

Machine learning Data Competition 2020

Report I.

Shreyasvi Natraj (Team ML_B)

1 Introduction

For the given data competition, we were provided with data pertaining to previous advertisement campaigns as well as demographics of users who have been a part of the survey conducted.

For the objective of the given task, we are required to train a model that can be used in order to predict whether if a user is likely to have a “conversion” where a “conversion” refers to the user clicking on the advertisement and subscribing to the service.

Since the data provided is used in order to predict a categorical variable i.e. “conversion/y”, we initially started with a quick and dirty implementation of all the categorical classification models to check for the model which gives us the best accuracy out of the box. We later on transitioned to working in depth on each of the model to find the best model for the current dataset and then check which one provides us with the best accuracy under the best configuration.

We initially carried out with exploratory data analysis for the data provided to us by considering na values as NaN values.

2 Exploratory data analysis

We observed from this that it would not be a good idea to not consider the na values as NaN but as a separate level. However, based on similarity in between the classes, we can merge different levels of a factor into lesser number of levels so they are easier for our model to interpret.

2.1 Data Manipulation

We carried out all of the following tests in 3 different fashion: - Converting na values as NaN

- Converting the factor values into integer values
- Converting all the variables into a continuous variable format and checking

2.2 Interpretation

The main interpretation that we found from the given dataset were:

- Converting categorical variables into integer format tends to show a similar fashion to the current analysis being carried out.
- Removing the na values as NaN tends to take out a large portion of the data that might be useful for training the model. However, if we replaced “na” values with 0, the data becomes much more consistent.

We also observed that `time_spent`, `outcome_old` and `X3` tends to hold a very high significance when predicting conversion `y`. In order to try to check if the data becomes more consistent if the na values are replaced. We also planned on implementation of data imputation in order to fill up the missing values.

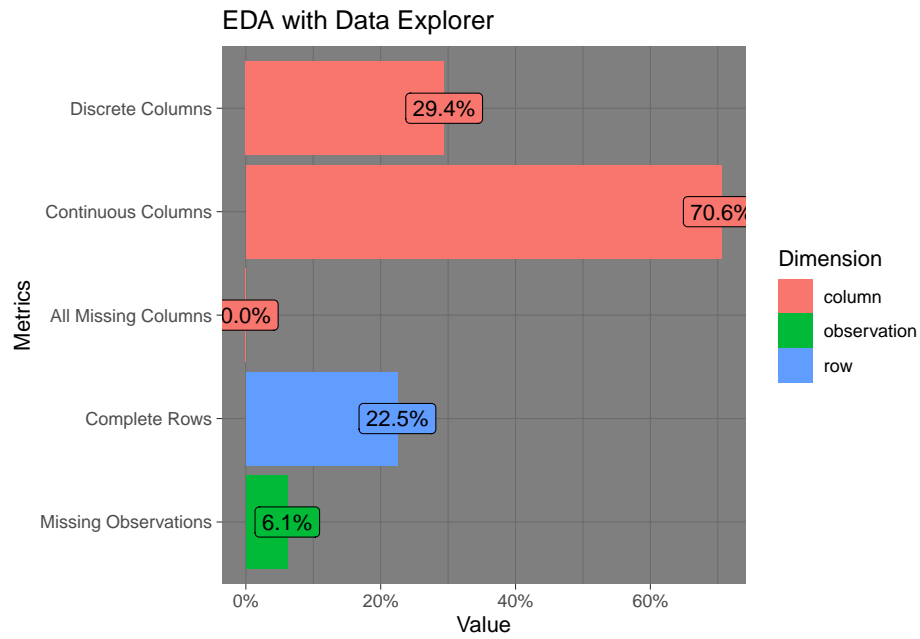


Figure 1: Data Distribution

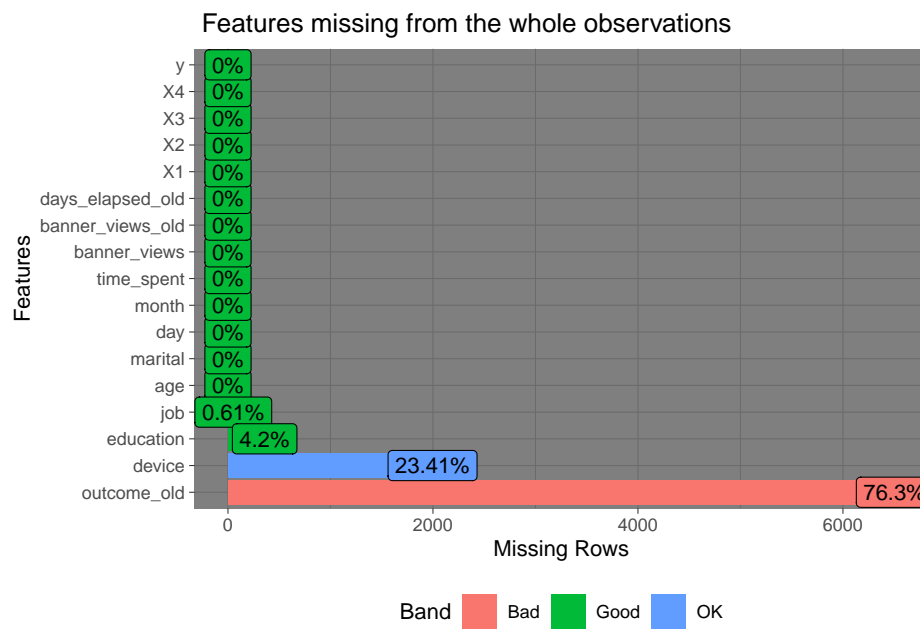


Figure 2: Missing Columns

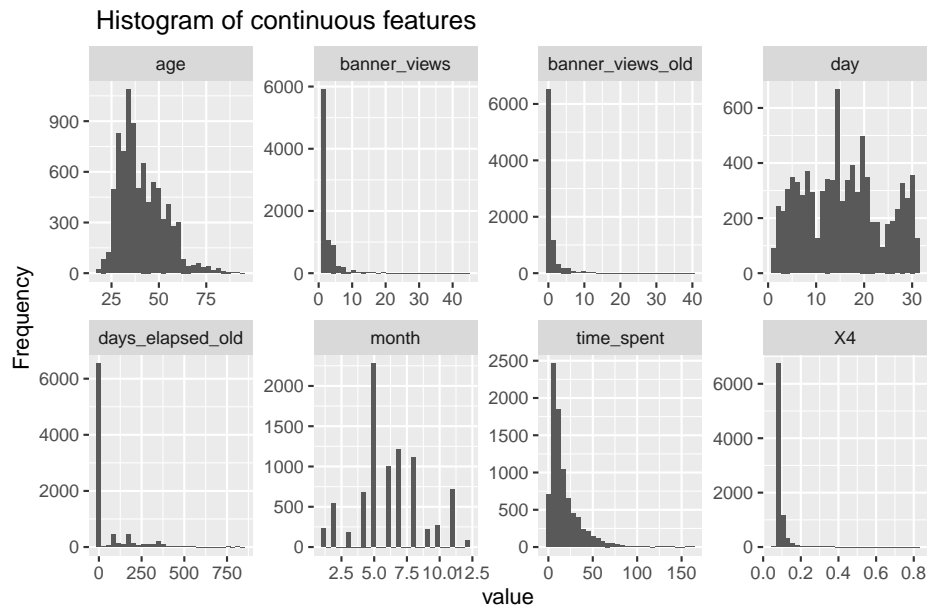


Figure 3: Continous Variables

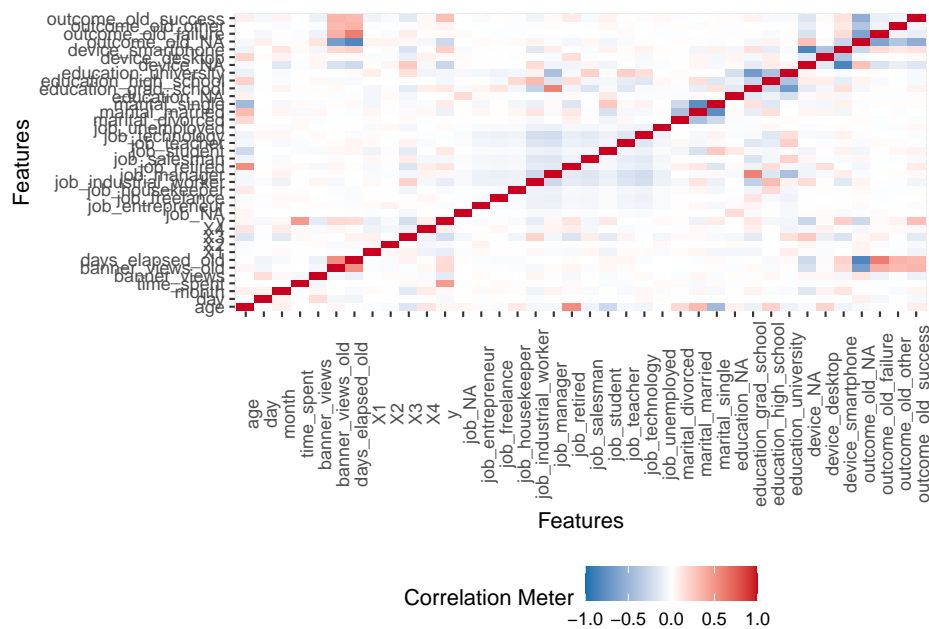


Figure 4: Correlation Plot

Model	Accuracy	Balanced Accuracy
kNN	80.53	80.90
Decision Trees	79.46	79.20
Naive Bayes	74.44	73.94
Logistic Regression	81.85	82.10
random forest	86.3	85.92
SVM	84.76	84.38

Table 1: Prediction Accuracy and Balanced Accuracy for different models

3 Models

We carried out a preliminary implementation of several different types of models which gave us the following result for prediction accuracy:

Based on the preliminary implementation of several models include kNN, SVM, random forest etc., we identified that random forest tends to give out the best prediction accuracy. We therefore explored random forest in order to improve the model to get the best possible accuracy for the same. We started by carrying out by implementing a GBM on the given dataset for 750 trees, 3 fold and 4 interaction depth.

3.1 Random Forest

Out of the box, random forest method provided us with a very good training as well as prediction accuracy on our current dataset. Therefore, the first approach was to fit a random forest model, which calculated its mean square error using the following formula.

$$MSE = 1/n \sum_{i=1}^n (fi - yi)^2,$$

where n is number of data points fi is the factor prediction made by the model and yi is the actual factor value.

We carried out cross validation for the same by using a 10 fold cross-validation.

```
## [1] "=====C5.0====="
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##      y_pred
```

```
##      0      1
```

```
## 0 1082  158
```

```
## 1  122  770
```

```
##
```

```
##              Accuracy : 0.8687
```

```
##              95% CI : (0.8536, 0.8827)
```

```
##      No Information Rate : 0.5647
```

```
##      P-Value [Acc > NIR] : < 2e-16
```

```
##
```

```
##              Kappa : 0.7317
```

```
##
```

```
## Mcnemar's Test P-Value : 0.03647
```

```
##
```

```
##              Sensitivity : 0.8987
```

```
##              Specificity : 0.8297
```

```
##      Pos Pred Value : 0.8726
```

```
##      Neg Pred Value : 0.8632
```

```
##           Prevalence : 0.5647
##           Detection Rate : 0.5075
##           Detection Prevalence : 0.5816
##           Balanced Accuracy : 0.8642
##
##           'Positive' Class : 0
##
```

We fit several different linear models to find the positive or negative dependency as well as significance of each variable with respect to y . We then used them in order to reduce the number of levels in the factor in order to improve the model further. Using this we were able to get around 86.87 training accuracy and 86.557 for the test set. The main changes to the dataset were as follows: - Replacing “na” values as NaN.

- Adding non-significant factor levels to “na” level.
- Replacing all factors into integer format (except y)
- Merging factor levels with negative and positive non-significant correlation into one level.

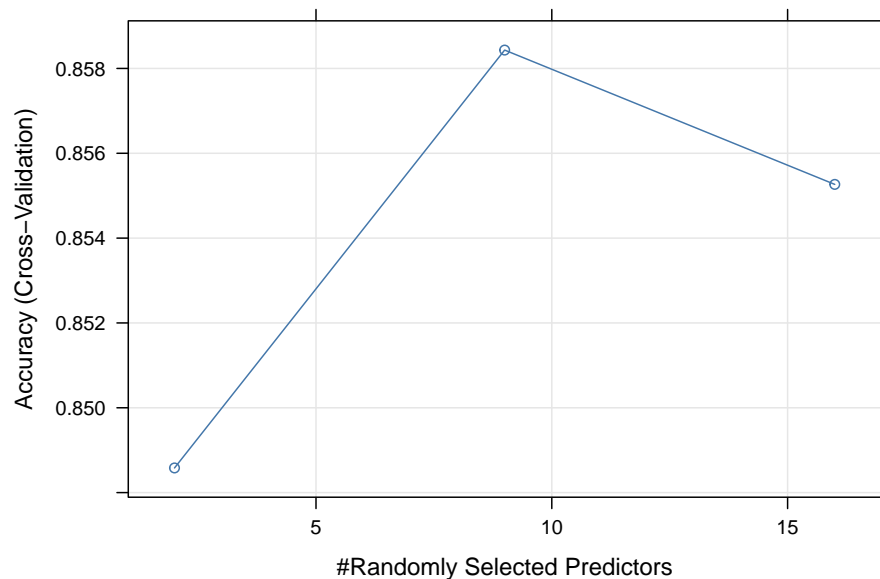


Figure 5: Random Forest Cross-Validated Accuracy

We also carried out imputation of the given dataset replacing the values of “na” with values generated using the *rfImpute()* function. However, we did not observe any increase in the prediction accuracy.

3.2 GBM Cross Validation

In order to identify the most significant variables, we also implemented a generalized boosted regression model (GBM) to the given dataset and cross validated again that time-spent as well as outcome old tends to have the most significance. A major result we also observed was that there seems to be specific age groups, *time_spent* values as well as other factor levels which tend to have very less influence on the prediction y . Therefore, we plan to convert the age group into categorical variable in the next step along with finetuning some other categorical variables to make the dataset simpler for the model to understand.

```
##           var      rel.inf
## time_spent      time_spent 49.6997337
```

```

## outcome_old          outcome_old 14.8535040
## month                month      8.6110871
## device                device     7.2223816
## X3                    X3         5.4987384
## age                  age         3.6024227
## days_elapsed_old     days_elapsed_old 2.9312266
## day                  day         2.1541162
## job                  job         1.5040134
## banner_views         banner_views 1.3335633
## X4                    X4         1.2479805
## X1                    X1         0.4514397
## education            education   0.3257785
## banner_views_old     banner_views_old 0.3055869
## marital              marital     0.2584273
## X2                    X2         0.0000000

```

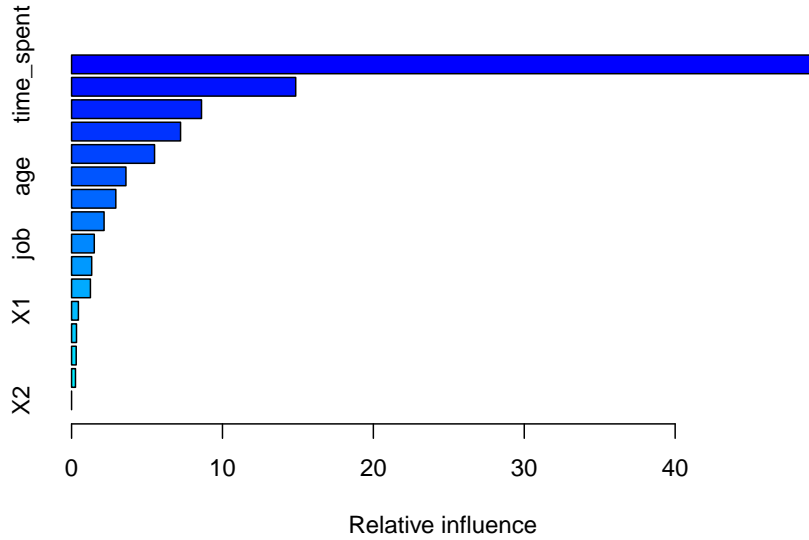


Figure 6: Relative influence(GBM)

3.3 C5.0 Implementation

While implementing several models, a good prediction accuracy was also observed in case of C5.0(87.15) over the given dataset. C5.0 uses the concept of entropy for measuring purity. The entropy of a sample of data indicates how mixed the class values are; the minimum value of 0 indicates that the sample is completely homogenous, while 1 indicates the maximum amount of disorder. The definition of entropy can be specified as:

$$S = \sum_{i=1}^c -p_i \cdot \log(p_i),$$

For a given segment of data (S), the term c refers to the number of different class levels, and p_i refers to the proportion of values falling into the class level i . We carry out the implementation of the C5.0 with and without winnowing. The winnow algorithm is a technique from machine learning for learning a linear classifier from labeled examples. We will be presently exploring more on the same in order to try to add interaction

and pre-process the dataset in order to increase the prediction accuracy further for the same. It applies the typical prediction rule for linear classifiers: If,

$$\sum_{i=1}^n w_i \cdot x_i > \theta, y = 1, \text{ else } 0$$

Here θ is a real number called threshold, w_i are the weights and x_i are the features, y is the prediction label as a factor.

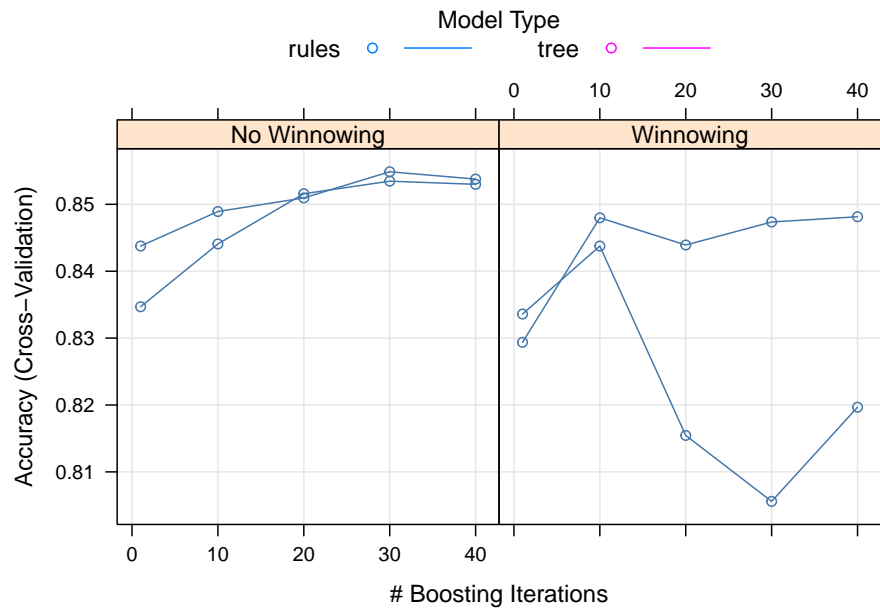


Figure 7: C5.0 Accuracy (With and without winnowing)

```
## [1] "=====Random Forest====="
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##      y_pred
```

```
##      0      1
```

```
## 0 1098  142
```

```
## 1  132  760
```

```
##
```

```
##              Accuracy : 0.8715
```

```
##              95% CI : (0.8565, 0.8854)
```

```
##      No Information Rate : 0.5769
```

```
##      P-Value [Acc > NIR] : <2e-16
```

```
##
```

```
##              Kappa : 0.7363
```

```
##
```

```
##      McNemar's Test P-Value : 0.5866
```

```
##
```

```
##              Sensitivity : 0.8927
```

```
##              Specificity : 0.8426
```

```
##      Pos Pred Value : 0.8855
```

```
##      Neg Pred Value : 0.8520
```

```
##      Prevalence : 0.5769
```

```

##          Detection Rate : 0.5150
##    Detection Prevalence : 0.5816
##          Balanced Accuracy : 0.8676
##
##          'Positive' Class : 0
##

```

4 Results

We obtained the best prediction accuracy from the C5.0 model(87.15%) followed by Random forest with the data manipulation (86.87%) by decreasing the number of factor levels. As a next step we will be:

- Exploring more over the predictions provided by ElasticNet as it tends to give a good training accuracy at initial stages.
- Looking forward to running the C5.0 with data manipulation
- Checking for other models that might give a better accuracy than the models presently used.

5 Tests

5.1 ElasticNet Implementation

We carried out implementation of ElasticNet in order to understand more regarding the dataset as well as identifying details regarding each variable and its dependency with the predictor.

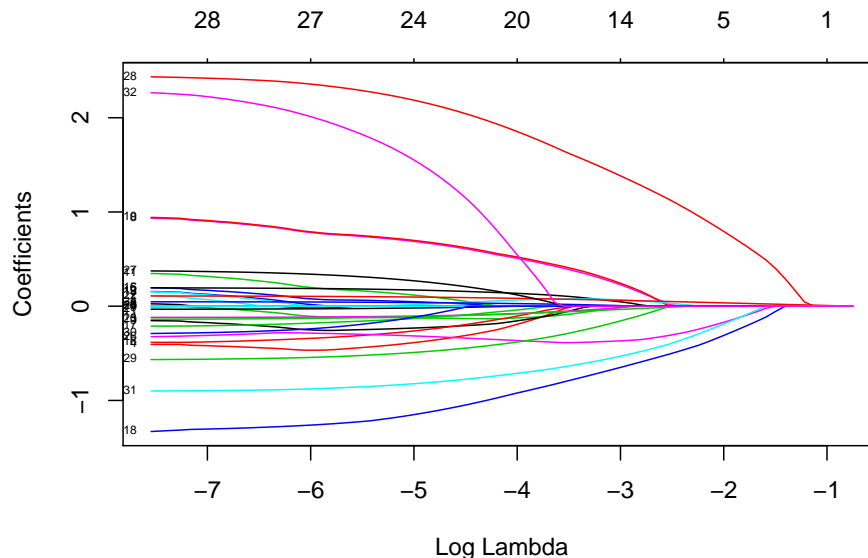


Figure 8: ElasticNet Dependency Chart

We were able to get around 82.45779 prediction accuracy and 81.48264 training accuracy using this method using a 10 fold cross validation which was the best across Ridge and Lasso models which were also implemented. Furthermore, when considering the training set as a matrix full of numerical variables or converting factors into numerical variables, this accuracy decreased by 2%. Therefore, we fit a matrix model initially to consider the factor variables in the correct manner and not loose accuracy during prediction.

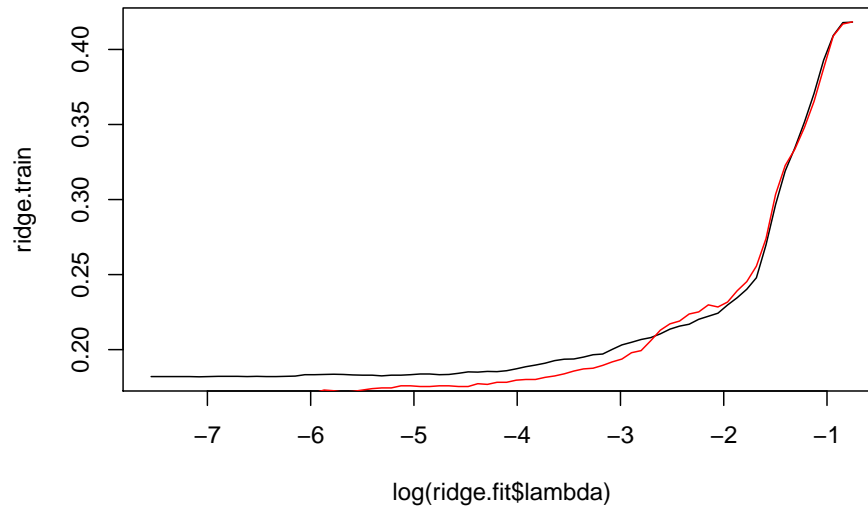


Figure 9: ElasticNet Error Rate

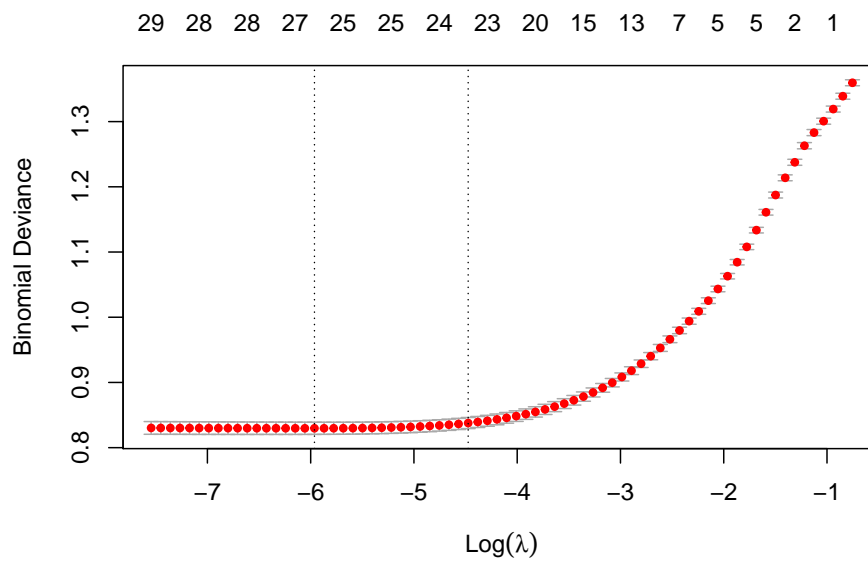


Figure 10: ElasticNet Crossvalidation Plot

5.2 Detailed kNN implementation

We also implemented a kNN at initial stage in order to identify if it could perform good on the given dataset. However, it was only able to obtain a prediction accuracy of 79.97 which was outperformed by random forest

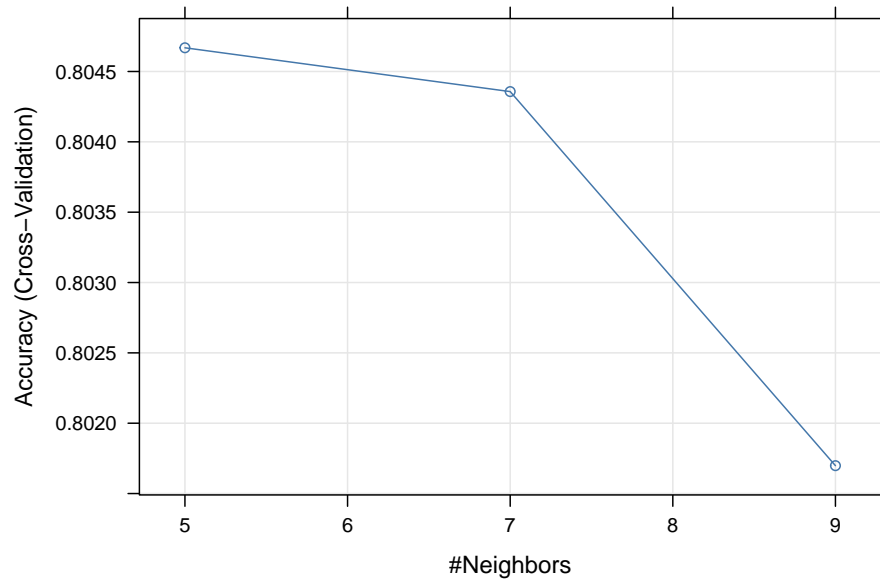


Figure 11: kNN Accuracy Plot

```
## [1] "=====Random Forest====="
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##      y_pred
```

```
##        0    1
```

```
## 0 1055  185
```

```
## 1  242  650
```

```
##
```

```
##              Accuracy : 0.7997
```

```
##              95% CI : (0.7821, 0.8165)
```

```
##      No Information Rate : 0.6083
```

```
##      P-Value [Acc > NIR] : < 2.2e-16
```

```
##
```

```
##              Kappa : 0.5847
```

```
##
```

```
##      McNemar's Test P-Value : 0.006728
```

```
##
```

```
##              Sensitivity : 0.8134
```

```
##              Specificity : 0.7784
```

```
##      Pos Pred Value : 0.8508
```

```
##      Neg Pred Value : 0.7287
```

```
##              Prevalence : 0.6083
```

```
##      Detection Rate : 0.4948
```

```
##      Detection Prevalence : 0.5816
```

```
##      Balanced Accuracy : 0.7959
```

```
##
```

```
##      'Positive' Class : 0
##
```

6 Annex

6.1 Lasso and Ridge Implementation

This is presently something we are carrying out in order to identify a good fit for the model. Here, $n_{folds}=10$ (Number of Folds)

```
## [1] "Ridge Implementation"
```

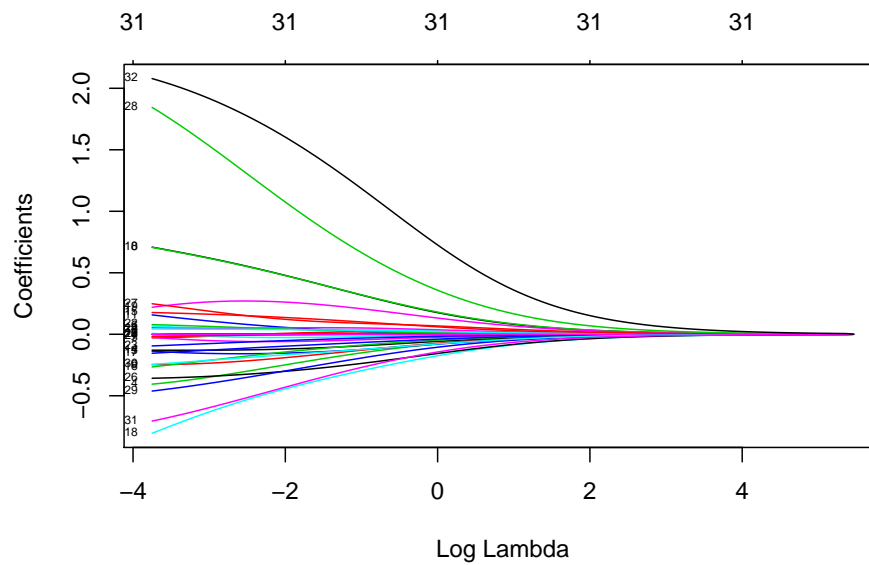


Figure 12: Ridge Glmnet Dependency chart

```
## [1] "Testing Accuracy: 0.818480300187617"
## [1] "Training Accuracy: 0.811072880825774"
```

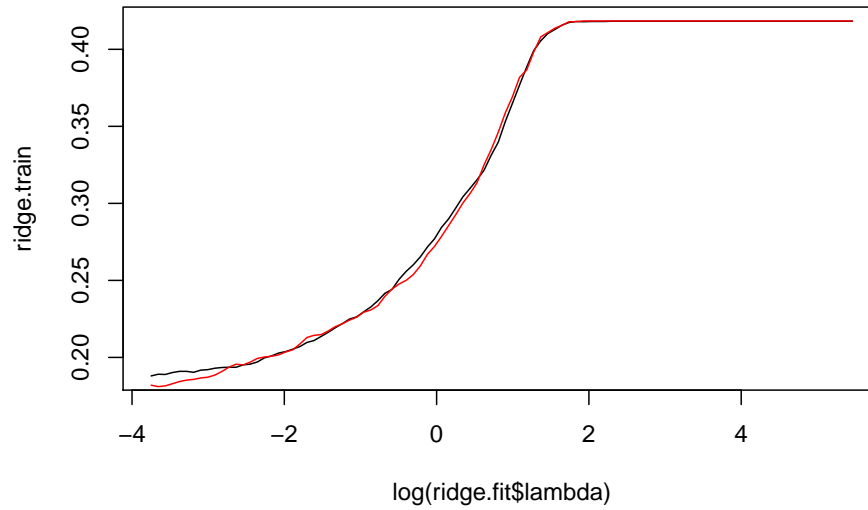
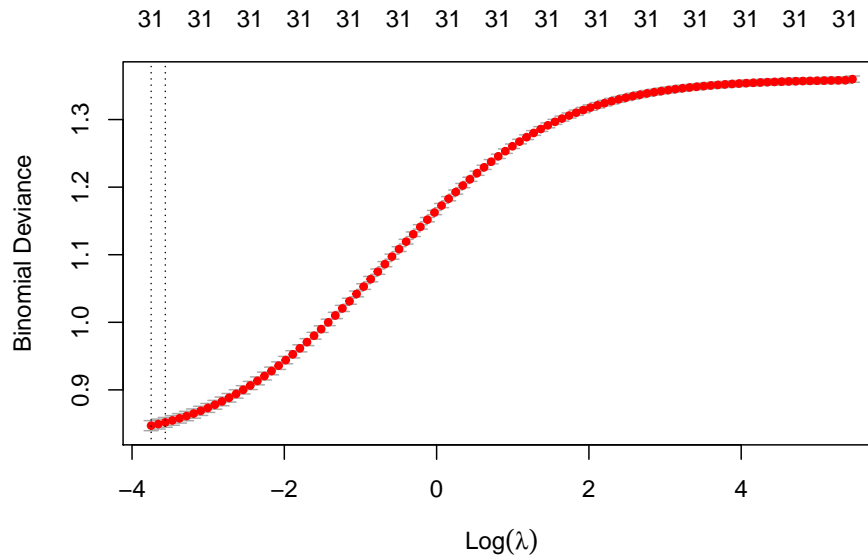


Figure 13: Ridge Glmnet Error Rate



Lasso Implementation seems to perform better compared to Ridge implementation. We then tried to adjust the α value to 0.5 (ElasticNet) to check if it gave better result. Moreover, based on the analysis done using GBM, we will be adjusting the input variables given to the current model to check the increase in their accuracy.

```
## [1] "Lasso Implementation"
## [1] "Testing Accuracy: 0.823639774859287"
## [1] "Training Accuracy: 0.814357209884267"
```

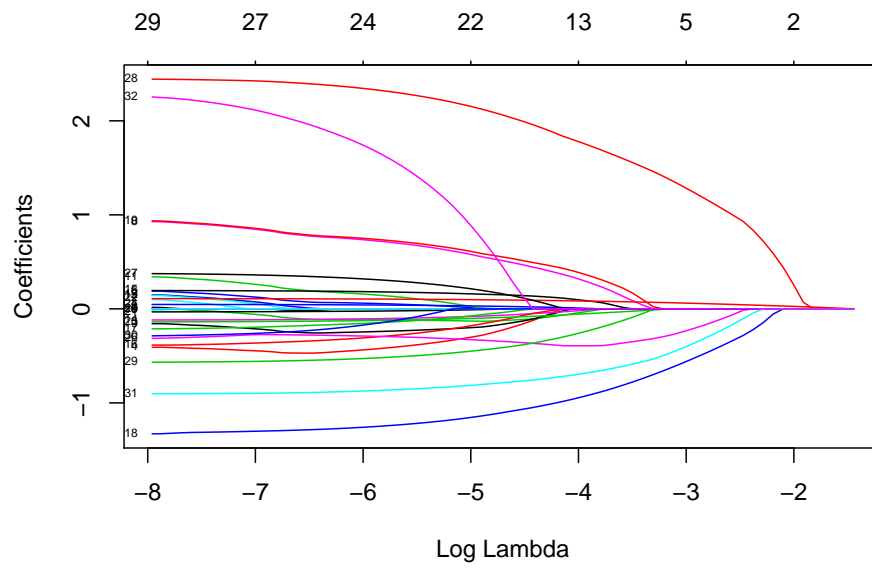


Figure 14: Lasso Glmnet Dependency Chart

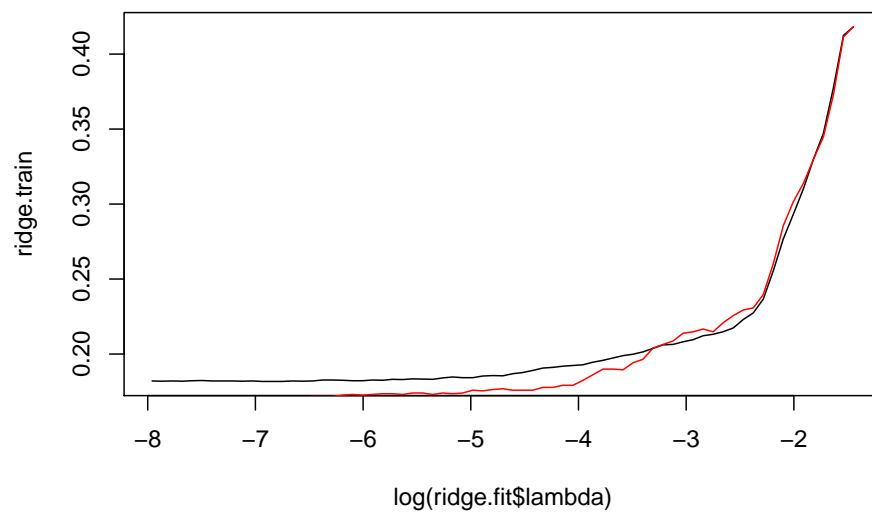


Figure 15: Lasso Glmnet Cross Validation

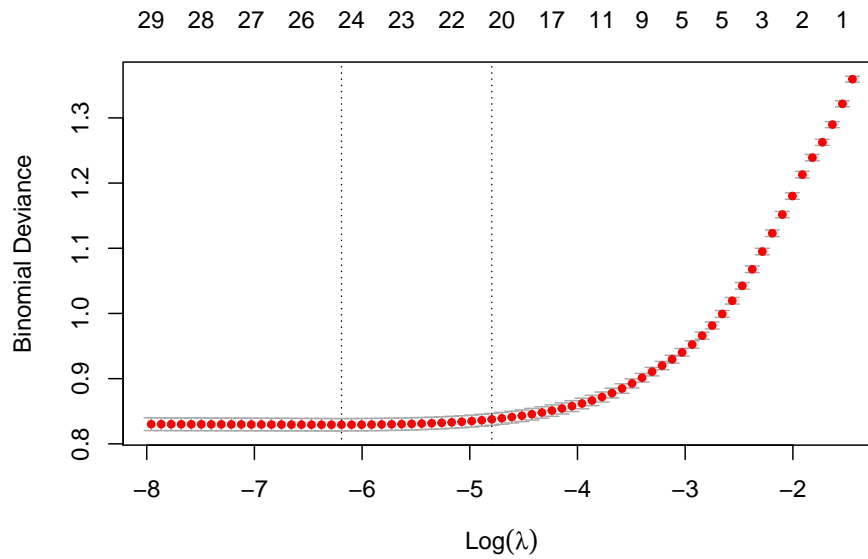


Figure 16: Lasso Glmnet Crossvalidation

6.2 Preliminary Implementation

We started by dividing the set into 75-35 percent split and running them through different machine learning models in a crude manner to check out of the box which model tends to perform best on the given dataset.
Support Vector Machine

```
## [1] "=====SVM====="
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##      y_pred
```

```
##      0      1
```

```
## 0 1083  157
```

```
## 1  168  724
```

```
##
```

```
##              Accuracy : 0.8476
```

```
##              95% CI : (0.8316, 0.8626)
```

```
##      No Information Rate : 0.5868
```

```
##      P-Value [Acc > NIR] : <2e-16
```

```
##
```

```
##              Kappa : 0.6862
```

```
##
```

```
##      McNemar's Test P-Value : 0.5791
```

```
##
```

```
##              Sensitivity : 0.8657
```

```
##              Specificity : 0.8218
```

```
##      Pos Pred Value : 0.8734
```

```
##      Neg Pred Value : 0.8117
```

```
##              Prevalence : 0.5868
```

```
##      Detection Rate : 0.5080
```

```
##      Detection Prevalence : 0.5816
```

```
##          Balanced Accuracy : 0.8438
##
##          'Positive' Class : 0
##
```

Random Forest Classification

```
## [1] "=====Random Forest====="
```

```
## Confusion Matrix and Statistics
##
##      y_pred
##      0      1
## 0 1092  148
## 1  144  748
##
##              Accuracy : 0.863
##              95% CI : (0.8477, 0.8774)
##      No Information Rate : 0.5797
##      P-Value [Acc > NIR] : <2e-16
##
##              Kappa : 0.7188
##
## Mcnemar's Test P-Value : 0.8606
##
##      Sensitivity : 0.8835
##      Specificity : 0.8348
##      Pos Pred Value : 0.8806
##      Neg Pred Value : 0.8386
##      Prevalence : 0.5797
##      Detection Rate : 0.5122
##      Detection Prevalence : 0.5816
##      Balanced Accuracy : 0.8592
##
##      'Positive' Class : 0
##
```

Logistic Regression

```
## [1] "=====Logistic Regression====="
```

```
## Confusion Matrix and Statistics
##
##      y_pred
##      0      1
## 0 1110  130
## 1  257  635
##
##              Accuracy : 0.8185
##              95% CI : (0.8015, 0.8346)
##      No Information Rate : 0.6412
##      P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 0.6194
##
## Mcnemar's Test P-Value : 1.504e-10
##
```

```
##          Sensitivity : 0.8120
##          Specificity : 0.8301
##          Pos Pred Value : 0.8952
##          Neg Pred Value : 0.7119
##          Prevalence : 0.6412
##          Detection Rate : 0.5206
##          Detection Prevalence : 0.5816
##          Balanced Accuracy : 0.8210
##
##          'Positive' Class : 0
##
```

Naive Bayes

```
## [1] "=====Naive Bayes====="
```

```
## Confusion Matrix and Statistics
##
##      y_pred
##      0      1
## 0 1020  220
## 1   325  567
##
##          Accuracy : 0.7444
##          95% CI : (0.7253, 0.7628)
##          No Information Rate : 0.6309
##          P-Value [Acc > NIR] : < 2.2e-16
##
##          Kappa : 0.4659
##
##  Mcnemar's Test P-Value : 8.394e-06
##
##          Sensitivity : 0.7584
##          Specificity : 0.7205
##          Pos Pred Value : 0.8226
##          Neg Pred Value : 0.6357
##          Prevalence : 0.6309
##          Detection Rate : 0.4784
##          Detection Prevalence : 0.5816
##          Balanced Accuracy : 0.7394
##
##          'Positive' Class : 0
##
```

Decision Tree

```
## [1] "=====Decision Trees====="
```

```
## Confusion Matrix and Statistics
##
##      y_pred
##      0      1
## 0 1064  176
## 1   262  630
##
##          Accuracy : 0.7946
##          95% CI : (0.7768, 0.8115)
```



```

##      No Information Rate : 0.622
##      P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 0.5721
##
##      McNemar's Test P-Value : 4.877e-05
##
##              Sensitivity : 0.8024
##              Specificity : 0.7816
##              Pos Pred Value : 0.8581
##              Neg Pred Value : 0.7063
##              Prevalence : 0.6220
##              Detection Rate : 0.4991
##      Detection Prevalence : 0.5816
##      Balanced Accuracy : 0.7920
##
##      'Positive' Class : 0
##

```

kNN

```

## [1] "=====KNN=====
## Confusion Matrix and Statistics
##
##      y_pred
##          0      1
## 0 1107  133
## 1   282  610
##
##              Accuracy : 0.8053
##              95% CI : (0.7879, 0.822)
##      No Information Rate : 0.6515
##      P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 0.5904
##
##      McNemar's Test P-Value : 3.729e-13
##
##              Sensitivity : 0.7970
##              Specificity : 0.8210
##              Pos Pred Value : 0.8927
##              Neg Pred Value : 0.6839
##              Prevalence : 0.6515
##              Detection Rate : 0.5192
##      Detection Prevalence : 0.5816
##      Balanced Accuracy : 0.8090
##
##      'Positive' Class : 0
##

```