**Machine Learning for Resource classification on the basis of user role**

*Project Report submitted*

*In partial fulfilment of the requirements for the degree of*

**Bachelor of Technology**

*By*

**Yash Phogat**

**131100063**

**Harsimran Bhasin**

**131100021**

Under the Supervision of

**Ms. Akanksha Juneja**



Department of Computer Science and Engineering

National Institute of Technology Delhi

May 2017

**Acknowledgment**

Firstly, we would like to express our sincere gratitude to our supervisor Ms. Akanksha Juneja, for her expert guidance, support and encouragement. Without her help, we would never have been able to make any progress in our Project work entitled “Machine Learning for Resource classification on the basis of user role”. The brainstorming sessions are going to stay inside our hearts forever. These discussions prompted us to think beyond the obvious and pushed us to deliver our best.

Her involvement in the project was remarkable. It’s only because of her efforts that we have come so far, in this field.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Yash Phogat Harsimran Bhasin

131100063 131100021

**Declaration**

We, **Yash Phogat** and **Harsimran Bhasin**, hereby declare that the work contained in the project titled “Machine Learning for Resource classification on the basis of user role” submitted to the Department of Computer Science and Engineering National Institute of Technology, New Delhi, in partial fulfillment of the requirements for Bachelor of Technology has been carried out by us under the supervision of our supervisor. The results embodied in this work have not been submitted in part or in full, to any other university or institute.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Yash Phogat Harsimran Bhasin

131100063 131100021

**Certificate**

This is to certify that the project titled **Machine Learning for Resource classification on the basis of user role** submitted by **Yash Phogat** and **Harsimran Bhasin** to National Institute of Technology, Delhi, is a record of bonafide work done under my supervision and I consider it worthy of acceptance for the award of the degree of Bachelor of Technology of the Institute.

Signature of Supervisor Signature of HOD

**Ms. Akanksha Juneja**

Date:

**Abstract**

In this project the idea is to learn and predict the resource (or files) allocation nature for different resources in a firm on the basis of an employee’s role in that firm. The role of an employee can be decided on the basis of few attributes associated with every employee (such as department, position etc.). This report elaborates the process of digging important information from the data which apparently is highly unstable and applying Machine learning algorithms to get a fair prediction model. Initially we started with linear prediction algorithms such as Support Vector Machines (SVM) and Decision Trees. Going forward we used ensemble prediction algorithms like Random Forest and Gradient Boosting. Since the data has got large categorical features hence we applied One Hot Encoding for the feature categories and again applied Random Forest. Finally an integrated model of the encoded and the usual Random Forest helped to obtain 88.5% accuracy.

***Keywords:*** *Unbalanced data, categorical features, Support Vector Machine, Decision Tree, Random Forest, Gradient Boosting, One Hot Encoding.*

**List of Figures/Tables**

**Chapter No. Figure No Figure Caption Page No**

5 5.1 Sample ROC curve 11

6 6.1 ROC curve comparison for all the models 16

Appendix 1 A1 SVM AUC Curve 19

Appendix 1 A2 Weighted SVM 20

Appendix 1 A3 Decision tree with SMOTE 21

Appendix 1 A4 Gradient Boosted Machine 22

Appendix 1 A5 Random Forest 23

Appendix 1 A6 Random Forest with One Hot Encoding 24

**Chapter No. Table No. Table Caption Page No**

2 2.1 Description of each feature in data 3

2 2.2 Example of One Hot Encoding 5

6 6.1 Contribution of every feature in classification 14

**Contents**

Title page i

Acknowledgment ii

Declaration iii

Certificate iv

Abstract v

List of figures/tables vi

Contents vii

**Chapter 1** Introduction 1

1.1 Problem Statement 1

1.2 Contribution 2

1.3 Organisation of report 2

**Chapter 2** Literature Review

2.1 Data 3

2.2 Classification Models 4

2.3 Optimization techniques 5

**Chapter 3** Prediction Models – Linear Classifiers

3.1 Support Vector Machines 7

3.2 Decision Tree 7

**Chapter 4** Prediction Models – Ensemble Classifiers

4.1 Gradient Boosting 9

4.2 Random Forest 10

**Chapter 5** Evaluation Metric: Area under the ROC curve 11

**Chapter 6** Simulation and Results

6.1 SVM 13

6.2 Weighted SVM 13

6.3 Decision Tree 14

6.4 Gradient Boosting Machine 15

6.5 Random Forest 15

6.6 One Hot Encoded random forest 16

6.7 Integrating of our Best models 16

**Chapter 7** Conclusion and Future Work 17

Appendices 19

References 25

**Chapter 1: Introduction**

The scope of software for performing tasks that can be performed automatically has increased exponentially in the past few years. Software developers today, aim to make machines that may act like humans and reduce the human effort in laborious jobs that demand repetition.

One such task is the allocation of resources in a large company. Every employee requires access to huge number of resources during his/her career at the company. These access privileges tend to change regularly, with the movement of the employee to different verticals of the company. The daily human effort needed to grant, revoke and change these access privileges is massive. Hence there was a need to design software models that may be able to perform these tasks automatically.

The predictions are based on the following two hypothesis:

1. Approvals/Rejections were based on relationship between roles and resources

2. The relationship varied for each resource or “class” of resource

**1.0.1. Motivation**

Determining resource access privileges of employees is a popular real world problem for many big companies such as Microsoft, Google, Amazon, TCS etc. In this direction Amazon Inc. held a global contest (2013) to design machine learning models that may predict the access privileges for its employees on the basis of their historic data of employee access (2010-2011). It is established that employees need to know what kinds of resources they are or they are not supposed to get access to, in order to perform their tasks. To save the cost and time devoted to this activity, our aim must be to build a prediction model that automatically determines resource access privileges of employees.

**1.1 Problem Statement**

The objective of this project is to understand the nature of the data available for every employee role and to design a prediction model that may automatically approve or reject employee’s resource allocation request. We need to design the prediction model from past data available for various resources.

**1.2 Contribution**

We have been able to dig some important information from the training data which implies high order of instability in the data. The training data consists of a proportionally negligible number of rejection instances. The problem constitutes of categorical attributes with large number of categories for each attribute. Due to the large dimension of the instances of the data, the choice of using a SVM model was made initially. A Cross-validation framework was formulated for approximating the accuracy of the SVM model learned for different cost and gamma values. The learned model gave ~93% accuracy, in the train set. Although the proportional inconsistency of the rejection instances in the train set led to a ~51% accuracy on the test set. Working on the inconsistency problem led to the use of class weights to increase the weightage of minority class in sample selection. This increased the test accuracy to ~61%. Further decision trees were used, but the effect on accuracy was improved only by 0.37%. Moving forward to Ensemble models we obtained a significant increase in accuracies with GBM giving 81.6% accuracy, Random Forest giving 85.7% accuracy. An integration of the two models gave an accuracy of 85.1%. Going forward there was an attempt to encode the highly categorical data for extremely boosted random forest algorithm which increased the accuracy of the model to 87.5%. Further integration of normal Random Forest and Encoded Random Forest algorithms yielded a model with 88.5% accuracy.

**1.3 Organization of the Report**

The report starts with the study of the literature we referred to, from data sources to algorithm studies. The literature section gives a short introduction to each algorithm used providing an overview of the algorithm and additive parameters. The following three chapters give details about the linear classification algorithms, ensemble classification algorithms and the evaluation metric- Receiver Operating characteristic (ROC) curve. The algorithms are elaborated and details of important parameters is given. The significance of the ROC curve used for evaluation has been talked about in the fifth chapter. The sixth chapter talks about the flow of events in the project along with the algorithm specific details of every model learned. This chapter talks about parameter values and the results of each algorithm used. The seventh chapter gives a short summary of the project and the results obtained. It also touches upon the future prospects of the problem. The Appendices enclose the ROC curves of each algorithm to present a visual schema for easy comparison. Finally the report closes with references to the worthy sources that gave us a direction to proceed in this project.

**Chapter 2: Literature Review**

**2.1 Data**

The data for the project is obtained from the Kaggle[1] website; which is the historical data of Amazon Inc. from 2010 to 2011. The data consists of a training set of 32769 samples stating the employee role attributes, the resource and whether the resource was approved or rejected. The test set consists of 58922 samples without the ACTION attribute (i.e. approved: 1/rejected: 0). The following table shows the description of each feature in the data.

TABLE 2.1 Description of each feature in data

|  |  |  |
| --- | --- | --- |
| **Feature Name** | **Feature Meaning** | **Feature type (No. of unique categories)** |
| ACTION | “1” : Approved  “0” : Rejected | Categorical (2) |
| RESOURCE | Resource ID | Categorical (7518) |
| MGR\_ID | ID of the employee’s manager | Categorical (4243) |
| ROLE\_ROLLUP\_1 | Company role category ID1 (e.g. US Engineering) | Categorical (128) |
| ROLL\_ROLLUP\_2 | Company role category ID2 (US Retail) | Categorical (128) |
| ROLE\_DEPTNAME | Department description | Categorical (449) |
| ROLE\_TITLE | Business title description | Categorical (343) |
| ROLE\_FAMILY | Role family extended description | Categorical (67) |
| ROLE\_CODE | Unique ID for each company role (e.g. Manager) | Categorical (343) |

The data analysis also implied that the train data has 1897 samples of class 0 and 30872 samples of class 1, thereby leading to a result that on every 100 samples of class 1 there are approximately 6 samples of class 0, hence making class 0 a minority class.

**2.2 Classification Models**

**2.2.1 Support Vector Machines**

While researching for classification models, we came across an analogy of the problem to resource allocation in clouds. Clouds use SVM regression to estimate the response time in the next measurement period, and the resources are redistributed based on the current status of all virtual machine installed in physical machines[2]. This fact drove us to start by using SVM classification for our prediction model. An advantage of using SVM was that the SVM model can project a high-dimensional hyper-plane to an SVM kernel dimension.

**2.2.2 Decision Trees**

Decision tree is a machine learning algorithm that builds class prediction models in the form of a tree data structure. It breaks down a dataset into smaller subsets based on the potential information gain in every feature. An associated decision tree is incrementally developed following a similar process. The ﬁnal result is a tree with leaf nodes that represents a classiﬁcation or decision.

**2.2.3 Gradient Boosted Machine**

Gradient Boosting[7] follows the boosting principle of machine learning, where various weak learners are iteratively applied over the sampled data to obtain a strong learner.

Gradient boosting involves three elements:

1. A loss function that determines the accuracy loss or prediction error of the model.
2. A weak learner is a prediction model that has failed to make an acceptable prediction of the class labels.
3. An additive model is a method to integrate different weak learners to achieve minimal loss.

**2.2.4 Random Forest**

Random forest[6] follows the methods of bagging with random samples of data. The main idea is to build various decision trees depending on the random sampling of training data and random selection of features when building each tree. In prediction, we take all the decisions made by each tree into account and select the majority result as our prediction.

**2.3 Optimization techniques**

**2.3.1 Synthetic Minority Over-sampling Technique (SMOTE)**

As the name suggests SMOTE [5] can be used with decision trees to solve the data imbalance problem. The SMOTE function oversamples the rare class instances by using bootstrapping and k-nearest neighbor to synthetically create additional instances of that class label.

**2.3.2 One Hot Encoding**

One Hot Encoding[8] is an encoding technique used to convert categorical data to one hot format to individually identify a category as a unique feature. One-hot refers to a group of bits among which has only a single set (1) bit and all the other bits shall be reset (0).

Table 2.2 shows the example with one hot encoding on a feature with three categories, which will be encoded into “001”, “010” and “100” separately. Since most of the sample values are “0”s, we use sparse matrix to represent the newly encoded feature space.

TABLE 2.2 Example of One Hot Encoding

|  |  |  |
| --- | --- | --- |
| **S.No** | **Category Data** | **One Hot Encoded Data** |
| 1 | 118322 | 001 |
| 2 | 118321 | 010 |
| 3 | 112565 | 100 |

**Chapter 3: Prediction Models – Linear Classifiers**

A linear classifier makes a classification decision based on the linear combination of the data features. In the following sections we introduce some of the linear classification algorithms we applied.

**3.1 Support Vector Machines**

Support Vector Machine (SVM) [9] is a supervised machine learning algorithm which we have used for resource classification. In this algorithm, we plot every instance as a point in k-dimensional space (where k is number of features in the data). Every feature is assigned a particular axis in the plot and the feature value gives the feature coordinate for that instance. Then, we attempt to learn the hyper-plane that differentiate the two class labels.

In R we have an SVM function in the e1071 library. Some important parameters of the SVM function have been discussed below:

***Cost*** is a soft margin function, which controls the influence of each individual support vector. This process involves trading error penalty for stability.

***Gamma*** is used to denote the effect of biasness and variance on an SVM model. Large gamma leads to high bias and low variance models and vice-versa

***Kernel*** is a similarity function. It is a function that we, as the domain expert, provide to a SVM algorithm. It takes couple of inputs and give out there similarity measure.

***Class.weights*** is a parameter to define the weights of the class labels as a vector of class labels.

**3.2 Decision Trees**

A decision tree takes as input an instance described by a set of features, and outputs a 1/0 “decision”. Each tree node tests the value of a feature. Branches from the node correspond to possible values of the corresponding feature. Leaf nodes supply the class labels to be returned if that leaf is reached. The features allocated to each node is decided on attributes such as information gain for every feature. We start with features having maximum information gain possible in the root node and thereafter iteratively decide the feature orientation on other levels based on their respective information gains.

As one of the major concerns of our problem is the data imbalance, therefore we needed weightage of the classes in the Decision Trees as well. For this we use oversampling of the minority class by SMOTE [5] function of DMwR package.

The basic algorithm of SMOTE [5] is that for each minority class instance (i.e. 0), we generate a synthetic instance from some of the k nearest neighbors of the original instance. This process can be interpreted as choosing a random point in the line between two feature vectors as our new samples. Two important attributes of SMOTE that must be defined to set the extent of oversampling or undersampling (for majority class) are perc.over and perc.under.

***perc.over*** – determine the percentage increase in the instances of minority class.

***perc.under*** – determine the percentage decrease in the instances of majority class

Note: SMOTE performs both undersampling and oversampling.

**Chapter 4: Prediction Models – Ensemble Classifiers**

Ensemble method is a type of supervised learning, which employs a set of classiﬁers and their decisions are combined in certain way. Generally, the ensemble model involves bagging and boosting.

The main idea of bagging is that we generate several new training data set by random uniform sampling the original data set with replacement. Then use each new sample to train a different model. Boosting is an ensemble method combining a set of weak learner properly, to obtain a strong learner.

**4.1 Gradient Boosting Machine(GBM)**

As the name suggests Gradient Boosting [7] is a Boosting technique that typically uses Decision Trees as the weak learner over random samples, to integrate the learned models and obtain a stronger learner. As we already know that GBM involves 3 components:

1. Loss Function: A function to measure the relative loss of accuracy for every weak learner model we learn on various samples. E.g.: Squared error, logarithmic loss.
2. Weak Learner: The learner algorithm that gives a dim accuracy on its respective sample. We iteratively attempt to identify better learners, to obtain an integrated strong learner.
3. Additive Model: Weak learners are added one at a time, and the previous existing learners in the model are not modified. A gradient descent procedure is applied to minimize the loss when adding the weak learners; where weights are assigned to every primitive weak learner in every iteration, to finally obtain a strong learner. This is analogous to the weights assigned to nodes in neural networks.

Some parameters of the “gbm” model are discussed below:

1. *Number of trees*: we use this parameter to limit the number of trees to be used for boosting.
2. *Tree depth*: this parameter limits the depth of each tree in the boosting technique. It is proportional to the algorithms complexity.
3. *Number of nodes or number of leaves*: this parameter is used to limit the tree size.
4. *Number of observations per split:* this parameter defines a minimum limit on the train sample required to make a split.
5. *Minimum improvement to loss:* This parameter acts as a constraint on the minimum improvement that shall be made by any split before it is added to the model tree. Anything deteriorating the accuracy is unwanted.

**4.2 Random Forest**

Random Forest [6] is often attributed as the most powerful ensemble learning algorithm that works on the principle of bagging. It can be considered as an integration of the decision trees using bagging technique. Here random samples of the data are chosen and there is random selection of features as well. Then a decision tree is constructed on this random sample set. Similarly large number of such trees are created in the random forest algorithm. Now for every instance the predicted class label of the majority of the trees is considered as the relevant class label.

**4.2.1 One Hot Encoding with Random Forest**

A further optimization to the Random Forest model is by creating one-hot encoded data of the feature categories using One Hot Encoding with Random Forest [8]. Categorical variables are intentionally encoded as numerical variables in order to be used as features in any given model.

e.g. [London, US, India] becomes [10211, 10212, 10213].

This imparts an ordinal property to the variable, i.e. London< US< India that is usually not desired, one hot encoding is necessary for the proper representation of the distinct elements of the variable.

**Chapter 5: Evaluation Metric: Area under the ROC curve**

A receiver operating characteristic curve (ROC curve)[4] is a graphical representation of the performance of a potential learner. The curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The TPR is also known as sensitivity, probability of detection or recall. The FPR is also known as fall-out or probability of false alarm. FPR is equal to (1 − specificity).

Curves for each learner are plotted in the ROC plot. The significance of the ROC curve for learner evaluation is that, the learner’s accuracy is equal to the area under this ROC curve.

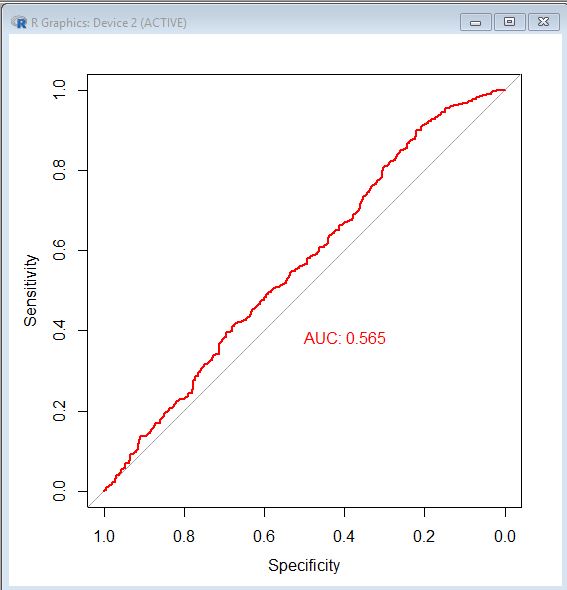


Figure 5.1 Sample ROC curve

In the above ROC plot, the red curve determines the curve of a weak learner covering an area of 0.565. This directly implies the accuracy of the learner as 56.5%. The mid black line, is the roc plot of random assumptions for test class labels that gives 50% accuracy. The area under the curve is calculated by integrating large number of rectangular strips (highlighted).

True Positive Rate = Sensitivity = TP/ (TP+FN) (5.1)

False Positive Rate = 1- Specificity= 1 - TN/ (TN+FP) = FP/ (TN+FP) (5.2)

**Chapter 6: Simulation & Results**

In the following section, we will discuss the simulation results with the discussed prediction models. Firstly, we try linear predication models which have been applied directly on categorical data like SVM and decision tree. We again implement the linear models tackling the class imbalance problem. Next, we establish the fact that linear classiﬁers does not predict well on the given data. Then we further try ensemble models including random forest and gradient boosting based on decision tree. In addition, we evaluate the random forest model along with one- hot encoding to improve the AUC [4] by efficient implementation of probabilistic classification of categorical features. Finally, we combine our two best prediction results from random forest and one-hot encoding based random forest to obtain further performance improvement.

**6.1 SVM**

We started with SVM due to our motivation from the analogy of the current problem and the cloud resource allocation problem. We used the e1071 library in R. To best tune our model over the training data in order to obtain cost and gamma values for classification we used best.tune () function in R. The function returned the best cost and gamma value over the linear and the radial basis function of 1 and 0.11111 respectively. Then the SVM model was run over the cross validated train data, sampled by the 80:20 rule. We obtained a learner accuracy of 56% in train data locally but 51% accuracy on test data. Figure A1 (Appendix 1) displays the AUC curve for SVM. This happened due to data imbalance. The learner model overfitted the train data by classifying all the instances to the majority class.

**6.2 Weighted SVM**

To address the class imbalance problem we moved to weighting. We increased the weightage of minority class by 94 times and reduced the weights of majority instances by 6. The weighting values were decided on the basis of the presence of complimentary class samples in the train data. In other words per every 100 train instances the majority class holds 94 instances and minority class holds 6 instances. The weighted SVM increased the accuracy to 61%. The local learner AUC for weighted SVM is given in figure A2 (Appendix 1).

**6.2.1 Feature Importance**

An attempt was made now, to select important features from the array of features to obtain a better result. For this, weighted SVM model was run individually on every feature but none of the features gave a significant contribution towards classification thereby resulting in dropping the idea of feature selection at this stage. The individual contributions of every feature are given in table 6.1

TABLE 6.1 Contribution of every feature in classification

|  |  |
| --- | --- |
| **Feature** | **Contribution In Classification (%)** |
| RESOURCE | 5.808 |
| MGR\_ID | 8.09 |
| ROLE\_ROLLUP\_1 | 8.5 |
| ROLE\_ROLLUP\_2 | 5.82 |
| DEPTNAME | 7.14 |
| ROLE\_TITLE | 5.45 |
| ROLE\_FAMILY\_DESC | 5.45 |
| ROLE\_FAMILY | 5.4 |
| ROLE\_CODE | 6.1 |

**6.3 Decision Tree**

After SVM we made an attempt with decision trees as they tend to handle categorical data in a much comfortable manner. The rpart () function of the rpart package. Keeping in mind the class imbalance in the data we considered SMOTE [5] for oversampling of the minority features. SMOTE () is function in the DMwR package used to introduce synthetic instances of the minority class to the training data. The perc.over parameter was valued 100 and the perc.under parameter was valued at 200. This implied doubling of the minority class samples and halving of the majority class samples respectively.

On training the sample and then locally testing the model a training AUC of 0.607 was obtained which when applied over the test data gave a test AUC of 0.62. The training AUC can be seen in Figure A3 (Appendix 1).

**6.4 Gradient Boosting Machine**

Since the linear models couldn’t give a great accuracy we decided to move on to ensemble models that apply either bagging or boosting techniques on weak learners to obtain a strong learner. The first ensemble model used was Gradient Boosting machine [7] that follows the principle of boosting. In this algorithm various weak learners are iteratively applied over the sample to reduce loss for each weak learner when integrated. It follows the boosting principle of model enhancement.

In R there is a gbm () function in the gbm package to implement GBM [7] algorithm. In a GBM algorithm, typically the weak learner is the decision tree. Typically there does not exit a general method to determine the parameters such as maximum depth and minimum sample splits in the decision tree. Therefore this was a hit and try approach. The values on which we got the best training AUC were 8 for the maximum depth, adaboost distribution and 400 no. of decision tree iterations. We obtained the train AUC of 0.802 and the implemented test AUC of 0.817. The training AUC can be seen in Figure A4 (Appendix 1).

**6.5 Random Forest**

Though we obtained a significant raise in model accuracy by using GBM for ensembling, we moved to random forest for moving forward. Random forest [6] is claimed to be among the most powerful learner algorithms working on the concept of bagging. In this algorithm, random sampling of the instances and the features is done to create n number of decision or regression trees and the result for the classification of an instance is made by the result obtained via majority of trees. In R we used the randomForest () function of the randomForest library.

Here again we had to set a limit for the trees, which was set to 200 for optimal results (by hit and trial method). The train and test AUC’s were 0.872 and 0.538 respectively. The train AUC can be seen in figure A5 (Appendix 1).

**6.6 One Hot Encoded Random Forest**

Due to the presence of highly categorical data, we decided to encode our data. For this purpose we used One-Hot Encoding [8] where every category can be identified uniquely on the basis of bits. The advantages of this has been discussed earlier. Further since One Hot Encoding introduces large number of zeroes in the data, hence we use sparse matrix that ignores the zeroes, thereby saving us from computational limitations.

For sparse matrix creation we used the sparse.model.matrix () function of the Matrix library in R and for further classification we used xgboost library in R. Tweaking the parameters of the xgboost () function gives us the random forest algorithm implementation. This encoded random forest gave us a further increase of 0.001 in train AUC but 0.03 in test AUC.The train AUC can be seen in Figure A6 (Appendix 1).

**6.7 Integration of our best models**

Finally we attempted to integrate the results of our final two best learners. For this we simply multiplied their predictions with their respective AUC values and then divided by the sum of those AUC’s. This gave us the final accuracy for our model as 88.5 % pink line in the figure.

The following figure shows the ROC curve comparison of all our models Figure 6.1

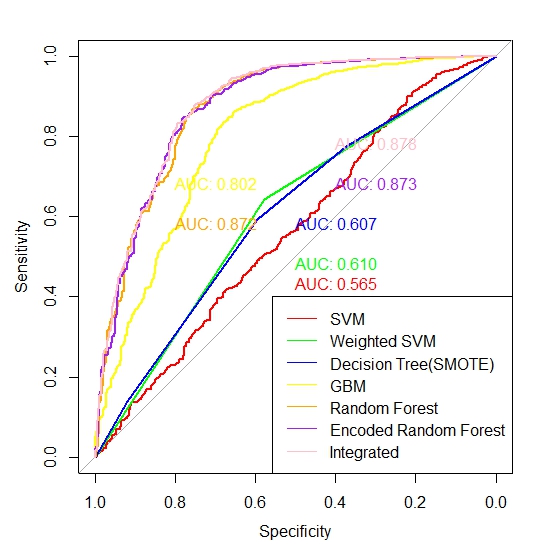


Figure 6.1 (ROC curve comparison for all the models):

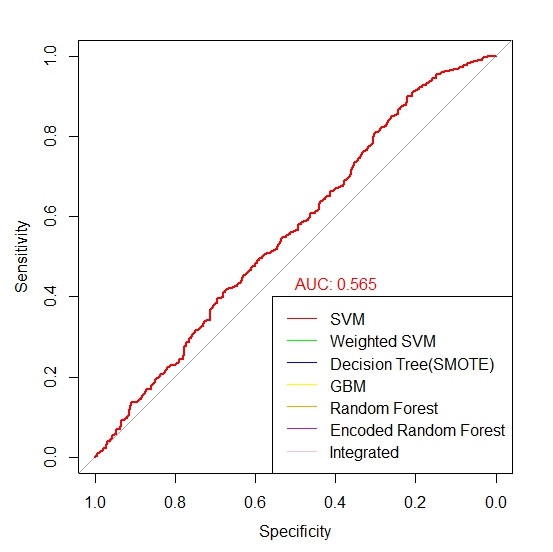
**Chapter 7: Conclusions & Future Work**

The following are the conclusions drawn from the project:

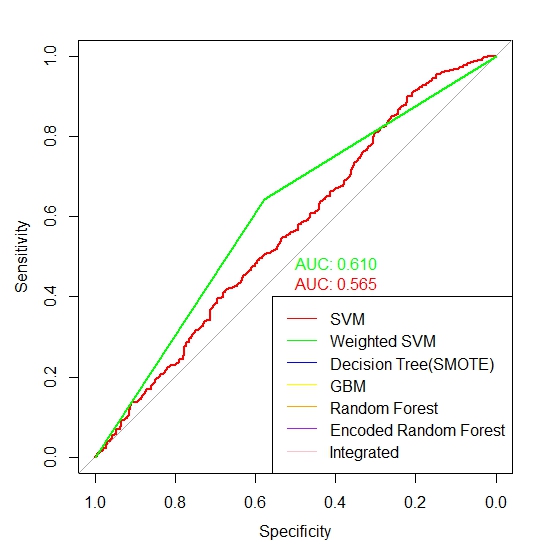
1. The train data was imbalanced with 1 as the majority class and 0 as the minority class.
2. We applied SVM linear model over the data after weighting to obtain an accuracy of 61%.
3. Further we applied the Decision tree algorithm with SMOTE [5] to get the accuracy upto 62%.
4. Moving on to ensemble models we get an 81% accuracy in GBM[7] and 85% accuracy in Random Forest [6].
5. 87.5% accuracy is obtained when we encode the data for random forest classification [8].
6. Finally we end, with 88.5% accuracy on integrating our two best models i.e. random forest and encoded random forest.
7. In future to increase the accuracy, feature selection and extraction may be made.
8. Further integration of different one-hot encoded models [8] (linear as well as ensemble) may be made.

**Appendix 1**

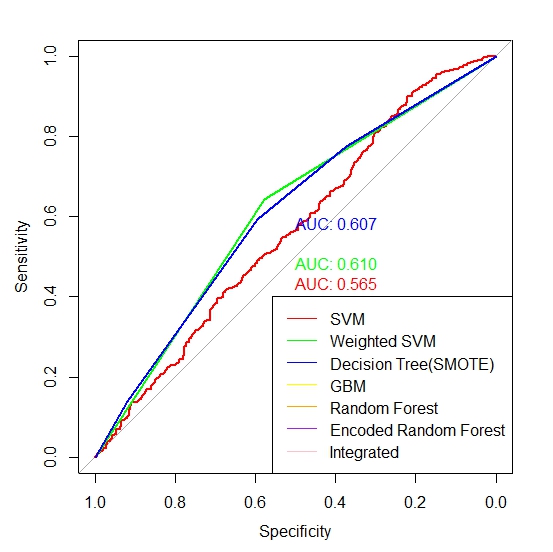
**FIGURES**



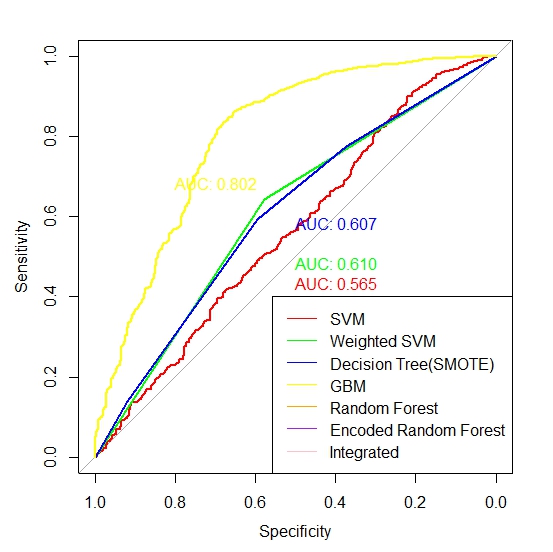
**Figure A1: SVM AUC Curve**



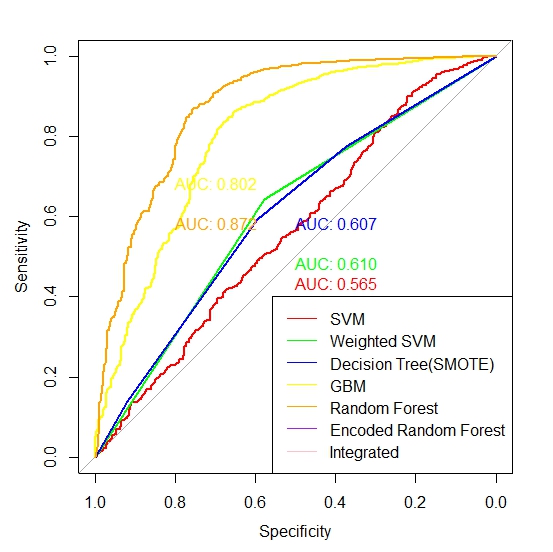
**Figure A2: Weighted SVM (GREEN)**



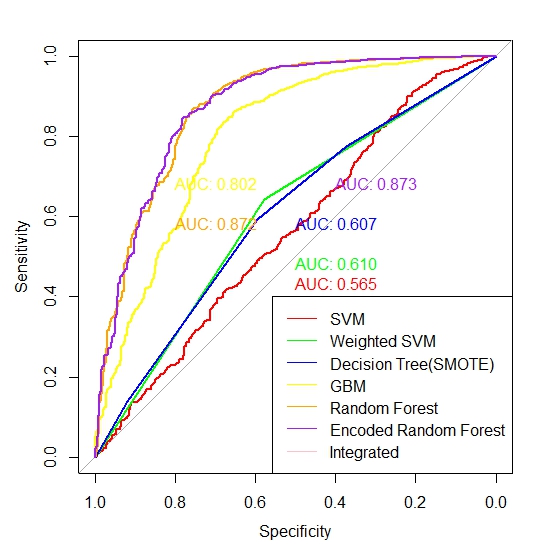
**Figure A3: Decision Tree with SMOTE**



**Figure A4: Gradient Boosted Machine**



**Figure A5: Random Forest**



**Figure A6: Random Forest with One-Hot Encoding (PURPLE)**

**References**

[1] https://www.kaggle.com/c/amazon-employee-access-challenge

[2] Chenn-Jung Huang, Yu-Wu Wang, Chih-Tai Guan, Heng-Ming Chen, and Jui-Jiun Jian”

Applications of Machine Learning to Resource Management in Cloud Computing.”

International Journal of Modeling and Optimization, Vol. 3, No. 2, April 2013

[3] Shijian Tang, Jiang Han and Yue Zhang, “Amazon Employee Access Control System

”, Stanford University/Department of Computer Science, Vol. 1, No. 1, 2014

[4] Hand, David J.; and Till, Robert J. (2001); A simple generalization of the area under the ROC curve for multiple class classification problems, Machine Learning, 45, 171–186.

[5] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, W. Philip Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique”, Journal of Artificial Intelligence Research 16 (2002) 321–357, Submitted 09/01; published 06/02

[6] Ho, Tin Kam (1998). "The Random Subspace Method for Constructing Decision Forests" (PDF). IEEE Transactions on Pattern Analysis and Machine Intelligence. 20 (8): 832–844. doi:10.1109/34.709601.

[7] [http://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm- machine-learning/](http://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-%20%20%20%20%20%20%20machine-learning/)

[8] <https://cran.r-project.org/web/packages/xgboost/vignettes/discoverYourData.html>

[9] Cortes, C.; Vapnik, V. (1995). "Support-vector networks". Machine Learning. 20 (3): 273–297. Doi: 10.1007/BF00994018