure 1.1. Block Diagram of Proposed System

We use a dataset with obtained from YOOCHOOSE, who have collected from user visits over a period of 6 months from a European retailer's website. The dataset consists of two files namely, clicks and buys. The clicks data contains information about every click made by a user on the website. The database undergoes preprocessing first, that involves filling in missing values, smoothing noisy data, identifying and removing outliers, and resolving inconsistencies. The data is then reduced by using data reduction which results in a reduced representation of the dataset that is smaller in volume than the original, but still produces the same analytical results. Preprocessing of data is followed by feature selection and feature extraction. Feature selection is done so as to make the data easier to interpret with reference to system, reduce training time. Based upon the selected features, the system is trained for prediction of user tendency to buy the item in his future sessions. The database is used to visualize the trends in the user buying behavior such as peak buying time/day/ month of the year. The trained data is then used to predict the probability of buying i.e. either a yes or a nay for the buy as well as what item is the user going to buy.

1.5. SCOPE OF THE PROJECT

In this project, we analyze a dataset obtained from an e-commerce website that contains clicks of the users with details like session id, timestamp, item id, category, etc. These clicks are grouped by sessions, which make up clickstreams.

Since clickstreams may contain dirty data and may be in a very complicated format we need to preprocess it. We then need to extract features from the raw data. These features are further selected and narrowed down to those which can be used to predict whether a user will make a purchase and if yes, what item will be purchased by the customer using various machine learning algorithms. The dataset is also analyzed visually to depict user purchase patterns and behavior.

Machine learning is applied on the training data provided to give us a trained model. Further, this model is used to test the results using the test data. The main aim of this project is to provide the most accurate results even with the least amount of resources used.

CHAPTER 2: LITERATURE REVIEW

2.1. FEATURE EXTRACTION

Features are functions of the original measurement variables that help in classification. Feature extraction is the process of defining a set of features, which will efficiently represent the information which is important for the classification problem. The goal of feature extraction is to improve the effectiveness and efficiency of analysis and classification. By reducing the dimensionality of the input set, correlated information is eliminated at the cost of accuracy. Feature space expansion involves generating new features based on the original ones instead of reducing dimensionality. Attribute construction is the process of constructing new attributes from the given attributes, so as to help improve the accuracy and understanding of structure in high-dimensional data. Accuracy measures the ratio of correctly classified instances across all classes. Feature subset selection removes redundant features from the data set as they can lead to a reduction of the classifier accuracy or clustering quality and lead to an increase in computational cost The advantage is that no information about a single feature is lost. However, if a small set of features is required and the original features are very diverse, information may be lost, which in turn will reduce the accuracy of classification. The dataset used by us, contains many hidden features which play a vital role in reaching the goal of our project. We propose a method where we first carry out attribute construction, followed by feature subset selection.

2.2. FEATURE CONSTRUCTION METHODS

A common approach for constructing a feature set such that the new feature space is larger than the original feature space ($|F_n| > |F_o|$), so as to improve accuracy, is to apply a set of operators (eg. {+, -, *}) on the original feature values. We need methods that:

- Generate a set of features that help improve prediction accuracy.
- Are computationally efficient.
- Are generalizable to different classifiers.
- Allow for easy addition of domain knowledge.

2.3. FEATURE SUBSET SELECTION

2.3.1. Filters

Filters select the feature subsets independent of the machine learning/prediction algorithm to be applied later. They provide a generic ordering of features which are not adapted for any particular learning algorithm. Information gain can be used to measure how well each single feature separates the dataset. Some methods may select variables by computing correlation with the output or through techniques such as clustering or principal component analysis (PCA).

2.3.2. Correlation-based feature selection (CFS):

CFS uses a search algorithm along with a function to evaluate the merit of feature subsets. The goodness (G) of the feature subset considers the usefulness of individual features for predicting the class label along with level of inter-correlation among them.

2.3.3. PCA

PCA tries to find a new set of attributes or PCs which are linear combinations of original attributes, orthogonal to each other, and capture the maximum amount of variation in the data. The covariance of 2 random variable x and y of a sample with size n and mean $\bar{x}$ and $\bar{y}$ can be calculated as

$$\mathrm{Cov}(x, y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

2.3.4. Wrappers

Wrappers are methods that use the learning method to be used for prediction, to select the features. They search through the space of features and estimate the accuracy of a single learning algorithm for each feature that can added or removed from the subset. The training set is divided a train and validation set. For a given feature subset, the predictor is trained on the train set and the accuracy is calculated using the validation set. The highest scoring feature subset is considered. The wrapper method is a search in a space.

$W= \{0,1\}^n$ where w is a subset of W such that w[j]=0 if the input j does not belong to the set of features =1 if the input j belongs to the set of features. We need to find an optimal vector $w^* = \arg\min_{w \in W} MSE_w$ where MSE_w is the generalized error of the model based on the set of variables described by w. From above, it can be inferred that for n attributes, there are 2^n possible subsets. An exhaustive search for finding the optimal subset is not feasible. To solve this, a heuristic approach is used.

2.3.5. Stepwise forward selection

We start with an empty set of attributes as the reduced set. The best of the original attributes is determined and added to this set. In each subsequent iteration, the best of the remaining attributes is added. Lowest generalization error can be considered as a selection factor. The second input is selected such that it has the lowest error together with the first one until no improvement is possible.

2.3.6. Stepwise backward elimination

We start with the full set of attributes as the reduced set. In each subsequent iteration, the worst of the remaining attributes is removed. Combination of forward selection and backward elimination. In each iteration, the best attribute is selected and the worst is eliminated.

2.3.7. Decision tree induction

Decision tree algorithms such as ID3, C4.5, and CART may be used although originally they are classification algorithms. It involves constructing a decision tree where each non leaf node denotes a test on an attribute and each branch corresponds to an outcome of that test. The leaf nodes represent class prediction. The algorithm chooses the best attribute to partition the data into individual classes at each node. A tree is constructed of all the data. Attributes which are absent are assumed to be irrelevant and those appearing, form the reduced attribute subset. Some popular attribute selection methods are as follows.

2.3.8. Information Gain

ID3 uses information gain as its attribute selection measure. Consider node N, which represents or holds the tuples of partition D. The attribute with the highest information gain

is chosen as the splitting attribute for node N. This attribute minimizes the information needed to classify the tuples in the resulting partitions and indicates least randomness in these partitions. This minimizes the number of tests needed to classify a given tuple and guarantees that a simple tree is found. The information needed to classify a tuple in D is given by $\text{Info}(D0 = -\sum_{i=1}^{m} p_i \log_2(p_i)$; where p_i is the probability that an arbitrary tuple in D belongs to class C_i and is estimated by $|C_{i,D}|/|D|$. Info(D) is the average amount of information needed to identify the class label of a tuple in D. Info(D) can also be referred to as the entropy of D. If we partition the tuples D on some attribute A having v distinct values into v subsets, then these partitions correspond to the branches frown from node N. The amount of information still needed to arrive at an exact classification is given by

$$\text{Info}_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \text{Info}(D_j)$$

Information Gain can be defined as the difference between the original information requirement and the new requirement $\text{Gain}(A) = \text{Info}(D) - \text{Info}_A(D)$

2.3.9. Gain Ratio

C4.5 uses and extension to information gain i.e. gain ratio. Information Gain is biased in the sense that it prefers to select attributes having a large number of values. Gain Ratio attempts to overcome this bias. It uses a split information value which is given as: $\text{SplitInfo}_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \log_2\left(\frac{|D_j|}{|D|}\right)$. The Gain Ratio is defined as $\text{GainRatio}(A) = \frac{\text{Gain}(A)}{\text{SplitInfo}(A)}$ The attribute with the maximum gain ratio is selected as the splitting attribute.

2.3.10. Gini Index

CART uses gini index for attribute selection. The Gini index measures the impurity of a data partition or set of train tuples D as $\text{Gini}(D) = 1 - \sum_{i=1}^{m} p_i^2$. Where p_i is the probability that a tuple in D belongs to class C_i and is estimated as $|C_{i,D}|/|D|$. Gini index considers a binary split for each attribute. We compute a weighted sum of the impurity of each resulting partition as

$$\text{Gini}_A(D) = \frac{|D_1|}{|D|} \text{Gini}(D_1) + \frac{|D_2|}{|D|} \text{Gini}(D_2)$$

Where we consider a binary split on attribute A, partitions D into D_i and D_2. The subset that gives the minimum Gini index for that attribute is selected as its splitting subset. The reduction in impurity that could be incurred by a binary split on an attribute A is given below and the attribute with minimum Gini index is selected as the splitting attribute.

$$\Delta\mathrm{Gini}(\mathrm{A}) = \mathrm{Gini}(\mathrm{D}) - \mathrm{Gini}_{\mathrm{A}}(\mathrm{D})$$

2.3.11. Embedded

These methods are highly specific to learning models. They combine the process of feature selection and model learning.

Table 2.1: Summary of attribute subset selection methods

Method	Pros	Cons
Filters	Easy to scale for high dimensional data sets. Computationally fast. Independent of classification algorithms. Feature selection needs to be done only once, followed by evaluation of different classifiers.	Ignores interaction with classifier. Each feature is considered separately and ignores feature dependencies.
Wrappers	Interaction feature subset selection and learning model. Takes into account feature dependencies.	Risk of overfitting. Computationally intensive if cost of building the classifier is high.
Embedded	Computationally less intensive than wrapper methods.	Specific to a learning machine

2.4. DATA MINING AND LEARNING ALGORITHMS

2.4.1. Decision tree

Decision Tree algorithm makes use of a graph like structure has a one root node and multiple leaf nodes. A root node is a node that has no incoming edges. It is the starting point of the tree structure. An internal node that has incoming as well as outgoing edges, it denotes the test condition that predicts the resultant outcome. The leaf nodes denote the final outcome of the tree. The learning and classification steps of a decision tree are simple and fast. Decision trees are a form of multiple variable (or multiple effect) analyses. All forms of multiple variable analyses allow us to predict, explain, describe, or classify an outcome (or target). An example of a multiple variable analysis is a probability of sale or the likelihood to respond to a marketing campaign as a result of the combined effects of multiple input variables, factors, or dimensions. [5] This multiple variable analysis capability of decision trees enables you to go beyond simple one-cause, one-effect relationships and to discover and describe things in the context of multiple influences. Multiple variable analysis is particularly important in current problem-solving because almost all critical outcomes that determine success are based on multiple factors. Further, it is becoming increasingly clear that while it is easy to set up one-cause, one-effect relationships in the form of tables or graphs, this approach can lead to costly and misleading outcomes.

Figure 2.1: Decision tree for whether to go to play based upon the current weather

2.4.2. C4.5

C4.5 is an algorithm used to generate a decision tree. The decision trees generated by C4.5 can be used for classification, and for this reason, C4.5 is often referred to as a statistical classifier. C4.5 builds decision trees from a set of training data in the same way as ID3, using the concept of information entropy. The training data is a set S = {s1, s2, ...} of already classified samples. Each sample si consists of a p-dimensional vector (x{1,i}, x{2,i}, ...,x_{p,i}) , where the xj represent attribute values or features of the sample, as well as the class in which si falls. At each node of the tree, C4.5 chooses the attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. The splitting criterion is the normalized information gain (difference in entropy). The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithms then recurs on the smaller sub-lists. This algorithm has a few base cases. All the samples in the list belong to the same class. When this happens, it simply creates a leaf node for the decision tree saying to choose that class. None of the features provide any information gain. In this case, C4.5 creates a decision node higher up the tree using the expected value of the class. Instance of previously-unseen class encountered. Again, C4.5 creates a decision node higher up the tree using the expected value.

2.4.3. Random forest

Random forest is a collection of decision tree, wherein the algorithm builds multiple decision tree and combines their output to obtain the final solution. Random forest algorithm is a very powerful algorithm which can be applied in various applications. The principle behind random forest algorithm is to combine many binary decision trees built using several bootstrap samples coming from the learning sample L and choosing randomly at each node a subset of explanatory variables X.

2.4.4. REPTree

REPTree uses the regression tree logic and creates multiple trees in different iterations. After that it selects best one from all generated trees. That will be considered as the representative. In pruning the tree, the measure used is the mean square error on the predictions made by the tree. Basically Reduced Error Pruning Tree ("REPT") is fast

decision tree learning and it builds a decision tree based on the information gain or reducing the variance. REP Tree is a fast decision tree learner which builds a decision/regression tree using information gain as the splitting criterion, and prunes it using reduced error pruning. It only sorts values for numeric attributes once. Missing values are dealt with using C4.5's method of using fractional instances.

2.4.5. SVM

Support Vector Machine uses decision plane concept to define boundaries that can classify data. For example, as shown in the diagram below the line is used to actually separate and classify the data set into the red part and the green part. Below is a graphical representation of how the data is classified based upon attributes of the data and how a partition is constructed to separate them SVM performs classification primarily by constructing hyperplanes in the multidimensional space and thus it separates and classifies data. In the case of linearly separable data, once the optimum separating hyperplane is found, data points that lie on its margins are known as support vector points and the solution is represented as a linear combination of only these points. Other data points are ignored. Therefore, the model complexity of an SVM is unaffected by the number of features encountered in the training data (the number of support vectors selected by the SVM learning algorithm, is usually small). For this reason, the SVM are well suited to deal with learning tasks where the number of features is large with respect to the number of training instances. Even though maximum margin allows SVM to select among multiple candidate hyperplanes at all because the data contains misclassified. The selection of an appropriate kernel function is important, since the kernel function defines the feature space in which the training set instances will be classified. Limitation of SVM is low speed of training. Training the SVM is done by solving Nth dimensional QP problem, where N is the number of sample in the training dataset. [4]

2.4.6. Bayesian Network

Bayes Network is based upon the Bayes theorem of statistical classification. Bayesian Network uses probabilities to check the likelihood of the membership of a particular data item to a pre-defined class. A Bayesian Network allows class conditional dependencies to

be defined between subsets and provides a graphical model of causal relationship on which learning model can be designed. This probability about the tuple that belongs to the particular class or not. Bayesian classification is based on Bayes theorem. Suppose X is a data tuple. It is considered as "evidence". H is the hypothesis that the data tuple X belongs to a specified class C. We also determine P(H/X) the probability that the hypothesis H holds given the evidence or observe data tuple X. P(H/X) is the posterior probability or a posteriori probability of H conditioned on X. For example, the data tuple is confined to the University described by attribute placement and results and X is a University with placements are good and results are average. Suppose H is a hypothesis that the university gets A grade from UGC. Then P(H/X) is the probability that University X will get A grade and results and placement are known. In contrast P(H) is the prior probability of H. For example, this probability that any given University will get A grade, regardless of placement and results. The posterior probability P(H/X) is based on more information about the University in comparison of prior probability P(H), which is independent of X. Similarly, P(X/H) is the posterior probability of X conditioned on H. That is University has good placement and average results, and University will get A grade. P(H), P(X/H) and P(X) may also be estimated from given data. Bayes theorem provides a useful way of calculating the posterior probability, P(H/X) from P(H), P(X/H) and P(X). Bayes theorem is as follows:

$$P(H/X) = P(X/H).\ P(H)/P(X)$$

2.4.7. Linear Regression

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable. For example, a modeler might want to relate the weights of individuals to their heights using a linear regression model. Least-Squares Regression: The most common method for fitting a regression line is the method of least-squares. This method calculates the best-fitting line for the observed data by minimizing the sum of the squares of the vertical deviations from each data point to the line (if a point lies on the fitted line exactly, then its vertical deviation is 0). Because the deviations are first squared, then summed, there are no cancellations between positive and

negative values. Before attempting to fit a linear model to observed data, a modeler should first determine whether or not there is a relationship between the variables of interest.

2.4.8. Ensemble Learning (bagging, boosting)

Bagging - bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. Given a standard training set D of size n, bagging generates m new training sets D_i, each of size n′, by sampling from D uniformly and with replacement. By sampling with replacement, some observations may be repeated in each D_i. If n′=n, then for large n the set D_i is expected to have the fraction (1 - 1/e) (≈63.2%) of the unique examples of D, the rest being duplicates. [1] This kind of sample is known as a bootstrap sample. The m models are fitted using the above m bootstrap samples and combined by averaging the output (for regression) or voting (for classification).

Boosting is a machine learning ensemble meta-algorithm for reducing bias primarily and also variance. in supervised learning, and a family of machine learning algorithms which convert weak learners to strong ones. While boosting is not algorithmically constrained, most boosting algorithms consist of iteratively learning weak classifiers with respect to a distribution and adding them to a final strong classifier. When they are added, they are typically weighted in some way that is usually related to the weak learners' accuracy. After a weak learner is added, the data is reweighted: examples that are misclassified gain weight and examples that are classified correctly lose weight. The main variation between many boosting algorithms is their method of weighting training data points and hypotheses. AdaBoost is very popular and perhaps the most significant historically as it was the first algorithm that could adapt to the weak learners. However, there are many more recent algorithms such as LPBoost, TotalBoost, BrownBoost, MadaBoost, LogitBoost, and others.

2.4.9. Relevance Vector Machine

A relevance vector machine (RVM) for data modeling is disclosed. The RVM is a probabilistic basis model. Sparsity is achieved through a Bayesian treatment, where a prior is introduced over the weights governed by a set of hyper parameters. As compared to a

Support Vector Machine (SVM), the non-zero weights in the RVM represent more prototypical examples of classes, which are termed relevance vectors. The trained RVM utilizes many fewer basis functions than the corresponding SVM, and typically superior test performance. No additional validation of parameters (such as C) is necessary to specify the model, except those associated with the basis.

CHAPTER 3: SYSTEM ANALYSIS

3.1. FUNCTIONAL REQUIREMENTS

- Interface: The input to the system consists of a dataset with clickstream data. The dataset is created by mining clickstream data of an e-commerce website. The input data should contain: Session id for the current session, Timestamp of the click, Item ID if product viewed/bought, Category of item, Price of item, Quantity of item purchased. The above fields have to be present in the clickstream dataset. Any missing values, extra fields have to be removed before analysis carried out by the model. For this, the data is preprocessed.
- Sufficient Data: To get an accurate classification model, the input test data must contain sufficient number of data entries to provide a good unbiased variation in the data to train the learning model. Consequently, the trained model gives a better prediction of unknown input test values.
- Structured Data: Data cleaning involves filling in missing values, smoothing noisy data, identifying and removing outliers, and resolving inconsistencies. The data is then reduced by using data reduction which results in a reduced representation of the dataset that is smaller in volume than the original, but still produces the same analytical results.
- Relevant Features: The model should select the features such as Total duration of the session, Maximal duration between two clicks, Average duration between clicks in the session, Maximal number of times the session has clicked on the same item, Average of items' buying probabilities in the session, Maximum of items' buying probabilities in the session, Probability of buying at least one item in the session, Maximal number of consecutive clicks on any item in, the session, to be used by the learning algorithm, Indicator of the item being the last item clicked, Number of times the item was clicked in the session, Time spent on the item during the session, Baseline probability of the item being bought, Number of consecutive clicks on the item, Maximal duration between consecutive clicks on item, Total number of clicks during the session, Average number of clicks per item, such that the prediction of the model is accurate.

- Graphical View of Data: The model should accurately visualize the dataset and provided a graphical interpretation of the same.

3.2. NON FUNCTIONAL REQUIREMENTS

- Reliability: The model must provide accurate results and work on even data which has inconsistencies.
- Performance: The model must be efficient in terms of time and space, providing fast and accurate results.
- Flexibility: The model should be portable to other systems datasets.

3.3. SPECIFIC REQUIREMENTS

- Weka version 3.7.x requires Java version 1.5 or greater
- Pandas, GraphLab Libraries require Python to be installed for the use of Data mining and machine learning tasks.
- System requirements RAM: 8GB or greater, Processor: Dual core or higher, OS: Ubuntu 12.04 or higher, Windows 7 or higher

3.4. USE-CASE DIAGRAMS AND DESCRIPTION

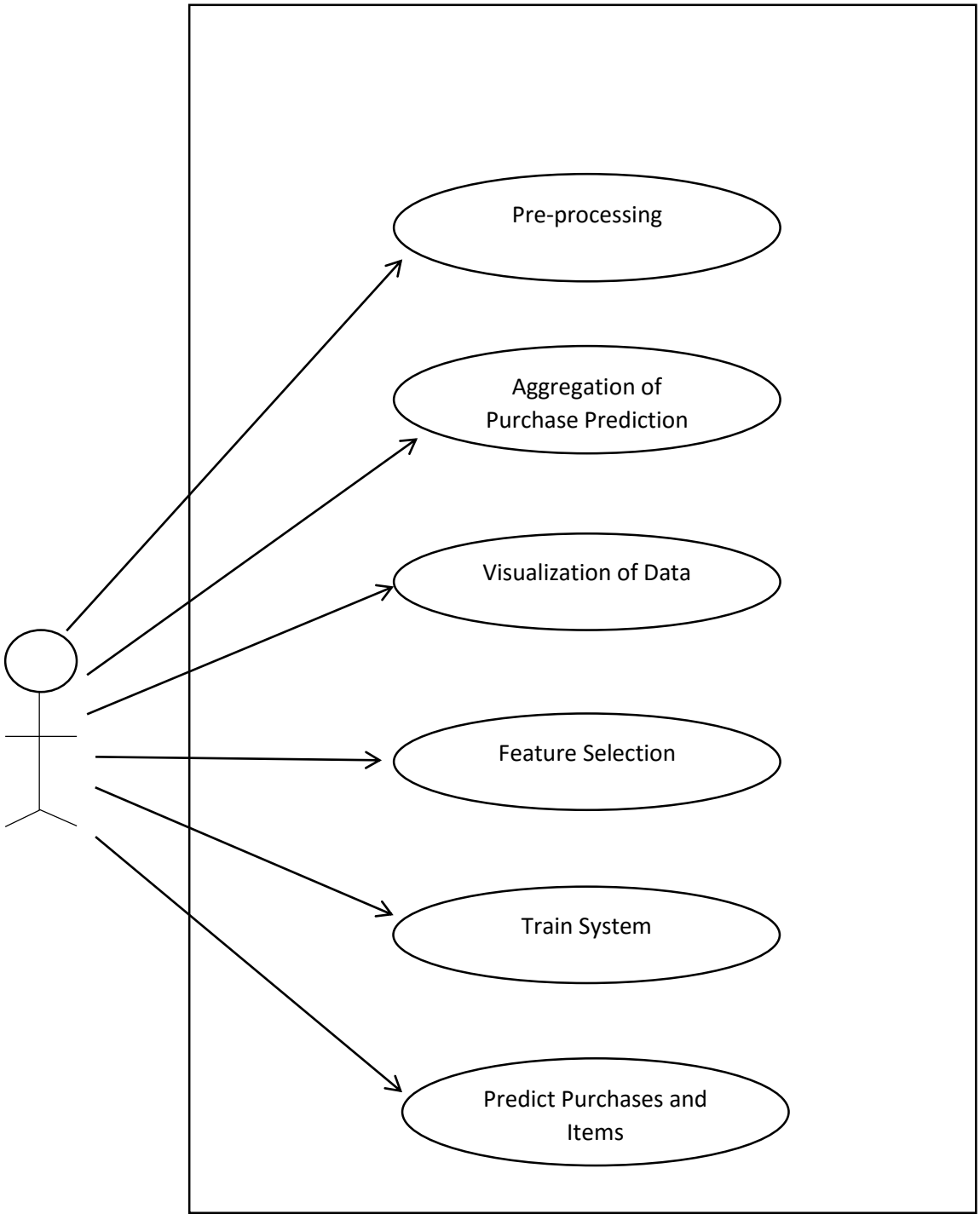

Figure 3.1. Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. In this system the admin of the system is the main user he can initiate various process in the system like data pre-processing, aggregation of session data, visualization, feature selection, training and after completing all these steps the user can use the system can use it to predict the items user is going to buy which can be used for various processes.

In the pre-processing step the data is cleaned and all incomplete data or duplicate data entries are removed. In the aggregation step we club all session data and their respective buy event together by matching their session id. In the visualization we plot various features graphically to get an idea of data distribution and select features accordingly. In feature selection we process the selection the selected features like session durations, no. of clicks, duration between clicks, etc. using various algorithms and techniques to get the final feature set that is to be used for training the system. In the training step we use various classification algorithms to make predictions related to the items the user may buy based on the clickstream data generated. Finally, we use the test data to test our system after the learning is completed and we measure the accuracy of the system, if found unsatisfactory we make changed to the algorithms or merge one or more of them.

CHAPTER 4: ANALYSIS MODELLING

4.1. DATA MODELLING

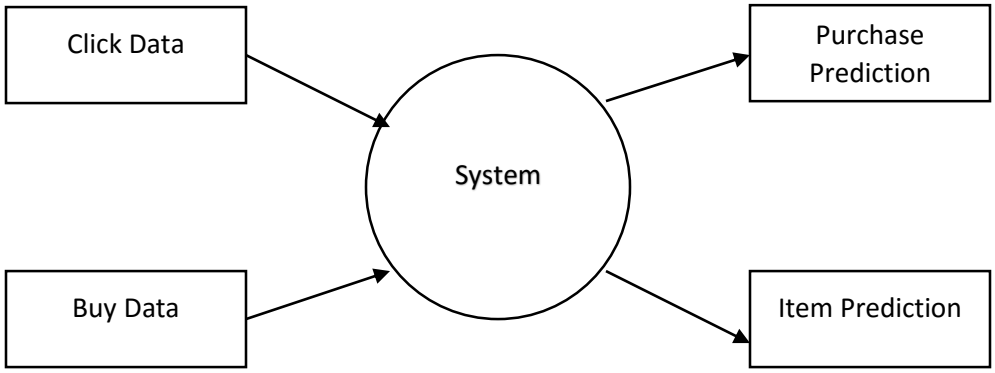

Figure 4.1: Level 0 DFD

Context level DFD, also known as level 0 DFD, sees the whole system as a single process and emphasis the interaction between the system and external entities. The system as a whole gets inputs data in the form of clicks and buys data and trains itself to predict the items. The system processes this data, trains itself and can finally predict whether the user will buy an item in this session and what items he shall buy. The system performs all tasks like pre-processing, feature selection, training and testing.

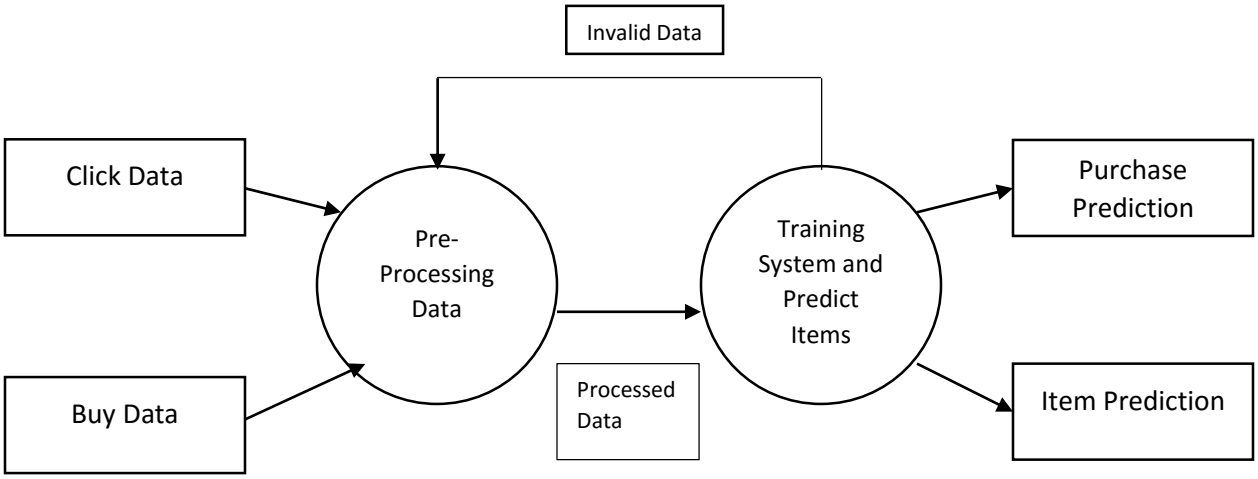

Figure 4.2: DFD Level 1

Level 1 Data Flow Diagram highlights the main functions carried out by the system. The main functions include pre-processing and training the processed data is passed to the training system which used it to predict the final outcomes. In pre-processing module, the data is cleaned and all incomplete data entries are removed further the features are identified and the selected features. These selected features are then used to train the system using learning algorithms which will enable it to predict what items users will buy

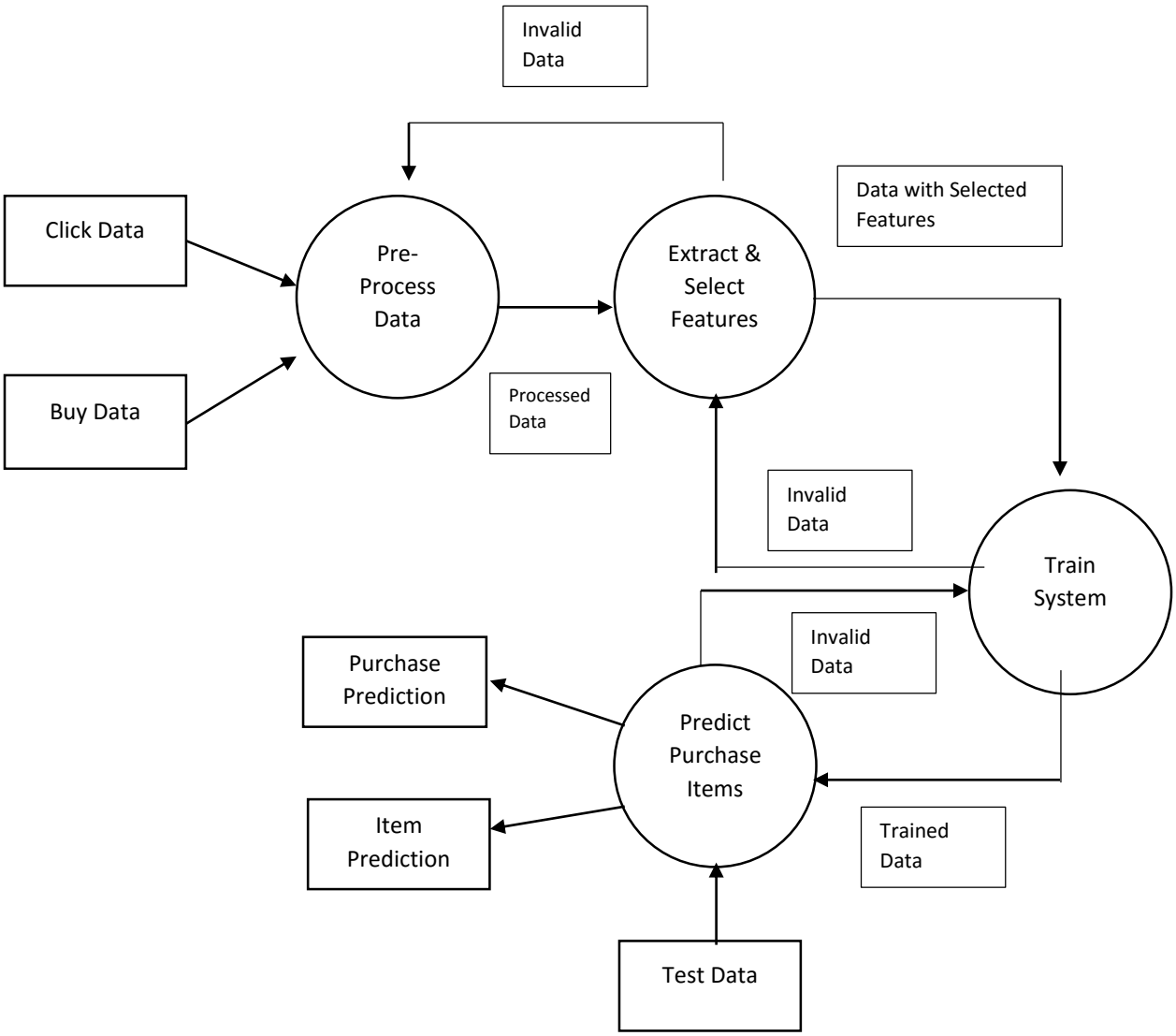

Figure 4.3: DFD Level 2

Level 2 expands on the main functionalities in the system by showing the various inputs and outputs. The detailed description of the system can be seen in this representation. The main modules of the system include pre-processing which passes on the processed data for feature extraction and selection after selecting the appropriate features are used to train the system. After training, the test data is used to finally predict the items the user may buy.

4.2. TIMELINE CHART

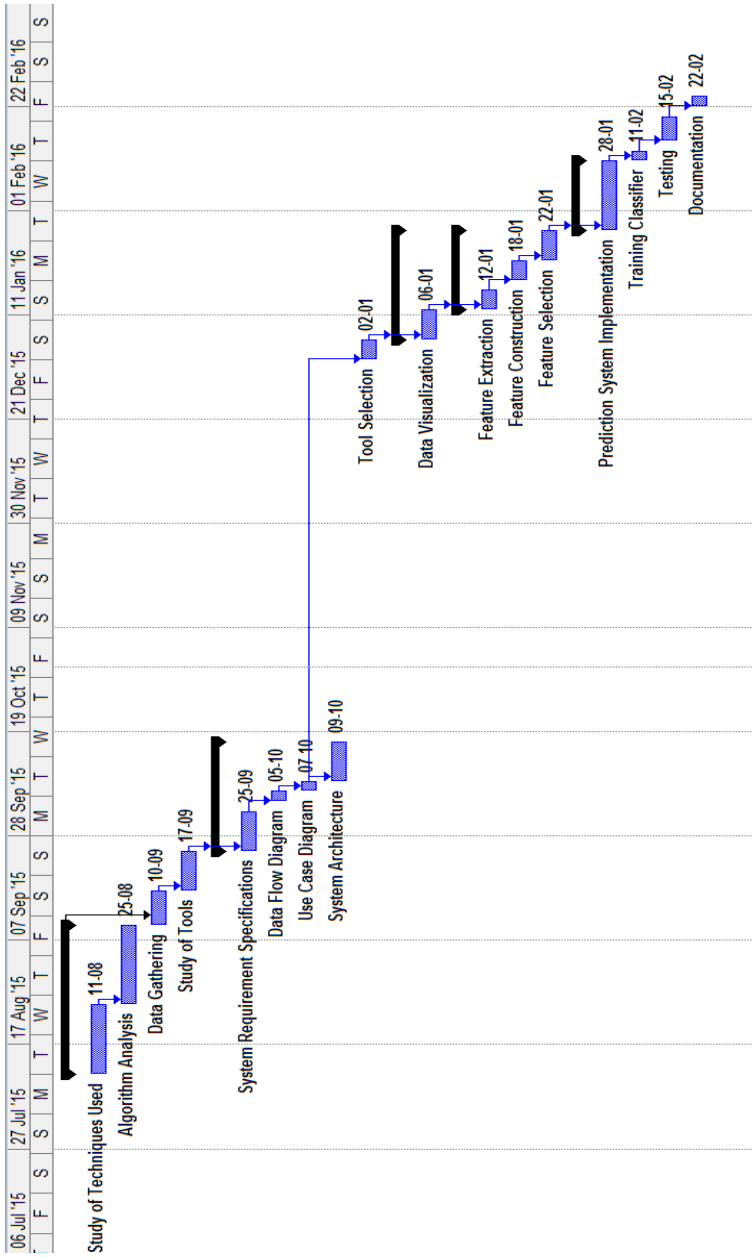

Figure 4.4: Timeline Chart

	Task Name	Duration	Start	Finish	Predecessors
1	**Literature Review**	**22 days**	**Tue 11-08-15**	**Wed 09-09-15**	
2	Study of Techniques Used	10 days	Tue 11-08-15	Mon 24-08-15	
3	Algorithm Analysis	12 days	Tue 25-08-15	Wed 09-09-15	2
4	Data Gathering	5 days	Thu 10-09-15	Wed 16-09-15	1
5	Study of Tools	6 days	Thu 17-09-15	Thu 24-09-15	4
6	**System Model Design**	**16 days**	**Fri 25-09-15**	**Fri 16-10-15**	**5**
7	System Requirement Specifications	6 days	Fri 25-09-15	Fri 02-10-15	5
8	Data Flow Diagram	2 days	Mon 05-10-15	Tue 06-10-15	7
9	Use Case Diagram	2 days	Wed 07-10-15	Thu 08-10-15	8
10	System Architecture	6 days	Fri 09-10-15	Fri 16-10-15	9
11	Tool Selection	3 days	Sat 02-01-16	Tue 05-01-16	9
12	**Data Processing**	**16 days**	**Wed 06-01-16**	**Wed 27-01-16**	**11**
13	Data Visualization	4 days	Wed 06-01-16	Mon 11-01-16	11
14	**Data Preprocessing**	**12 days**	**Tue 12-01-16**	**Wed 27-01-16**	**13**
15	Feature Extraction	4 days	Tue 12-01-16	Fri 15-01-16	13
16	Feature Construction	4 days	Mon 18-01-16	Thu 21-01-16	15
17	Feature Selection	4 days	Fri 22-01-16	Wed 27-01-16	16
18	**Model Implementation**	**10 days**	**Thu 28-01-16**	**Wed 10-02-16**	**17**
19	Prediction System Implementation	10 days	Thu 28-01-16	Wed 10-02-16	17
20	Training Classifier	2 days	Thu 11-02-16	Fri 12-02-16	19
21	Testing	5 days	Mon 15-02-16	Fri 19-02-16	20
22	Documentation	2 days	Mon 22-02-16	Tue 23-02-16	21

Table 4.5. Work Breakdown Structure

4.3. SYSTEM FLOWCHART

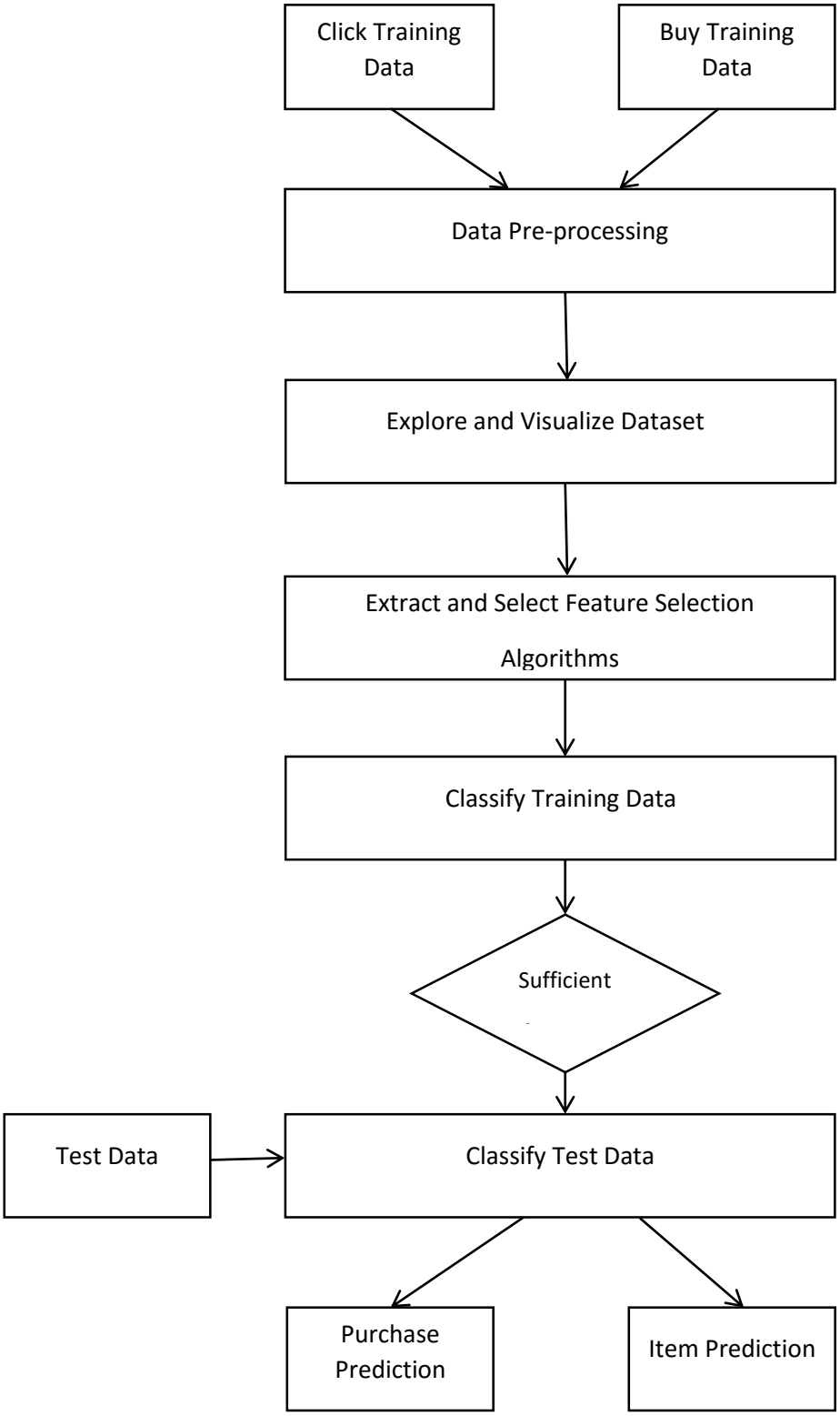

Figure 4.6.: System Flowchart

The system flowchart shows us the overall flow of the system. The initial input of the system is the two data files which are clicks data and buy data. The system is executed in two phases namely the training phase and the testing phase. The training phase involves pre-processing and applying classification algorithms. After a sufficient accuracy is met the system proceeds to the testing phase where the test data is used to finally predict item the user is going buy.

CHAPTER 5: DESIGN

5.1. ARCHITECTURAL DESIGN

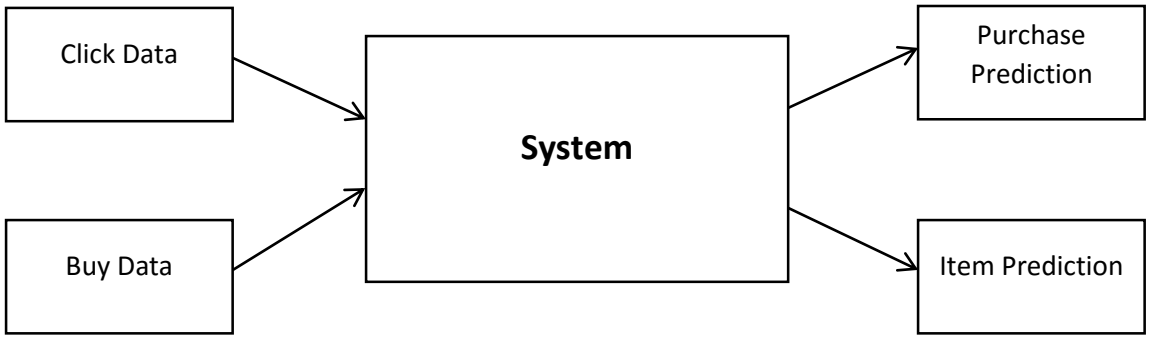

Figure 5.1: System Architecture

The database undergoes preprocessing first, that involves filling in missing values, smoothing noisy data, identifying and removing outliers, and resolving inconsistencies. The data is then reduced by using data reduction which results in a reduced representation of the dataset that is smaller in volume than the original, but still produces the same analytical results. Preprocessing of data is followed by feature selection and feature extraction. Feature selection is done so as to make the data easier to interpret with reference to system, reduce training time. Based upon the selected features, the system is trained for prediction of user tendency to buy the item in his future sessions. The database is used to visualize the trends in the user buying behavior such as peak buying time/day/ month of the year. The trained data is then used to predict the probability of buying i.e. either a yes or a nay for the buy as well as what item is the user going to buy.

5.2. USER INTERFACE DESIGN

The figures below show the user interface for interacting with main model. It allows the use to modify to select the database to be used for training. Once the dataset is loaded the user can run the individual phase on the dataset. In the end the results are obtained of whether a session results in a buy or not. If it results in a buy it predicts which item will be bought. The UI also supports viewing the visualizations on the dataset given.

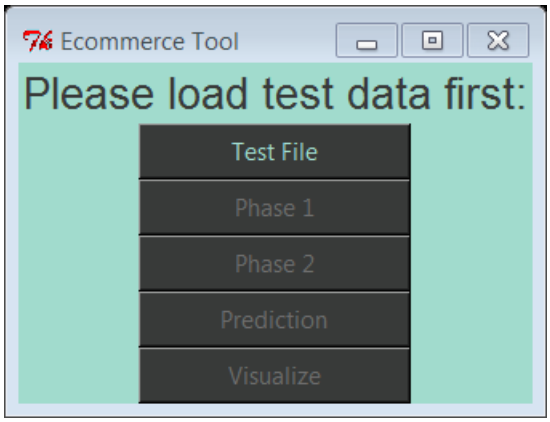

Figure 5.2. User Interface – Main Screen

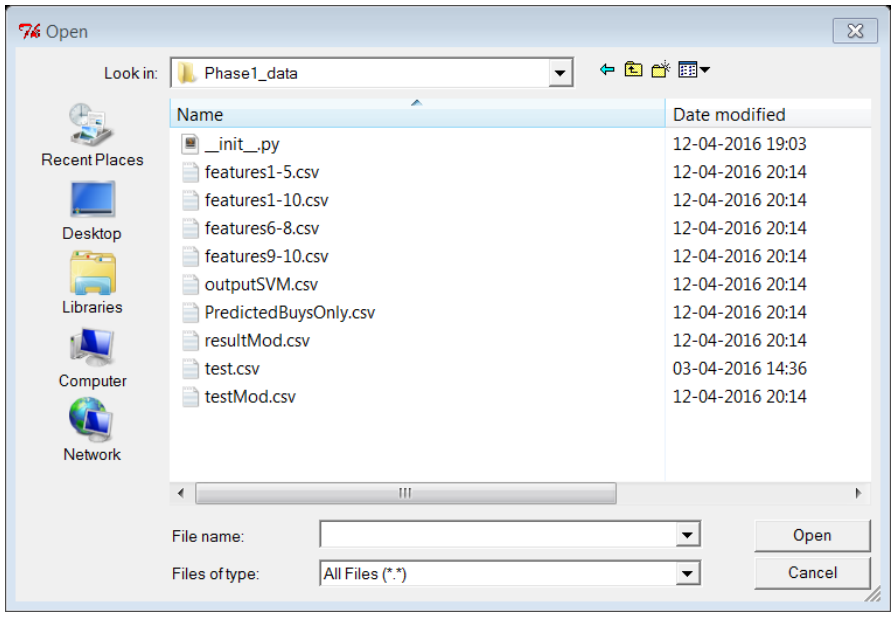

Figure 5.3 User Interface – Dataset Selection

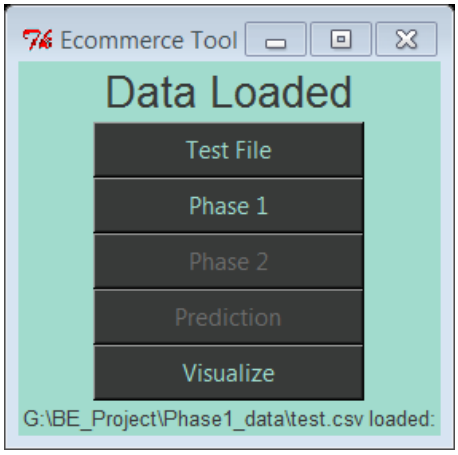

Figure 5.4. UI – Dataset Loading

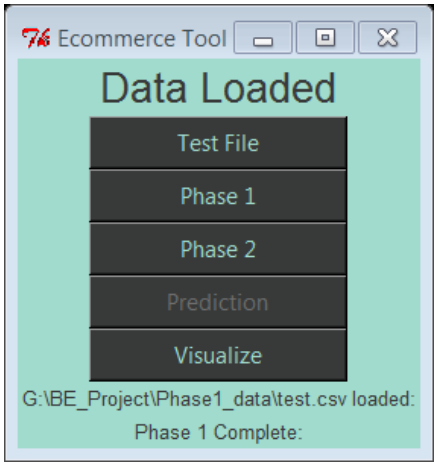

Figure 5.5. UI – Phase 1 Execution

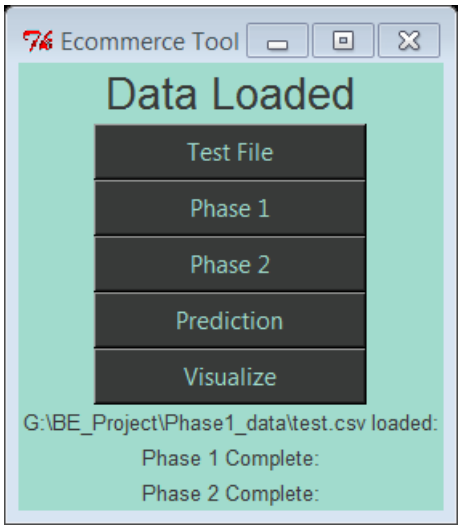

Figure 5.6. UI – Phase 2 Execution

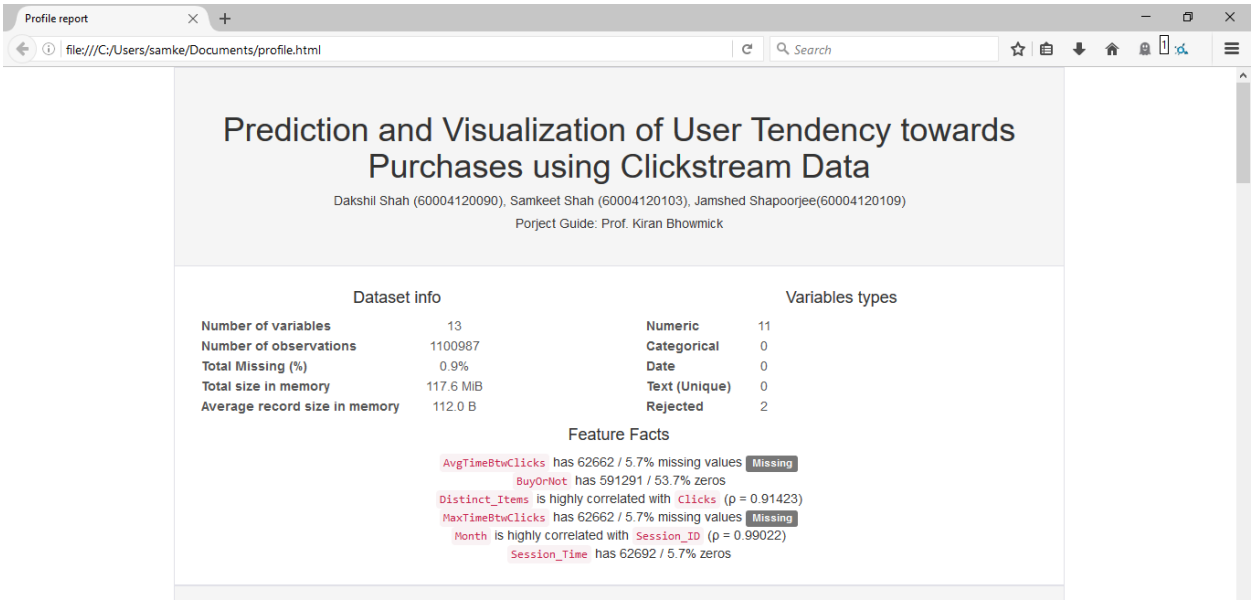

Figure 5.7. Project Visualization Info

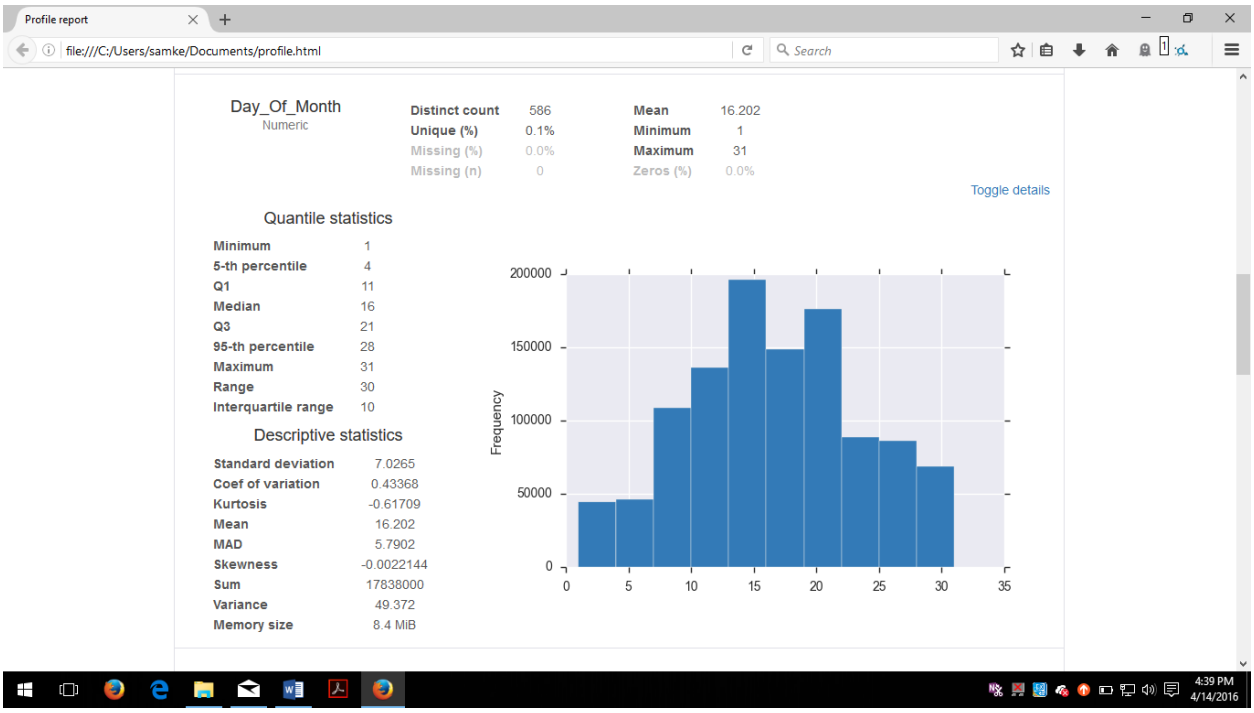

Figure 5.8. Visualization UI

CHAPTER 6: IMPLEMENTATION

6.1. ALGORITHMS USED

6.1.1. Support Vector Machine (SVM):

Support Vector Machine uses decision plane concept to define boundaries that can classify data. For example, as shown in the diagram below the line is used to actually separate and classify the data set into the red part and the green part. Below is a graphical representation of how the data is classified based upon attributes of the data and how a partition is constructed to separate them SVM performs classification primarily by constructing hyperplanes in the multidimensional space and thus it separates and classifies data. In the case of linearly separable data, once the optimum separating hyperplane is found, data points that lie on its margins are known as support vector points and the solution is represented as a linear combination of only these points. Other data points are ignored. Therefore, the model complexity of an SVM is unaffected by the number of features encountered in the training data (the number of support vectors selected by the SVM learning algorithm, is usually small). For this reason, the SVM are well suited to deal with learning tasks where the number of features is large with respect to the number of training instances. Even though maximum margin allows SVM to select among multiple candidate hyperplanes at all because the data contains misclassified. The selection of an appropriate kernel function is important, since the kernel function defines the feature space in which the training set instances will be classified. Limitation of SVM is low speed of training. Training the SVM is done by solving Nth dimensional QP problem, where N is the number of sample in the training dataset. [4]

6.1.2. Random Forest (RF)

Random forest is a collection of decision trees, wherein the algorithm builds multiple decision tree and combines their output to obtain the final solution. Random forest algorithm is a very powerful algorithm which can be applied in various applications. The principle behind random forest algorithm is to combine many binary decision trees built using several bootstrap samples coming from the learning samples and choosing randomly

at each node a subset of explanatory variables. Using a voting mechanism, the final result is obtained. The major benefits of Random Forest classifier are:

- It runs efficiently on large databases
- Maintains accuracy even when a large proportion of values in the given dataset are missing
- The model once generated can be reused again in future use and does not need re-computation of training model in future
- Random Forests attempts to mitigate the problems of high variance and high bias by averaging to find a natural balance between the two extremes.

Algorithm: Each tree is constructed using the following algorithm:

- Let the number of training cases be *N*, and the number of variables in the classifier be *M*.
- We are told the number *m* of input variables to be used to determine the decision at a node of the tree; *m* should be much less than *M*.
- Choose a training set for this tree by choosing *n* times with replacement from all *N* available training cases (i.e. take a bootstrap sample). Use the rest of the cases to estimate the error of the tree, by predicting their classes.
- For each node of the tree, randomly choose *m* variables on which to base the decision at that node. Calculate the best split based on these *m* variables in the training set.
- Each tree is fully grown and not pruned (as may be done in constructing a normal tree classifier).

DATASET DESCRIPTION

The dataset that we will be using in our project is obtained from YOOCHOOSE, who collected data from user visits over long period of a large European retailer's website over a period of 6 months. The dataset consists of two files namely clicks and buys. The clicks data contains information about every click the use makes on the site. Every entry in the clicks file contains the following information:

- Session id: It is the unique id used to identify all the clicks made in a particular session
- Timestamp: It includes the time and date when the click took place.
- Item ID: It is the unique identifier of the item
- Category: It identifies which category the item belongs to.

The buys data set represents purchases made by the user. Each entry in the buys file contains the following information:

- Session id: It is the unique id used to identify all the clicks made in a particular session
- Timestamp: It includes the time and date when the click took place.
- Item ID: It is the unique identifier of the item
- Price: It gives us the price at which the item was bought
- Quantity: It represents the number of items bought by the user

On combining all the clicks with respect to their session id and the buys data of the same session we obtain the entire user activity of that user. The test file only contains the click events with the attributes Session ID, Timestamp, Item ID, and Category. The clicks data file is almost 1.4 gigabytes in size while the test file is about 350 megabytes in size. The data set contains about 33 million clicks which can be clubbed to get more than 9 million sessions. There are close to 53,000 items that are listed in the dataset. Only 1.1 million sessions result in a buy event which is close to 5% of the total session, making the data significantly imbalanced. Also, the test file contains 8 million clicks and 2 million sessions.

6.2. WORKING OF PROJECT

The working of the Project is divided into two phases. Phase-I of the system involves data preprocessing and prediction whether the product is bought or not. Phase-II of the project is dependent upon the previous phase, which involves what product was bought given there was a buy in Phase-I.

Phase-I

1. Data Cleaning
    - The dataset consisting of 33 million entries contains null (NaN) values which become unusable, reducing the overall accuracy of the training model.
    - Data-Cleaning is the remove of null values either by Ignoring the data row, filling the missing values either using mean/median.
    - We have used 'Ignoring the data' row as a measure to counter the NaN values.
2. Data transformation
    - The attribute column 'Timestamp' contains the date, time and time zone in the standard 'dd:mm:yyyy;hh:mm:ss;zzz' format which needs to be transformed into usable format in day, month, year, hour, minute, seconds as separate attributes.
3. Under-Sampling
    - The training dataset is highly biased. It consists of 9 million sessions of which only 0.5 million are buy session.
    - Thus dataset is under sampled to have equal number of buys and non-buys training data. The final training data set consists of comparable number of buys and non buys sessions with total 1.1 million sessions.
    - In order to create a balanced data set, we tried subset selection. In this, we selected a block of K tuples of sessions not ending in purchases (where K is the number of sessions ending in purchases). However, as the data was sorted according the date, valuable information of trends in other months was discarded, which led to poor performance when using test data which was beyond the date boundaries of the block used for creating the model. Due to this, the model failed to predict accurately on all test samples. Instead, we go

for a random sampling approach where we take K samples from throughout the data such the training data consists of a wide variety of trends, compared to a block of data of adjacent tuples.

Code:

```
import pandas as pd
import numpy as np
data = pd.read_csv("C:/Users/Sony_owner/Desktop/BE Project/RecSys/Undersampledemoclicks.csv")
sample_size = len(data[data.Buys == 1]) # get the total count of low-frequency group(Buys: 1)
Buys_indices = data[data.Buys == 1].index
Buys_sample = data.loc[Buys_indices]
print Buys_sample
NotBuys_indices = data[data.Buys == 0].index
random_indices = np.random.choice(NotBuys_indices, sample_size, replace=False) # use the low-frequency group count to randomly sample from high-frequency group
NotBuys_sample = data.loc[random_indices]
print NotBuys_sample
# Merging all the low-frequency group sample and the new (randomly selected) high-frequency sample together
merged_sample = pd.concat([Buys_sample, NotBuys_sample], ignore_index=True)
print merged_sample
```

4. Data visualization:
    - The given dataset consists of 33 million entries. Hence it is not manually possible to examine the quality of data and its variation throughout.
    - Hence we Visualize the data using Python programming language to generate graphs that generate a series of graphs, giving us an estimate of the type and quality of data.
    - Accordingly steps are taken during preprocessing to clean and transform the data.

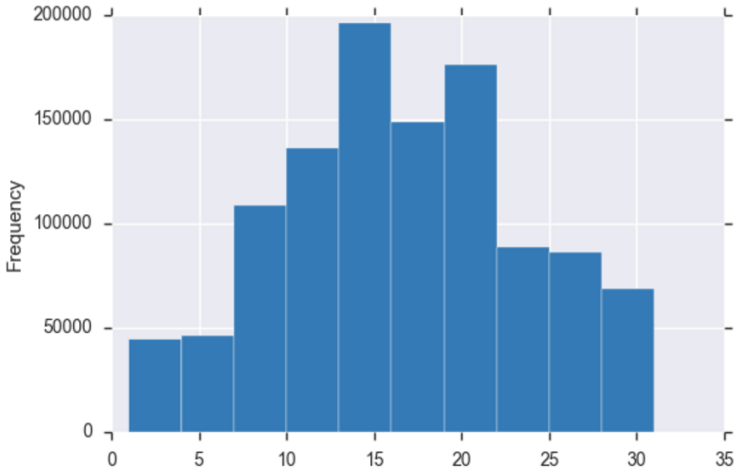

Figure 6.1. Frequency of Clicks vs Day of the Month

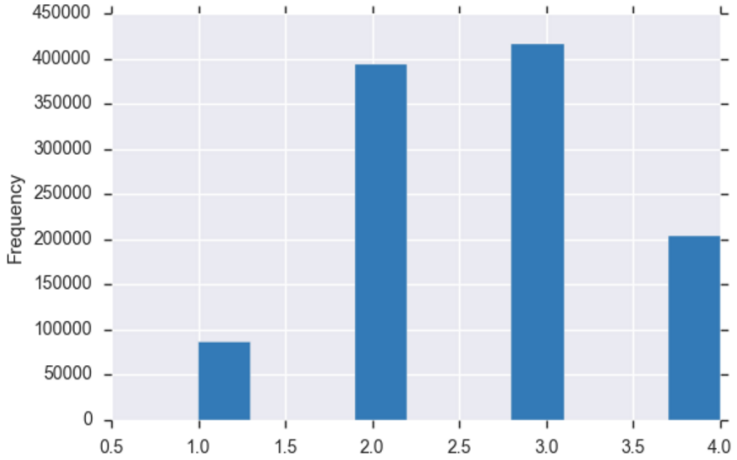

Figure 6.2. Frequency of Clicks vs Time of the Day

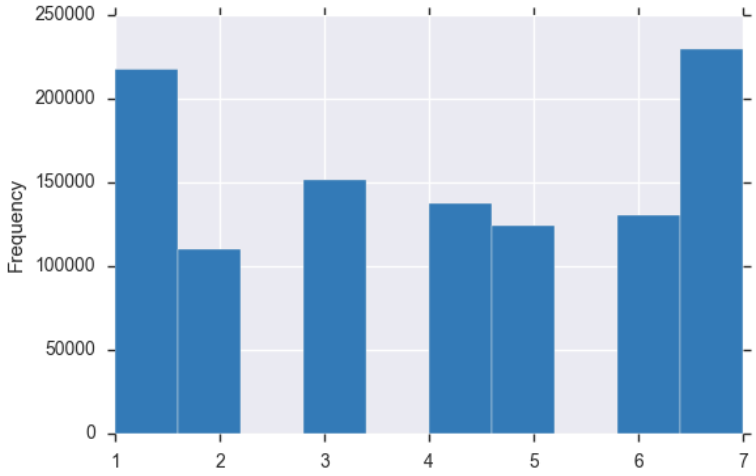

Figure. 6.3. Frequency of Clicks vs Day of the Week

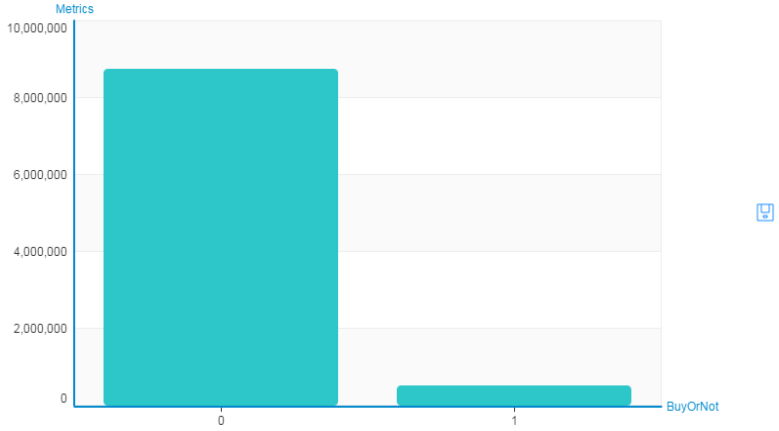

Figure. 6.4. Total Clicks vs BuyOrNot Events

5. Feature Extraction
    - The initial dataset consists of only 4 attributes SessionID, Timestamp, ItemID, Category.
    - After analyzing the data, we define multiple features using various aspects of a session in order to determine if the session is positive. We started with 14 features, but after running eliminating various features we ended up with a list of 10 features
    - Features Considered: Session Time, Average time between Clicks, Maximum Time Between Clicks, Day of the week, Month, Hour of Day, Period of Day, Number of clicks, Maximum clicks on one item, Number of distinct items, Average price of items, Maximum number of consecutive clicks on one item, Number of categories, Day of the Month.
    - Features Discarded: Hour of Day, Average price of items, Maximum number of consecutive clicks on one item, Number of categories.
    - We have used PCA and Stepwise backward selection to select the final 11 Features used for training
    - Useful data needs to be extracted from the given attributes, to facilitate the training. Useful attributes are the ones that influence the buying behavior of the user. The list of final features extracted are as follows:
        i. Clicks: It gives us the total number of clicks in a given session.
        ii. Month: It gives us the month in which the session was active. It is found by converting the timestamp into datetime format and then extracting the month.
        iii. Day_Of_Month: It gives us the day in the month during which the session was active. If the session spans across multiple days then we take the average value.
        iv. Day_Of_Week; It gives the day of the week in which the session took place. If the session spans across multiple days then we take the average value.
        v. Time_Of_Day: To calculate this attribute we took the average of the time at which the seesion was active and placed it into one of the

four classes created as morning, afternoon, evening, night but as numeric values.

vi. MaxClicksItems: It gives us the maximum number of clicks on any particular item in a given session.

vii. Distinct_Items: It gives the total number of distinct items the user browses in a session.

viii. Session_Time: It provides us with the total session duration, which can be obtained by subtracting the first and last timestamps of the respective sessions.

ix. MaxTimeBtwClicks: It gives us the maximum time between any two clicks. To calculate this attribute, we calculate the time between all clicks and take the maximum of the respective sessions.

x. AvgTimeBtwClicks: It gives us the avergage time between any two clicks. To calculate this attribute, we calculate the time between all clicks and take the average of the respective sessions.

6. Training:
   - The cleaned dataset, is passed onto the training algorithm, SVM in our case.
   - The model is constructed using Python programming language using the Graph-Lab API.
   - Training data uses the 11 extracted features to construct a training model.
   - *Model constructed in the previous step is saved for the testing phase.*

Code:

```
import graphlab as gl
data_train=gl.SFrame.read_csv('F:/BE_PROJECT/BE_PROJECT/yoochoose_data/final_data_22jan/final_feature(17feb).csv')
data_train=data_train.dropna()
data_test=gl.SFrame.read_csv('F:/Dakshil/BE_PROJECT/BE_PROJECT/yoochoose_data/final_data_22jan/final_features(1-10)test_nominal.csv')
```

```
model=gl.svm_classifier.create(data_train,target='BuyOrNot',
features=['Clicks','Month','Day_Of_Month','Day_Of_Week','Time_Of_Day','Max
ClicksItems','Distinct_Items','Session_Time','MaxTimeBtwClicks','AvgTimeBtwCli
cks'], max_iterations=50,class_weights='auto')
predictions=model.predict(data_test)
results=model.evaluate(data_test)
model.save('SVM_Model')
print results
```

7. Testing
   - Using the model constructed in the previous step, a python script is used to pass the separate test dataset, using the SVM algorithm that predicts the final 'Buy-or-Not' column of the dataset.
   - If a given session results in a buy, '1' is entered into the 'Buy/Not' column else '0' is entered.
   - Based upon the accuracy of multiple models constructed using RepTree, Multilayer Perceptron, SVM and Random Forest, we use SVM as it gives us the best accuracy from the above mentioned algorithms.

Code

```
import graphlab as gl
import csv
def svm():
        data =  gl.SFrame.read_csv('Phase1_data/features1-10.csv')
        model = gl.load_model('Phase1_codes/SVM_Model')
        predictions = model.predict(data)
        results = model.evaluate(data)
        predictions.save('Phase1_data/outputSVM.csv',format='csv')
```

8. Phase I Testing Results

Table 6.1. Phase 1 Test Results

Model	Mean absolute error	Accuracy
REPTree	0.2776	0.7224
MLP	0.2335	0.7665
SVM	0.202	0.7979
Random Forest	0.3109	0.689

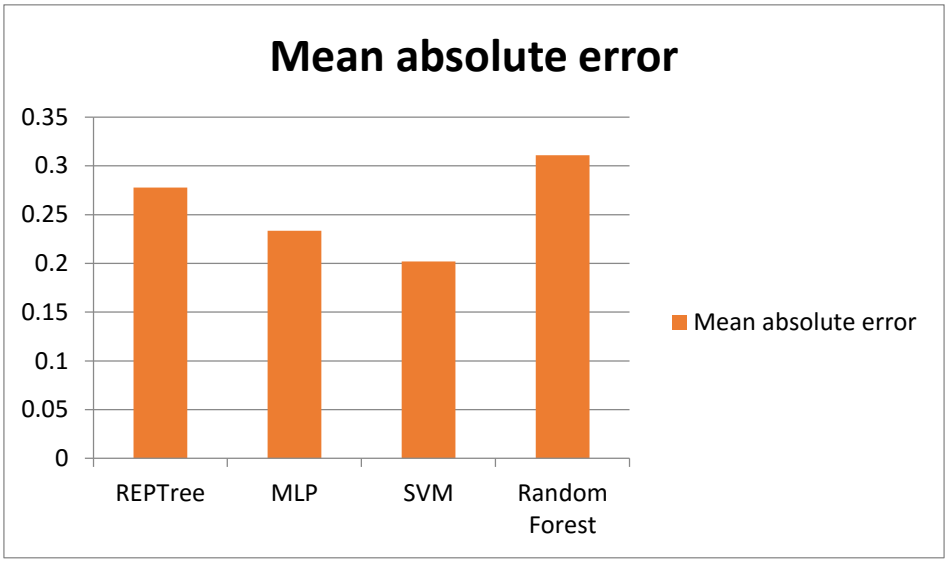

Figure 6.5. Mean Absolute Error Phase 1

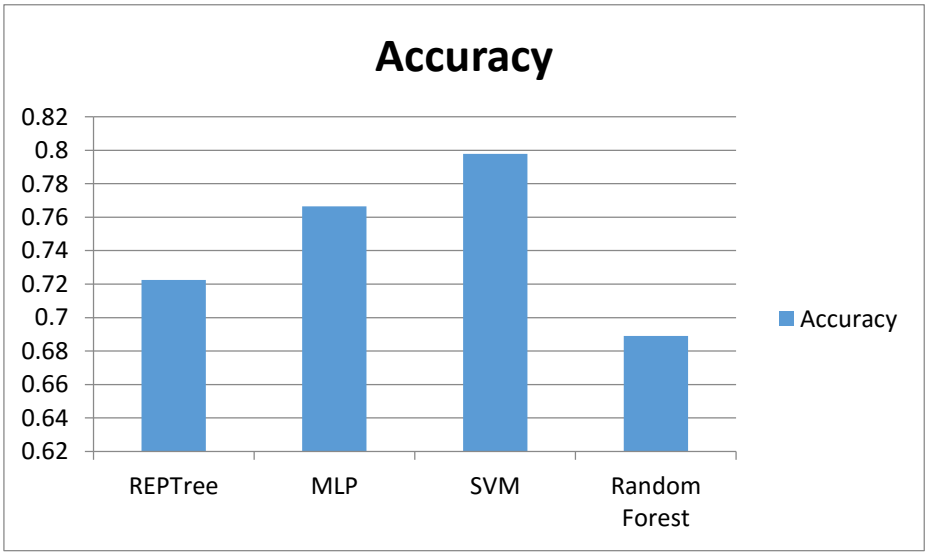

Figure. 6.6. Algorithm Accuracy Phase 1

Phase-II

Phase-II of the project is predicting what item was bought for each session that resulted in 'Buy=1' in the previous phase. It has the following subparts:

1. Feature Extraction

- The set of features in this phase provides various aspects on an item in a specific session in order to determine if the item is going to be bought. This set is comprised of features which refer to global information about the item based on all of the training data, as well as features that provide local information about an item's lifespan in a specific session.
- After analyzing the data, we define multiple features using various aspects of a session in order to determine if the session is positive. We started with 12 features, but after running eliminating various features we ended up with a list of 9 features.
- Features Considered: Weather the item was clicked first in the session, Weather the item was clicked last in the session, Number of appearances in a session, Relative no of clicks, Number of clicks on that item divided by the average number of clicks on items in that session, Time gap from first to last click on the item in that session, Item id, Maximum time gap between clicks on the item in the session, Time spent on that item in the session, Time gap from the first click on the item to the last click on it divided by the length of the session, Whether item appears more than one in session, Price of the item, Number of consecutive clicks in a session
- Features Discarded: Time gap from the first click on the item to the last click on it divided by the length of the session, whether item appears more than one in session, Price of the item, Number of consecutive clicks in a session.
- This phase requires different set of features which are extracted using techniques mentioned extracted and selected using Stepwise backward selection to select final 9 features listed as follows:
    i. Item_ID: To uniquely identify which particular item was bought in the session we need the item id.
    ii. FirstClick: It signifies if the item was the first to be clicked in the session. It has binary values as 0 or 1.

iii. LastClick: It signifies if the item was the last to be clicked in the session. It has binary values as 0 or 1.

iv. FirstLastClick: It gives us the time between the first and last click on that particular item.

v. TotalTimeItem: It gives us the total time spent on that particular item.its the sum of the time between two clicks of that particular item.

vi. MaxClicks: It gives us the maximum number of clicks on any item in that session.

vii. RelativeClicks: It gives us the number of clicks on that item divided by the maximum number of clicks on any item in that specific session.

viii. No_of_Appearance: It gives the number of times that particular item was clicked in that session.

ix. ClicksPerAvg: It is the ratio of the number of clicks on that particular item divided by the average number of clicks in that session.

2. Training:

- The cleaned dataset, is passed onto the training algorithms.
- The model is constructed using Python programming language using the Graph-Lab API.
- Training data uses the 9 extracted features to construct a training model.
- Model with the best results in the previous step is saved for the testing phase.

Code:

```
import graphlab as gl
import csv
data=gl.SFrame.read_csv('F:/Dakshil/BE_PROJECT/BE_PROJECT/yoochoose_data/final_codes_22jan/phase2/merge1.csv')
data=data.dropna()
data_train, data_test = data.random_split(0.8)
model=gl.random_forest_classifier.create(data_train,target='Buys',
features=['Session_ID','Item_ID','FirstClick','LastClick','TimeFirstLast','TotalTimeItem','MaxClicks','RelativeClicks','No_of_Appearances_x','Avg','ClicksPerAvg'])
```

```
predictions=model.predict(data_test)
results=model.evaluate(data_test)
model.save('Random_Forest_Model')
print results
predictions.save('outputTest.csv',format='csv')
w = csv.writer(open("resultsTest.csv", "w"))
for key, val in results.items():
w.writerow([key, val])
```

3. Testing:

   Code:

```
import graphlab as gl
import csv
def RandomForestfunc():
        data =  gl.SFrame.read_csv('Phase2_data/mergedFeaturesMod.csv')
        model = gl.load_model('Phase2_codes/Random_Forest_Model')
        predictions = model.predict(data)
        results = model.evaluate(data)
        print results
        predictions.save('Phase2_data/ItemsBought.csv',format='csv')
```

4. Phase II Testing Results

Table 6.2. Phase II Results

Model	Accuracy	Mean absolute error
SVM	0.7366	0.2633
Random Forest	0.7717	0.2282

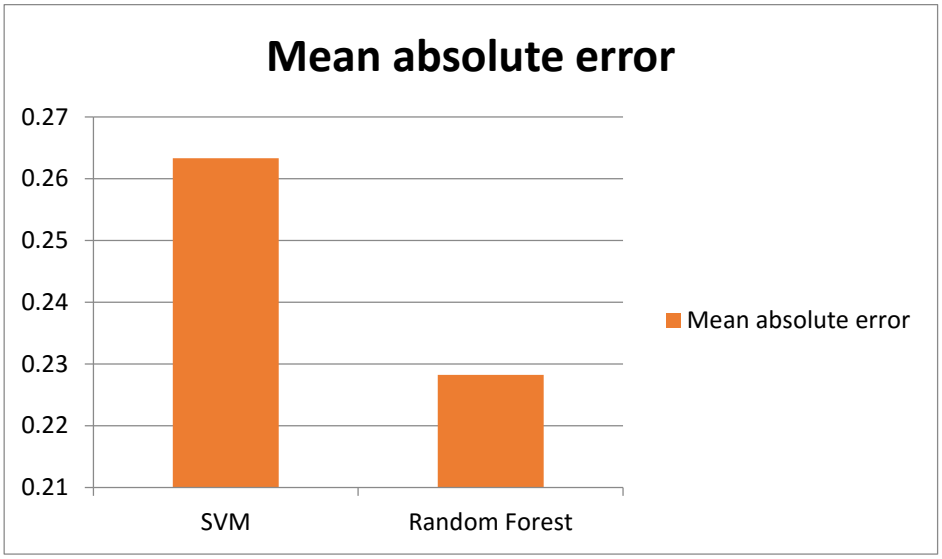

Figure 6.7. Mean Absolute Error Phase II

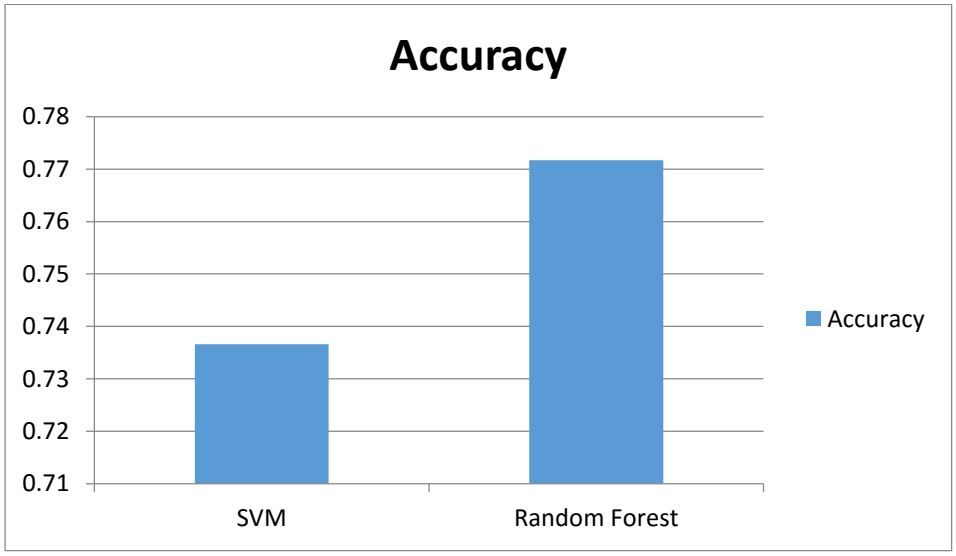

Figure 6.4. Algorithm Accuracy Phase II

CHAPTER 7: TESTING

7.1. TESTING PHASES

In this project we applied testing twice in each of the phases once while creating the model and once after the model was crated and new data was provided from the test data file.

7.1.1. Testing during Model Creation:

When the model was being created we created a split on the data to test immediately after training. We generally used a 80:20 split in which we used 80% of the data for training and 20% for testing. Since the test data in this step was similar to the data through which the model was built we had to implement a separate testing phase after model creation.

7.1.2. Testing after Model Creation:

In this project we also had a separate test data file that could be used to validate the final models. This data was in the initial clicks format, hence we had to first preprocess it and extract the features there after. After getting the features, the test data was applied to the models created in both the phases. This results obtained on the test data were almost similar to the results obtained while testing on split data, hence validating our model.

7.2. TYPE OF TESTING USED

7.2.1. Unit Testing

Each module of the project was created independently and is capable of functioning as a standalone program. Before the final integration testing, the modules were independently used to collect, process and evaluate data. The initial module was used to preprocess the data. The next modules included feature extraction phases, model generation and validation phases. Each feature extraction method was a separate code file which was independently tested before integrating it.

7.2.2. Integration Testing

Integration testing is used when different modules are combined and it has to be tested as a whole. It is used to expose faults in the interaction between the integrated units. It is performed after unit testing, and after the various modules are integrated. We used the bottom up approach to integration testing.

Integration testing was required by us because of the various modules present in our application were dependent. For example, the second phase of the project required the results of the first phase i.e. the sessions that were predicted to have buy events, to proceed further. Hence, to ensure that the data and results were being passed on correctly from one phase to another, we carried out integration testing.

7.2.3. Black Box Testing

Black box testing, also called Behavioral Testing, is a testing method in which the internal structure of the item being tested is not known to the tester. Incorrect/missing functions, data structure errors, performance errors and initialization and termination errors are identified by black box testing.

7.2.4. White Box Testing

White box testing us a testing method in which the internal structure of the item is known to the tester. The tester chooses inputs through the various paths in the code and determines the appropriate output. White box testing goes beyond the user interface and focuses on the details of a system.

CHAPTER 8: RESULTS

Input Data

Table 8.1 Clicks Input Data

Session_ID	Timestamp	Item_ID	Category
11	2014-04-03T10:44:35.672Z	214821275	0
11	2014-04-03T10:45:01.674Z	214821275	0
11	2014-04-03T10:45:29.873Z	214821371	0
11	2014-04-03T10:46:12.162Z	214821371	0
11	2014-04-03T10:46:57.355Z	214821371	0
11	2014-04-03T10:53:22.572Z	214717089	0

Table 8.2 Buys Input Data:

Session_ID	Timestamp	Item_ID	Price	Quantity
420374	2014-04-06T18:44:58.314Z	2.15E+08	12462	1
420374	2014-04-06T18:44:58.325Z	2.15E+08	10471	1
281626	2014-04-06T09:40:13.032Z	2.15E+08	1883	1
420368	2014-04-04T06:13:28.848Z	2.15E+08	6073	1
420368	2014-04-04T06:13:28.858Z	2.15E+08	2617	1
140806	2014-04-07T09:22:28.132Z	2.15E+08	523	1

Table 8.3 Phase 1 Extracted Features:

Index	Session_ID	Clicks	Month	Day_Of_Month	Day_Of_Week	Time_Of_Day	BuyOrNot	MaxClicksItems	Distinct_Items	Session_Time	MaxTimeBtwClicks	AvgTimeBtwClicks
0	11	12	4	3	4	2	1	3	9	13.062	6.420	1.187
1	12	2	4	2	3	2	1	2	1	2.990	2.990	2.99075
2	21	6	4	7	1	2	1	5	2	19.691	13.46	3.9382
3	33	16	4	6	7	3	1	8	4	4.688	1.592	0.312
4	46	6	4	3	4	2	1	6	1	59.766	58.062	11.953
5	87	27	4	7	1	1	1	3	20	31.998	12.339	1.230

Table 8.4. Phase II Extracted Features

Session_ID	Item_ID	FirstClick	LastClick	TimeFirstLast	TotalTimeItem	MaxClicks	RelativeClicks	Max_Click_Session	Appearances	Avg	ClicksPerAvg
70	214672841	0	0	0	0	2	0.5	2	1	1.5	0.67
70	214712272	0	0	0	0	2	0.5	2	1	1.5	0.67
70	214820252	0	1	1.1792	0	2	1	2	2	1.5	1.33
70	214827022	1	0	3.3679	0	2	1	2	2	1.5	1.33
75	214639841	0	0	0	0	2	0.5	2	1	1.14285	0.875
75	214652255	1	0	0.8775	0	2	1	2	2	1.14285	1.75
75	214652437	0	0	0	0	2	0.5	2	1	1.14285	0.875
75	214668575	0	0	0	0	2	0.5	2	1	1.14285	0.875
75	214705003	0	0	0	0	2	0.5	2	1	1.14285	0.875

Output

Table 8.5. Final output with BuyOrNot and Item Prediction

Session_ID	Item_ID	ItemPrediction
70	214672841	0
70	214712272	0
70	214820252	1
70	214827022	1
75	214639841	0
75	214652255	1
75	214652437	0
75	214668575	0
75	214705003	0

CHAPTER 9: CONCLUSION & FUTURE SCOPE

In the project 'Prediction and Visualization of User Tendency towards Purchases using Clickstream Data', we have successfully analyzed the dataset of an e-commerce website, to determine the overall user behavior during purchases. We have implemented a Training model to use the given dataset and learn the patterns within in to predict the new unseen samples of data. Whether the given product will be bought or not and if it is bought what is the corresponding item which will be purchased.

This project is not a user centric project focusing on the characteristics of user's browsing pattern and consequently providing appropriate recommendations. This model is in fact an analytical approach towards studying the click patterns that lead to a purchase, which provide the e-commerce with analytical and statistical model of the purchase behavior.

Given this model, it can be used accommodated as a backend system, which dynamically analysis each click and immediately provides a prediction probability of the purchase. Hence for every click the model can re-computes the possibility of a buy and the resulting item and add those items immediately into the cart, thereby saving the user's time and efforts.

Secondly this model provides the firms with analytical and clean visualized data. These visualizations can be used to design marketing strategies during key time periods to attract more customers.

Appendix

i. GraphLab: GraphLab Create, is an extensible machine learning framework that enables developers and data scientists to easily build and deploy intelligent applications and services at scale. It includes distributed data structures and rich libraries for data transformation and manipulation, scalable task-oriented machine learning toolkits for creating, evaluating, and improving machine learning models, data and model visualization for all aspects of development, and a client to define and deploy both distributed batch jobs to Dato Distributed™ as well as real-time machine learning services to Dato Predictive Services™. It is designed for end-to-end developer productivity, scale, and the variety and complexity of real-world data.

ii. Pandas: is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

iii. WEKA: Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.

iv. Pandas Profiling: Pandas profiling is an open source library used to automatically plot the distribution of all variables, discover correlations and highlight things that seem fishy like missing data, skewness, excessive zeros etc. We can then convert this file into a required format like an html webpage. It can be used for exploratory data analysis.

Literature Cited

[1] Boroujerdi et al. “A Study on Prediction of User's Tendency Toward Purchases in Websites Based on Behavior Models”

[2] Panagiotos et al. “Algorithms for Clustering Clickstream Data”, Elsevier, 2009

[3] Zhang et al. " Modified Logistic Regression: An Approximation to SVM and Its Applications in Large-Scale Text Categorization”

[4] Tarun Rao et al. “A Hybrid Random Forest based Support Vector Machine Classification Supplemented by Boosting”

[5] Andreas G.K. Janecek et al. “On the Relationship Between Feature Selection and Classification Accuracy”, University of Vienna

[6] Isabelle Guyon “An Introduction to Variable and Feature Selection”

[7] Jiawen Han, Micheline Kamber “Data Mining, Concepts and Techniques”

Publications

Dakshil Shah, Samkeet Shah, Jamshed Shapoorjee Kiran Bhowmick, "Proposed Model for Prediction and Visualization of User Tendency towards Purchases using Clickstream Data", IJIACS- ISSN 2347-8616, Volume 4, Issue 11, November 2015

Acknowledgements

This project consumed huge amount of work, research and dedication. Still, implementation would not have been possible if we did not have a support of our college and teachers. We would like to extend our sincere gratitude to all of them. We are highly indebted to our project guide Prof. Kiran Bhowmick for her guidance and constant supervision as well as for providing necessary information regarding the project & also for her support in completing the project.

We would also like to express our gratitude towards our Principal Dr. Hari Vasudevan and our Head of Department, Dr. Narendra M. Shekokar for providing us with all the facilities to complete our project. Lastly, we would like to thank our family and friends for supporting us in our project.

We would like to acknowledge GraphLab Create, Pandas python library for creating the models and supporting the coding, ACM Recsys challenge and Yoochoose for providing the initial dataset.