Minor Project Report on

Click-Through Rate Prediction

Bhuvan M S (12IT16)

S Ashish Bharadwaj (12IT63)

Siddharth Jain (12IT78)

Vinay Rao D (12IT94)

Under the Guidance of,

Mr. Dinesh Naik

Department of Information Technology, NITK Surathkal

Date of Submission: 06 April 2015

in partial fulfillment for the award of the degree

of

Bachelor of Technology

In

Information Technology

At

Department of Information Technology

National Institute of Technology Karnataka, Surathkal

April 2015

Department of Information Technology, NITK Surathkal
Minor Project
End Semester Evaluation Report (April 2015)

Course Code : IT 399

Course Title: Minor Project

Project Title: *Click-Through Rate Prediction*

Project Group:

Name of the Student	Register No.	Signature with Date
Bhuvan M S	12IT16	
S Ashish Bharadwaj	12IT63	
Siddharth Jain	12IT78	
Vinay Rao D	12IT94	

Place:NITK, Surathkal

Date:06 April 2016 *(Name and Signature of Minor Project Guide)*

Abstract

This study involves the prediction of ad click-through rate which is the most important business component of displaying advertisements on a website. The goal is to determine the most accurate machine learning model for the problem of binary classification to predict whether an ad will be clicked or not based on various features. The dataset contains 10 days of Avazu data to build and test prediction models. Our research involves experimentation with different preprocessing techniques and machine learning models for efficient classification. The big data needs to be handled using non-conventional techniques as it doesnt fit into memory. The results would be evaluated based on the log loss obtained from the output of different algorithms. Research work needs to be done to deal with large and high dimensional datasets with more efficient machine learning models.

Keywords: *Machine Learning; Big Data; Vowpal Wabbit; Logistic Regression; Neural Networks; Binary Classification*

Contents

1 **Introduction** 1

2 **Literature Survey** 2

2.1 Related Work . . . 2

2.2 Outcome of Literature Survey . . . 3

2.3 Problem Statement . . . 4

2.4 Objectives . . . 4

3 **Methodology** 5

3.1 Work Flow . . . 5

3.2 Data description . . . 6

3.3 Scikit learn . . . 7

3.3.1 Overview . . . 7

3.3.2 Preprocessing . . . 7

3.3.3 Training Strategy . . . 8

3.4 Vowpal Wabbit: The Online Learning Framework . . . 9

3.4.1 Overview . . . 9

3.4.2 Preprocessing . . . 9

3.4.3 Training Strategy . . . 10

3.5 Machine learning algorithms . . . 10

3.5.1 Logistic Regression . . . 11

3.5.2 Support Vector Machines: SVM . . . 12

3.5.3 Artificial Neural Network . . . 13

3.5.4 Gradient Descent and Stochastic Gradient Descent . . . 13

3.6 Evaluation metrics . . . 14

3.6.1 Training time . . . 14

3.6.2 Log Loss . . . 14

3.6.3 Receiver Operating Characteristic Curve: ROC . . . 15

3.6.4 Precision and Recall Curve . . . 16

4 **Work Done** 17

4.1 System Architecture . . . 17

4.2 Experimental setup 17
4.3 Implementation Details 17
4.3.1 Scikit Learn 17
4.3.2 Vowpal Wabbit 18

5 Results and Analysis **20**
5.1 Logistic Regression using scikit-learn 20
5.1.1 Results 20
5.1.2 ROC Curve 20
5.2 Logistic Regression using Vowpal Wabbit 21
5.2.1 ROC Curve 21
5.2.2 PR Curve 22
5.2.3 Results 23
5.3 Neural Networks on Vowpal Wabbit 23
5.3.1 ROC Curve 23
5.3.2 PR Curve 24
5.3.3 Results 24
5.4 Comparison of Results 25

6 Conclusion & Future Work **26**
6.1 Novelty of the work 26
6.2 Future work 27

List of Figures

3.1 Flowchart representing the Methodology work flow. 5

3.2 Structure of the dataset 6

5.1 ROC Curve for Logistic Regression on scikit-learn 20

5.2 ROC Curve for Logistic Regression without L2 Regularization 21

5.3 ROC Curve for Logistic Regression with L2 Regularization 21

5.4 PR Curve for Logistic Regression without L2 Regularization 22

5.5 PR Curve for Logistic Regression with L2 Regularization 22

5.6 ROC Curve for Neural Networks 23

5.7 PR Curve for Neural Networks 24

List of Tables

1 Division of training and test data 7

2 System Architecture 17

3 Experimental Setup 17

4 Results of Logistic Regression with scikit-learn 20

5 Comparision of Logistic Regression with and without L2 Regularization . 23

6 Results for Neural Networks 24

7 Comparison of Results 25

1 Introduction

Click-through rate (CTR) is a way of measuring the success of an online advertising campaign for a particular website as well as the effectiveness of an email campaign by the number of users that clicked on a specific link. The purpose of click-through rates is to measure the ratio of clicks to impressions of an online ad or email marketing campaign. Generally the higher the CTR the more effective the marketing campaign has been at bringing people to a website. Most commercial websites are designed to elicit some sort of action, whether it be to buy a book, read a news article, watch a music video, or search for a flight. People generally don't visit a website with the intention of viewing advertisements, just as people rarely watch TV with the purpose of consuming commercials.

Marketers want to know the reaction of the web visitor. Under current technology, it is nearly impossible to fully quantify the emotional reaction to the site and the effect of that site on the firm's brand. One piece of information that is easy to acquire, however, is the click-through rate. The click-through rate measures the proportion of visitors who initiated action with respect to an advertisement that redirected them to another page where they might purchase an item or learn more about a product or service. Here "click through rate" has been used on the advertisement (or link) because this is the generally used term, although other interactions are possible. The click-through rate is the number of times a click is made on the advertisement divided by the total impressions (the number of times an advertisement was served):

$$ClickThroughRate = numberOfClickThroughs/numberOfImpressions$$

2 Literature Survey

2.1 Related Work

The work in the field of click-through rate (CTR) prediction has been of growing importance. This problem is the centroid of multibillion dollar online advertising industry. Proper analysis of CTR helps in deciding the price per click and order of impressions and hence is very important for commercial search engine based companies. CTR can give us a base number of visitors who can be potentially converted to customer and hence needless to say more about its importance in this business dominated world.

H. Brendan McMahan et al. [9] provides case studies on improvements done in supervised learning using FTRL (Follow the Regularized Leader)-Proximal online learning algorithms in this field. He also explores hurdles in this approach like problems regarding saving memory, visualization of performance etc. The main goal was to establish the close relationship between the theoretical aspects of the solution used with its practical implementation and challenges faced while doing so. The usage of internet has grown exponentially and hence for learning at such a massive scale, basic linear models like logistic regression are advantageous. It was seen that use of online gradient descent was very effective. While giving excellent accuracy, it used very less resources for computation. In results it was seen that the results using FTRL algorithm were way better than the outcome using RDA (Regularized Dual Averaging) method. But with this, he has also mentioned some promising approaches which failed to show good results like feature hashing, dropout, feature bagging and feature vector normalization. Feature hashing [1] is used to reduce RAM cost, bust in this case reduction led to a significant loss in accuracy. In dropout, some features are randomly removed and the resulting vector was scaled accordingly. This is basically regularization in conjunction with bagging. In general, it gives better accuracy but for predictive analysis of CTR in experiments done in [9] it was found that it depreciated the quality of result. Even feature bagging resulted in reduced accuracy for prediction. Same was the case with normalization.

Steffen Rendle et al.[10] uses Factorization Machines (FM) to solve the problem of CTR prediction. FM combines the the flexibility of feature engineering and benefits of factorization models. This makes FM a very optimal choice for prediction problems where variable range over a wide domain. The problem of CTR was treated as a weighted

regression problem with CTR being the targeted variable. The system made use of predictor variables like user ID, ad ID, query ID and position of ad with some other attributes. As data varies with this variables, 3 different models were developed like ID model, this was based on user ID and query ID and the attribute model based on attributes and a mixed model. The best scores were seen with the mixed model. Efficient use of FMs for prediction problem with features varying over multiple domains was shown.

Xingxing Wang et al.[11] discusses CTR with hybrid models. First, they implemented Online Bayesian Probit Regression (BPR), Support Vector Machine and Latent Factor model and then they propose a rank-based ensemble method which showed improvement in results. The features used were classified into two categories: Original features and Synthetic features. Original feature set is set of continuous and discrete features. For instance, unique ID for each ad, query can be considered as original features. Synthetic features are the features when two original features are combined and used. After this, individual methods were implemented. BPR was used to predict CTR of sponsored search advertising in Bing search engine. With SVM, the problem was treated as of binary classification. The click percentage of total impressions was calculated by predicting whether a user will click the advertisement. With MLE, CTR was predicted for all features and only the original discrete features were considered. To put the training set to an efficient use, maximum likelihood estimation methods (MLE) were used. Individually, the result was best with MLE (0.7924) and with the blending model result increased to 0.8034.

2.2 Outcome of Literature Survey

As it can be seen, a lot of work has been done in this field but even then there is a lot of scope for improvement. As internet is a growing entity, the problem has grown with big data making computations costly. As feature engineering is totally up to the programmer, how features are considered also affects results and a lot of possibilities have been left to explore. Different and better models with some set of features can give better results than those seen till now and can be used for more accurate CTR prediction.

It is apparent from the Literature Survey that feature selection involving manipulation of various features that are available with the dataset needs to be considered and improved upon to build better prediction models for the click through rate. Big data and live

streaming data have not been handled in much detail in previous research and hence newer technologies to tackle these problems have to be explored as they are of utmost importance in today's modern data driven world. A comparison of various tools and algorithms based on different evaluation metrics has to be done in an organized way so that it progresses further research with ease.

2.3 Problem Statement

In online advertising, click-through rate (CTR) is a very important metric for evaluating ad performance. As a result, click prediction systems are essential and widely used for sponsored search and real-time bidding. 10 days worth of click-through data provided by Avazu has been obtained. The goal of this project is to find an optimal and effective strategy that beats standard classification algorithms in building a successful click-through rate prediction model.

2.4 Objectives

The main research objectives are to find better ways to handle the huge amounts of data that is available and thereby suggest better methods to analyze it. This is done by comparison of results obtained by various machine learning tools and techniques. Different models are generated to predict the click-through rate in order to achieve a higher success percentage.

1. Build a Binary Classification model to predict the click through rate using all the features available in the dataset.

2. In order to accommodate Big Data and Streaming data, implementation in suitable frameworks.

3. Determine the algorithm which performs the best according to chosen evaluation metrics.

3 Methodology

3.1 Work Flow

The research flow is undertaken to cover all the crucial steps to build a flexible and scalable prediction model. The flowchart representing the methodology is given in the figure 3.1.

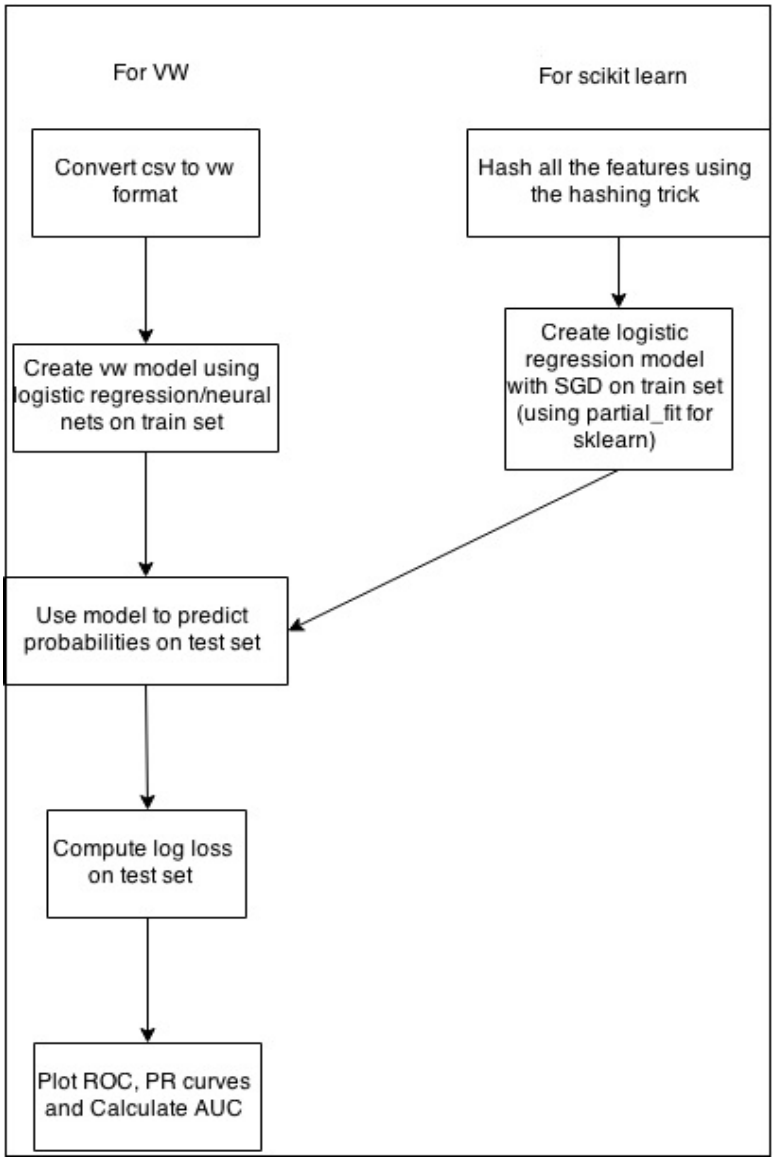

Figure 3.1: Flowchart representing the Methodology work flow.

3.2 Data description

The dataset is a single file which contains 10 days of click-through data, ordered chronologically. Non-clicks and clicks have already been subsampled according to different strategies. The size of the compressed file is 1.1 GB and when uncompressed, it becomes 6.3 GB. Each row in the dataset contains the following features :

```
'data.frame':    40428967 obs. of  24 variables:
$ id               : chr  "10000094181510942?3" "10(
$ click            : Factor w/ 2 levels "0","1": 1 1
$ hour             : num  14102100 14102100 1410210(
$ C1               : Factor w/ 7 levels "1001","100?
$ banner_pos       : Factor w/ 7 levels "0","1","2"
$ site_id          : Factor w/ 4737 levels "000aa1a4
$ site_domain      : Factor w/ 7745 levels "000129f1
$ site_category    : Factor w/ 26 levels "0569f928"
$ app_id           : Factor w/ 8552 levels "000d629?
$ app_domain       : Factor w/ 559 levels "001b87ae'
$ app_category     : Factor w/ 36 levels "07d7df22"
$ device_id        : Factor w/ 2686408 levels "0000(
$ device_ip        : Factor w/ 6729486 levels "0000(
$ device_model     : Factor w/ 8251 levels "0009742?
$ device_type      : Factor w/ 5 levels "0","1","2"
$ device_conn_type: Factor w/ 4 levels "0","2","3"
$ C14              : Factor w/ 2626 levels "10289",'
$ C15              : Factor w/ 8 levels "1024","120'
$ C16              : Factor w/ 9 levels "1024","20"
$ C17              : Factor w/ 435 levels "1008","1(
$ C18              : Factor w/ 4 levels "0","1","2"
$ C19              : Factor w/ 68 levels "1059","10(
$ C20              : Factor w/ 172 levels "-1","100(
$ C21              : Factor w/ 60 levels "1","100",'
```

Figure 3.2: Structure of the dataset

The number of rows in the dataset is 40428968. All the features except 'hour' are categorical variables. Structure of the data is analyzed using pandas. But as the dataset is huge and the whole dataset cannot be loaded, chunks of data is loaded and the structure

of the data is analyzed. The structure of the dataset is as shown in the figure 3.2.

As seen in the figure, 'click' feature has two factors or levels, 0 or 1 which means non-click/click. Other features have many levels as seen in the diagram. The number of clicks are about 20% of the dataset.

Training	**Test**	**Total**
32377421	8051546	40428967

Table 1: Division of training and test data

The dataset is divided into train and test sets. Train set contains 8 days of click-through data out of the 10 days and the remaining is used as the test set. The above table shows the number of samples in the train and test dataset.

As the dataset is huge, it can't be loaded in RAM fully. Hence, different methods are used to train model on the datasets. Using scikit-learn, the data can be fit partially iteratively. Online learning is also applicable here. Both of these are described in the sections below.

3.3 Scikit learn

3.3.1 Overview

Scikit-learn is an open source machine learning library for the Python programming language.[2] It features various classification, regression and clustering algorithms including support vector machines, logistic regression, naive Bayes, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. The scikit-learn project started as scikits.learn, a Google Summer of Code project by David Cournapeau.

3.3.2 Preprocessing

Scikit Learn is used for performing machine learning processing using python. It is a simple and efficient tool for data mining and data analysis. Before applying the algorithm, data is preprocessed so as to improve the performance of algorithm on the dataset. With Scikit Learn, feature hashing is used. It is a space efficient way of vectorizing features. A hash function is applied to features and the hash value is directly used as indices to

vector or a matrix. This method actually makes the data representation space efficient but actually uses more space considering that the reference dictionary also has to be stored. Hence the performance of feature hashing depends on the size of the training set.

3.3.3 Training Strategy

Scikit learn is built for python using numpy and scipy libraries as foundation. It loads all the data into memory prior to training process of any machine learning algorithm.

But for some applications the amount of examples, features (or both) and/or the speed at which they need to be processed are challenging for traditional approaches. In these cases scikit-learn has a number of options you can consider to make your system scale. In order to achieve this, following is a sketch of system design: of system design:

1. A way to stream instances: The data obtained for click through predictions comes in a streaming fashion. It can be loaded instance by instance into the scikit learn framework

2. A way to extract features from instances: Feature hashing trick can be used for this purpose. **Feature hashing** or **hashing trick** is a fast and space-efficient way of vectorizing features, i.e. turning arbitrary features into indices in a vector or matrix. It works by applying a hash function to the features and using their hash values as indices directly, rather than looking the indices up in an associative array. It then maps all the hashed features into a lower dimensional hash-space. Weinberger et al.,[12] demonstrated that even with a very large number of tasks and features, all mapped into a joint lower dimensional hash-space, one can obtain impressive classification results with finite memory guarantee.

3. An incremental algorithm: Machine learning algorithms where **Stochastic Gradient Descent (SGD)** is used instead of batch Gradient Descent (GD) are needed.

   Batch gradient descent computes the gradient using the whole dataset. Whereas Stochastic gradient descent (SGD) computes the gradient using a single sample. Hence GD requires more memory and takes more time as it needs to process all the instances. Therefore it is not suitable for large datasets like ours. SGD on the other hand, doesnot require the dataset to be loaded onto memory as it acts on single instance. Hence SGD can handle large datasets which cannot be loaded onto

RAM. Another benefit of SGD is that it's computationally a whole lot faster. this computational advantage is leveraged by performing many more iterations of SGD, making many more steps than conventional batch gradient descent. This usually results in a model that is very close to that which would be found via batch gradient descent, or better.

3.4 Vowpal Wabbit: The Online Learning Framework

3.4.1 Overview

Vowpal Wabbit (aka VW) [8] is an open source fast out-of-core learning system library and program developed originally at Yahoo! Research, and currently at Microsoft Research. It was started and is led by John Langford. Vowpal Wabbit is notable as an efficient scalable implementation of online machine learning and support for a number of machine learning reductions, importance weighting, and a selection of different loss functions and optimization algorithms.

3.4.2 Preprocessing

Vowpal Wabbit is an online learning method. Because of the size of data, this is very helpful. The input file for VW must be in a specific format which is as follows:
The data being fed as input to VW should have one instance or example per line. In that line also, the order of information provided is fixed. Each line is treated as one instance. Each instance is supposed to start with label,the real number that is to be predicted. If label is omitted in an example, training won't be performed on that example but VW will still compute prediction. After label, comes importance weight. It takes non negative values and is used to tell VW the relative importance of a particular example over other examples.If this is not specified, importance is taken as 1. Next comes in the line, the identifier for that example or a tag. It doesn't have to be unique.Empty string is the default value for this parameter. If you provide a tag without a weight you need to disambiguate: either make the tag touch the | (no trailing spaces) or mark it with a leading single-quote '. If you don't provide a tag, you need to have a space before the | (Pipeline representation will be explained in example ahead). After tag, namespace is mentioned which identifies source of information for that example. It can be followed by

an optional float value. The float value acts as a global scaling of all the values of the features in this namespace.Namespace should not have space between the separator "|", otherwise it can be interpreted as a feature.

At the end of example, features are mentioned which are a sequence of white space separated strings, each of which is optionally followed by a float. The string is considered as feature and value as the feature value for that example. If a feature is omitted, then it's default value is considered as 0.

Example for this format is given below:
1 1.0 |MetricFeatures:4.56 height:1.3 length:2.0 |Says black with yellow skin |OtherFeatures NumberOfLegs:4.0 HasStripes

3.4.3 Training Strategy

This is an online learning framework, meaning it can learn on streaming data by processing each instance at once as it enters the framework. It does not need to load all data into memory at all. Hence it can support data of any size. On the fly generation of feature interactions (quadratic and cubic) helps to handle the streaming raw data as conversion to feature vector happens automatically. It supports Multiple supervised learning algorithms with multiple loss functions. It has multiple optimization algorithms like SGD (Stochastic Gradient Descent), BFGS (BroydenFletcherGoldfarbShanno algorithms) etc. It is known for its scalability. Vowpal wabbit has been used to learn a tera-feature (1012) data-set on 1000 nodes in one hour. Its scalability is aided by several factors:

1. Out-of-core online learning: no need to load all data into memory
2. The hashing trick: feature identities are converted to a weight index via a hash
3. Exploiting multi-core CPUs: parsing of input and learning are done in separate threads.
4. Compiled C++ code

3.5 Machine learning algorithms

Machine learning is a scientific discipline that explores the construction and study of algorithms that can learn from data. Such algorithms operate by building a model from

example inputs and using that to make predictions or decisions, rather than following strictly static program instructions.

Several machine learning algorithms are present. But not all work equally well for all datasets. Hence our research needs to chose a set of machine learning algorithms which are hypothesized to perform well for the type of data being worked on.

3.5.1 Logistic Regression

Logistic Regression [2] is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a binary variable. It is a type of probabilistic classification model. Values of parameters like regularization parameter can be tuned to get optimal (both accuracy and efficiency) results.

Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables, which are usually (but not necessarily) continuous, by using probability scores as the predicted values of the dependent variable.

The previous research articles state that logistic regression outperforms other machine learning algorithms while predicting click through rates. Despite the probabilistic framework of logistic regression, all that logistic regression assumes is that there is one smooth linear decision boundary. It finds that linear decision boundary by making assumptions that the P(Y—X) of some form, like the inverse logit function applied to a weighted sum of our features. Then it finds the weights by a maximum likelihood approach. The decision boundary it creates is a linear* decision boundary that can be of any direction. So if there is data where the decision boundary is not parallel to the axes, then logistic regression picks it out pretty well, whereas other algorithms like decision tree will have problems. Hence this algorithm is chosen for experimentation. Because the output value from logistic regression is to be in the range [0,1], sigmoid function is used.

$$g(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

The variable x can be viewed as a linear function of the features.

$$x = \beta_0 + \beta_1 t_1 ... \tag{2}$$

and hence, the sigmoid function can be re written as

$$g(x) = \frac{1}{1 + e^{-(\beta_0 + \sum \beta_i t_i)}} \tag{3}$$

L2 regularized logistic regression is a rotaionally invariant algorithm which requires a sample size that grows linearly in the number of irrelevant features. The loss function used with L2 is least square errors function.This is not very robust but gives a stable solution. To explain simply, with L2,the sum of squares of weights is considered. It is computationally more efficient than L1 because it has unique solution.

3.5.2 Support Vector Machines: SVM

Support vector machines [7](SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

If the number of features is much greater than the number of samples, the method is likely to give poor performance. SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.In the binary case, the probabilities are calibrated using Platt scaling: logistic regression on the SVMs scores, fit by an additional cross-validation on the training data.Needless to say, the cross-validation involved in Platt scaling is an expensive operation for large datasets. In addition, the probability estimates may be inconsistent with the scores. Platts method is also known to have theoretical issues. libsvm also has scalability issues and is thus not suitable for a massive dataset as ours.Thus, due to the above reasons Support Vector Machines have been left out of our analysis.

3.5.3 Artificial Neural Network

In machine learning and cognitive science, artificial neural networks (ANNs) [4] are a family of statistical learning algorithms inspired by biological neural networks (the central nervous systems of animals, in particular the brain) and are used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown. Artificial neural networks are generally presented as systems of interconnected "neurons" which can compute values from inputs, and are capable of machine learning as well as pattern recognition thanks to their adaptive nature.

An ANN is typically defined by three types of parameters:

1. The interconnection pattern between the different layers of neurons
2. The learning process for updating the weights of the interconnections
3. The activation function that converts a neuron's weighted input to its output activation.

Advantage of ANN is that it is a nonlinear model that is easy to use and understand compared to statistical methods. ANN is non-parametric model while most of statistical methods are parametric model that need higher background of statistic. ANN with Back propagation (BP) learning algorithm is widely used in solving various classification and forecasting problems. Even though BP convergence is slow but it is guaranteed. Hence ANN was chosen to predict the click through rate where automatic adaptation is necessary due to large dataset with biased distribution.

3.5.4 Gradient Descent and Stochastic Gradient Descent

To minimize the cost function, gradient descent algorithm is used. Basically, it works to find the local minimum of the function. First a guess is made for solution and gradient is found at that point. Then the solution is stepped in negative direction of the grasdient and repeat the whole process till minimum is achieved. Algorithm converges where the gradient is zero. If while trying to minimize a function g(x) given initial value of x as x0, the gradient is taken and then algorithm tries to lower the function value:

$$x_{k+1} = x_k - \lambda \bigtriangledown g(x_k) \tag{4}$$

where lambda is the parameter which controls how large of a step that is taken. Proper value of lambda ensures that algorithm converges.

A variant of this algorithm is known as Stochastic Gradient Descent(SGD). While in Gradient Descent, you have to run through all the samples in your training set to do a single update for a parameter in a particular iteration, in SGD, on the other hand, you use only one training sample from your training set to do the update for a parameter in a particular iteration.Thus, if the number of training samples are large, then using gradient descent may take too long because in every iteration when you are updating the values of the parameters, you are running through the complete training set. On the other hand, using SGD will be faster because only one training sample is used and it starts improving itself right away from the first sample. With SGD, its a simple tradeoff of accuracy for speed. It converges faster than Gradient descent but the error function is not as well minimized as in the case of gradient descent.

3.6 Evaluation metrics

The following evaluation metrics are calculated for each of the algorithms implemented in each framework. This enables us to compare the performance of different algorithms in different frameworks.

3.6.1 Training time

It is the measure of time taken by the machine learning algorithms to learn on the training set. During the training the machine learning algorithm iterate over all the instance in the training set and learn the implicit patterns and structure of the data or the features present in the data. It is an important measure because the training is the more computationally expensive art of any machine learning algorithm. Hence to make an efficient prediction model, low training time is needed in order to run train on huge datasets.

3.6.2 Log Loss

Logarithmic Loss or Log loss [3] is defined as below for a multi class classification problem with M classes.

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j}\, log(p_{i,j}\,) \tag{5}$$

where N is the number of observations, M is the number of class labels, log is the natural logarithm, $y_{i,j}$ is 1 if observation i is in class j and 0 otherwise, and $p_{i,j}$ is the predicted probability that observation i is in class j.

Since in our scenario, it is a problem of binary classification, i.e. M has only two classes, the equation can be simplified as below.

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}(y_i log(p_i) + (1 - y_i) log(1 - p_i)) \tag{6}$$

Here, y_i takes values of 0 or 1 depending on whether the observation belongs to the first class or the second. In the case of click through rate prediction, there are only two classes namely clicked(1) and not-clicked(0) and hence it is a binary classification problem. Log loss is a widely used metric when it comes to click through rate prediction as it gives correct and most meaningful estimates of prediction accuracy of the model under consideration.

3.6.3 Receiver Operating Characteristic Curve: ROC

In statistics, a receiver operating characteristic (ROC) [6], or ROC curve, is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. The curve is created by plotting the true positive rate against the false positive rate at various threshold settings. ROC analysis provides tools to select possibly optimal models and to discard suboptimal ones independently from (and prior to specifying) the cost context or the class distribution. ROC analysis is related in a direct and natural way to cost/benefit analysis of diagnostic decision making.

The Area Under the Curve (AUC) is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. This measure is used to evaluate different binary classifiers. Higher the ROC AUC the better is the classifier.

3.6.4 Precision and Recall Curve

In pattern recognition and information retrieval with binary classification, precision (also called positive predictive value) is the fraction of retrieved instances that are relevant, while recall (also known as sensitivity) is the fraction of relevant instances that are retrieved. Both precision and recall are therefore based on an understanding and measure of relevance.

1. Precision (P): is defined as the fraction of ground truth positives among all the examples that are predicted to be positive.

$$Precision = TP/(TP + FP) \tag{7}$$

   Where, TP, FP stand for True Positives and False Positives respectively

2. Recall (R): is defined as the fraction of ground truth positives that are predicted to be positives.

$$Recall = TP/(TP + FN) \tag{8}$$

   Where, FN denotes False Negatives.

3. Precision-Recall Curve [5]: The tradeoff between the precision and recall can be studied by looking at the plot of how precision changes as a function of recall. This plot is known as the Precision-Recall (PR) curve. The precision is plotted on the y-axis and recall on x-axis. The area under precision recall curve is a cost-effective metric for evaluation of classifiers. High value of area under the PR curve indicates better performance. Area under the PR curve is preferred way of studying the performance of classifiers in skewed (i.e. imbalance between positives and negatives) datasets such as the one considered in this work [5].

4 Work Done

4.1 System Architecture

Operating System	Microsoft Windows 8 Pro and Ubuntu 14.04 LTS
Processor	Intel(R) Core(TM) i7-3610QM CPU @2.30GHz
RAM	8.00 GB (7.89 GB usuable)
System Type	64-bit OS, x64-based processor

Table 2: System Architecture

4.2 Experimental setup

Tool	**Version**
python	2.7.9
pandas	0.16.0
scikit-learn	0.16.0
matplotlib	1.4.3
NumPy	1.9.2
Vowpal Wabbit	7.10

Table 3: Experimental Setup

4.3 Implementation Details

4.3.1 Scikit Learn

The code snippet for our program is as follows :

```
# Train classifier
clf = SGDClassifier(loss="log", n_iter=1)
train = pd.read_csv("../Data/train_split_orig.csv", chunksize = 1000000,
    iterator = True)
all_classes = np.array([0, 1])
for chunk in train:
```

```
    y_train = chunk["click"]
    chunk = chunk[cols]
    chunk = chunk.join(pd.DataFrame([dayhour(x) for x in chunk.hour],
        columns=["wd", "hr"]))
    chunk.drop("hour", axis=1, inplace = True)
    Xcat = fh.transform(np.asarray(chunk.astype(str)))
    clf.partial_fit(Xcat, y_train, classes=all_classes)

# Test classifier
X_test = pd.read_csv("../Data/test_split_orig.csv", usecols = cols + ["id"],
    nrows = 1000000)
X_test = X_test.join(pd.DataFrame([dayhour(x) for x in X_test.hour],
    columns=["wd", "hr"]))
X_test.drop("hour", axis=1, inplace = True)

X_enc_test = fh.transform(np.asarray(X_test.astype(str)))

y_pred = clf.predict_proba(X_enc_test)[:, 1]
```

An SGD Classifier is trained with the loss function as log loss. The data is loaded in chunks of 1 million rows iteratively as a means to handle the big data and fit the model using partial_fit method. Before passing the data to the classifier, the values are transformed using feature hasher's tranform method. The same procedure is followed for the test set and then predict the probabilities. The results thus obtained are compared with the true labels and evaluation metrics log loss is generated and ROC curve is plotted.

4.3.2 Vowpal Wabbit

The commands used for logistic regression without regularization were :

```
#For training
vw -d train_split_vw.vw -f avazu_log.model.vw --loss_function logistic

#For testing
vw -d test_split_vw.vw -t -i avazu_log.model.vw -p avazu_log.preds.txt
```

The preprocessed dataset is trained using logistic regression algorithm and logistic as the loss function and the generated model is saved.

This trained model is loaded from the file and is used to predict the classification probabilities of each entry in the test set. The prediction are saved and the results thus obtained are compared with the true labels and evaluation metrics log loss is generated and ROC curve is plotted.

Here, -d is for specifying the dataset, -f is for saving the model, -t specifies its testing, -i is for using the saved model and -p is to save the predictions.

The commands used for logistic regression with l2 regularization were :

```
#For training
vw -d train_split_vw.vw -b 22 -l 0.158789 --l2 4.7e-14 -f avazu_log.model2.vw
   --loss_function logistic


#For testing
vw -d test_split_vw.vw -t -i avazu_log.model2.vw -p avazu_log.preds2.txt
```

In the above commands, -b specifies the feature hasher's space which is $2 * 22$ and -l specifies the learning rate for online gradient descent. The commands used for neural networks with a single hidden layer containing 3 neurons were :

```
#For training
vw -d train_split_vw.vw -binary -f avazu_nn.model.vw --loss_function logistic


#For testing
vw -d test_split_vw.vw -t -i avazu_nn.model.vw -p avazu_nn.preds.txt
```

5 Results and Analysis

5.1 Logistic Regression using scikit-learn

5.1.1 Results

Property	Value
Training Time	0:41:42.712519
Log Loss	0.406976
Area under ROC curve	0.68

Table 4: Results of Logistic Regression with scikit-learn

5.1.2 ROC Curve

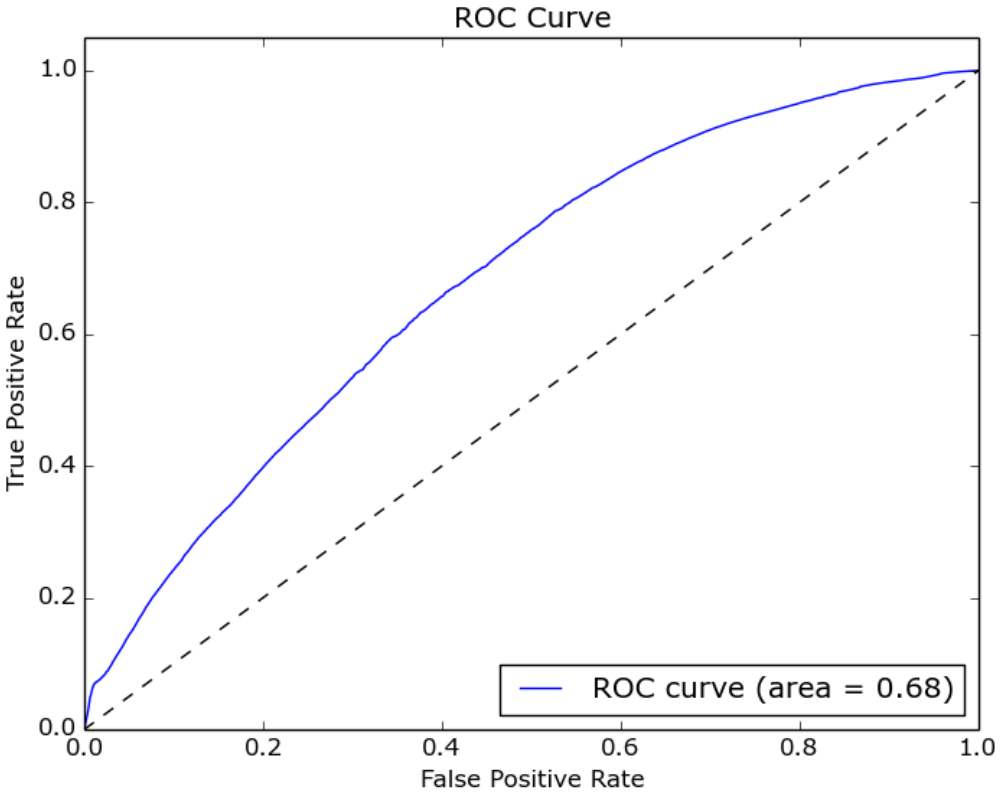

Figure 5.1: ROC Curve for Logistic Regression on scikit-learn

5.2 Logistic Regression using Vowpal Wabbit

5.2.1 ROC Curve

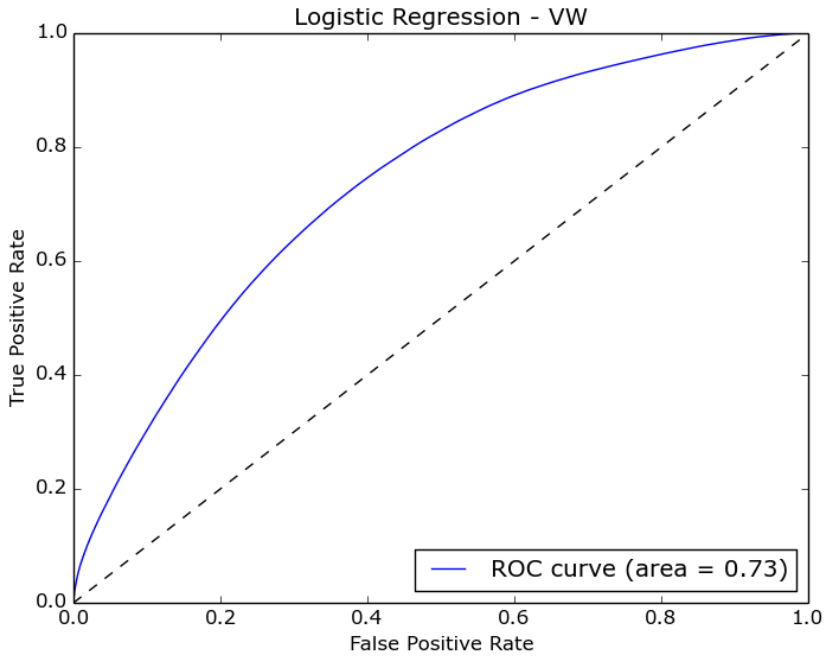

Figure 5.2: ROC Curve for Logistic Regression without L2 Regularization

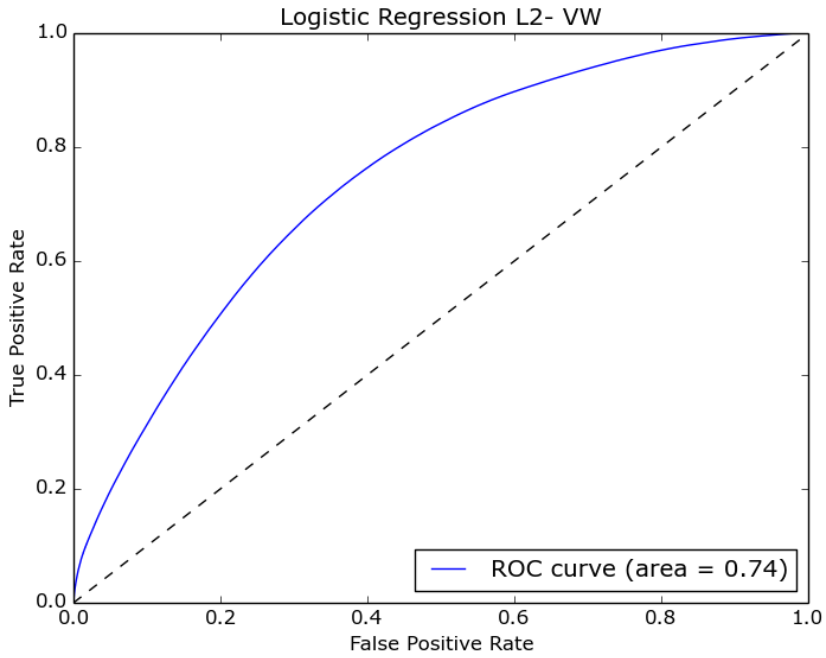

Figure 5.3: ROC Curve for Logistic Regression with L2 Regularization

5.2.2 PR Curve

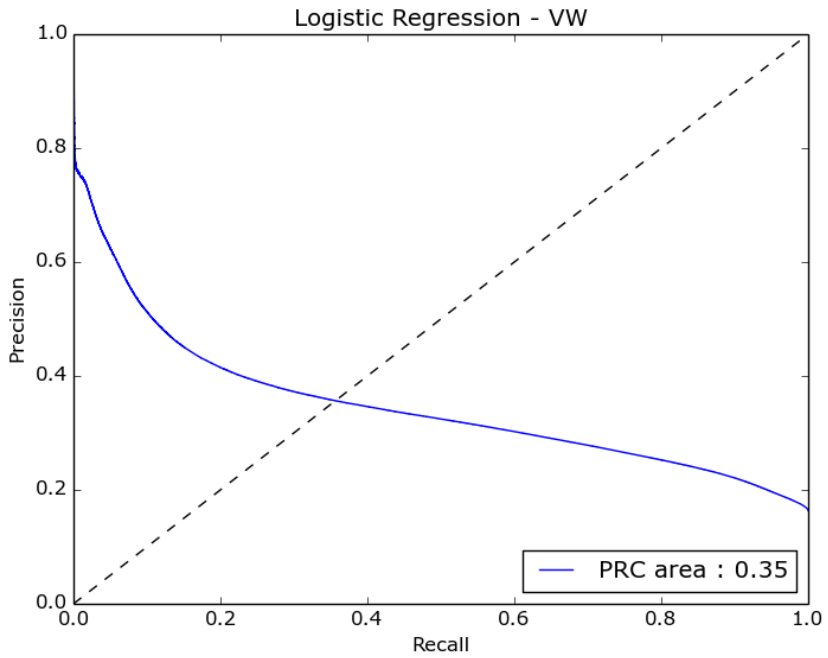

Figure 5.4: PR Curve for Logistic Regression without L2 Regularization

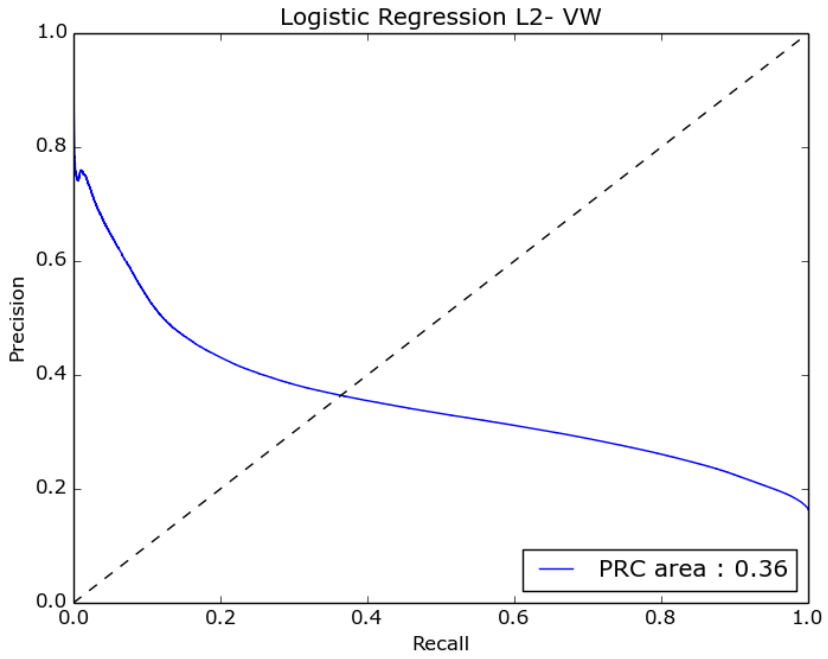

Figure 5.5: PR Curve for Logistic Regression with L2 Regularization

5.2.3 Results

	without L2 Regularization	with L2 Regularization
Training Time	0:04:26	0:02:42
Log Loss	0.456804	0.391805
Area under ROC curve	0.73	0.74
Area under PR curve	0.35	0.36

Table 5: Comparision of Logistic Regression with and without L2 Regularization

As it can be seen from Table 6, Logistic Regression with L2 Regularization performs better than one without L2 regularization. This is visible clearly from the fact that area under the ROC curve for the latter is smaller than the former. Hence, only Logistic Regression with L2 Regularization is considered for our analysis.

5.3 Neural Networks on Vowpal Wabbit

5.3.1 ROC Curve

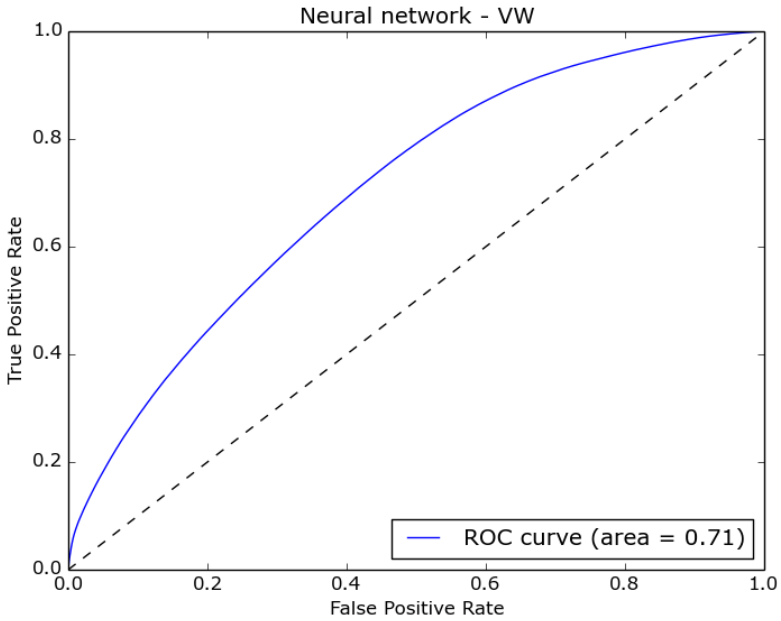

Figure 5.6: ROC Curve for Neural Networks

5.3.2 PR Curve

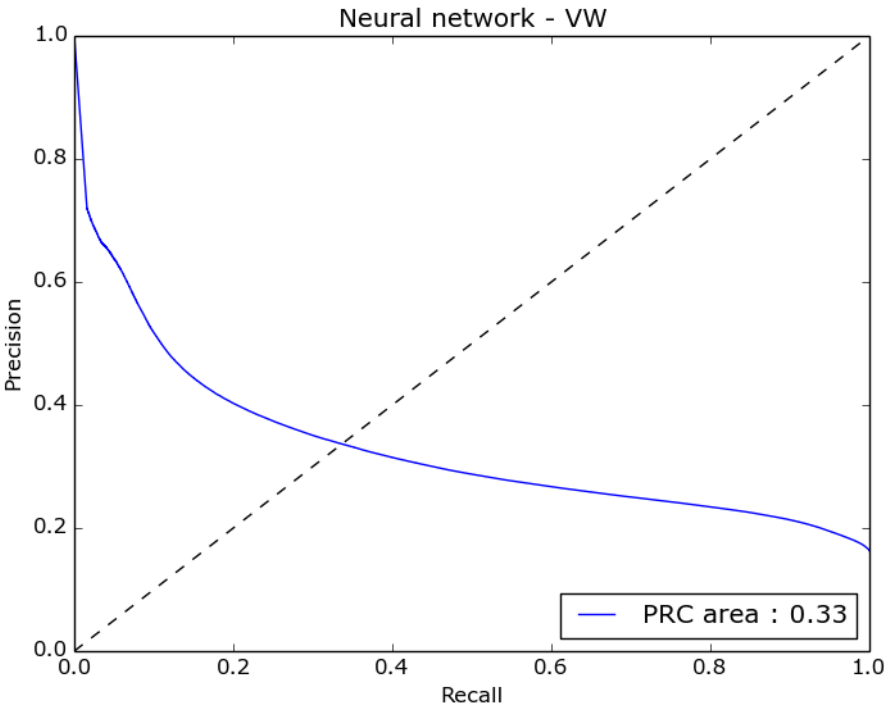

Figure 5.7: PR Curve for Neural Networks

5.3.3 Results

Property	Value
Training Time	0:10:14.4232
Log Loss	0.446836
Area under ROC curve	0.71

Table 6: Results for Neural Networks

5.4 Comparison of Results

Property	**Logistic Regression (sklearn)**	**Logistic Regression (VW)**	**Neural Networks (VW)**
Training Time	0:41:42.712519	**0:02:42.0237**	0:10:14.4232
Log Loss	0.406976	**0.391805**	0.446836
ROC AUC	0.68	**0.74**	0.71

Table 7: Comparison of Results

6 Conclusion & Future Work

As observed from the results based on the various evaluation metrics, the following things are clear. The three ROC curves are significantly distant from the $45deg$ line implying that our prediction models are quite accurate and definitely perform better than a random classifier. The training time for Vowpal Wabbit which is an online learning tool is drastically lesser than that of scikit-learn as expected from an online learning method. Hence it is capable of handling large datasets more efficiently in lesser training time. Hence Vowpal Wabbit is a better framework than scikit learn for big datasets. Vowpal Wabbit seems to generate a higher logloss for all the algorithms except when L2 regularization was used on Logistic Regression. Neural Networks also gave a higher AUC than Logistic Regression in scikit-learn and lesser AUC than L2 regularized logistic regression on Vowpal Wabbit. Area under the curve is seen to be higher for the ROC curve generated when logistic regression with L2 regularization is run on Vowpal Wabbit proving that it performs better as a prediction model compared to the other methods. Moreover the logloss for L2 regularization with logistic regression on Vowpal Wabbit was the least (0.39) of all the experiments hence proving that it is the best machine learning model amongst the experiments run above.

From all the above experiments, it can concluded that Click through rate prediction with a ROC AUC of 0.74 and logloss of 0.39 is obtained using the prediction model which is trained over the big dataset using L2 Regularized logistic regression in 2 minutes 42 seconds when implemented on Vowpal Wabbit.

6.1 Novelty of the work

In the project, various machine learning techniques were implemented and results were analyzed. Different algorithms such as Logistic Regression and Neural Networks were executed on the Click Through dataset and thus their effectiveness was compared. Analysis was done based on metrics such as training time, logloss and ROC curves which are highly relevant in today's data analysis field. Comparing various metrics gave insight into which of them were important and which was most crucial in decision making. Feature Hashing was also included as part of the methodology to tackle the problem. This is a very important technique as it reduces RAM cost and hence also improves running time

of the algorithms.These were clearly visible in the results that were obtained on performing the experiments. A relatively new approach to data analysis, Online Learning, was also experimented with the use of the tool Vowpal Wabbit. Online Learning is one of the newest technologies which shows promise in the field of Big Data and the results obtained showed huge difference in time taken between traditional approach and Vowpal Wabbit approach proving that online learning was here to stay.

6.2 Future work

Though good results were obtained in this research project, subsequent work can be done this area using more advanced algorithms and implementation techniques to get better accuracy with more efficiency. Following are a list of possible improvements in the near future to build prediction model for click through rate.

1. Follow The Regularized leader (FTRL) method for regularizing the logistic regression [9].

2. Implementing Factorization Machines as a machine learning algorithm.

3. Enhanced Feature Extraction methods to derive complex features.

4. Apache Spark implementation to parellelize the machine learning model to leverage the cloud architecture thereby improving the performance.

References

[1] Feature hashing. Wikipedia, The Free Encyclopedia.Web, 2014.

[2] Logistic regression. Web, 2014.

[3] Loss function. Wikipedia, The Free Encyclopedia.Web, 2014.

[4] Neural networks. Wikipedia, The Free Encyclopedia.Web, 2014.

[5] Precision recall curve. Wikipedia, The Free Encyclopedia.Web, 2014.

[6] Receiver operating characteristics. Wikipedia, The Free Encyclopedia.Web, 2014.

[7] Support vector machines. Wikipedia, The Free Encyclopedia.Web, 2014.

[8] John Langford, L Li, and A Strehl. Vowpal wabbit. *URL https://github. com/John-Langford/vowpal_wabbit/wiki*, 2011.

[9] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230. ACM, 2013.

[10] Steffen Rendle. Social network and click-through prediction with factorization machines. 2012.

[11] Xingxing Wang, Shijie Lin, Dongying Kong, Liheng Xu, Qiang Yan, Siwei Lai, Liang Wu, Alvin Chin, Guibo Zhu, Heng Gao, et al. Click-through prediction for sponsored search advertising with hybrid models.

[12] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120. ACM, 2009.