Machine learning Data Competition 2020

Report I.

Shreyasvi Natraj

1 Introduction

For the given data competition, we were provided with data pertaining to previous adverstisement 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 planned to use do a quick an dirty implementation of the following models to check for their accuracy:

K-Nearest Neighbours Random Forest *LDA, QDA & C5.0* Supported Vector Machines *Logistic Regression

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 Interpretation

We also carried out the same process of data analysis after converting categorical variables into inteager format which tend to show a similar fashion to the current analysis being carried out. However, we replaced "na" values with 0 which made the data much more consistent. We also observed that `time_spent`,`outcome_old` and `X3` tends to hold a very high significance when predicting conversion `y`.

2.2 Data Distribution

```
## Rows: 8,526
## Columns: 17
## $ age              <int> 35, 42, 38, 71, 37, 26, 27, 28, 57, 44, 34, 26, 33...
## $ job              <fct> manager, manager, industrial_worker, retired, unem...
## $ marital          <fct> single, married, married, married, single, married...
## $ education        <fct> grad_school, grad_school, university, high_school,...
## $ device           <fct> NA, smartphone, NA, smartphone, desktop, smartphon...
## $ day              <int> 30, 17, 14, 13, 27, 17, 30, 7, 26, 28, 12, 2, 6, 2...
## $ month            <int> 5, 7, 5, 11, 4, 7, 6, 5, 5, 12, 8, 12, 5, 5, 11, 2...
## $ time_spent       <dbl> 37.65, 39.25, 10.50, 8.80, 20.80, 57.00, 3.75, 10....
## $ banner_views     <int> 1, 1, 2, 2, 2, 3, 5, 1, 5, 1, 1, 1, 1, 2, 1, 2, 1,...
## $ banner_views_old <int> 0, 0, 0, 1, 3, 0, 0, 1, 0, 0, 0, 0, 3, 0, 0, 0, 5,...
```

```
## $ days_elapsed_old <int> -1, -1, -1, 98, 179, -1, -1, 339, -1, -1, -1, -1, ...
## $ outcome_old      <fct> NA, NA, NA, success, other, NA, NA, other, NA, NA,...
## $ X1               <int> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,...
## $ X2               <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,...
## $ X3               <int> 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0,...
## $ X4               <dbl> 0.07793293, 0.07283969, 0.07607176, 0.09640840, 0....
## $ y                <int> 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1,...
```

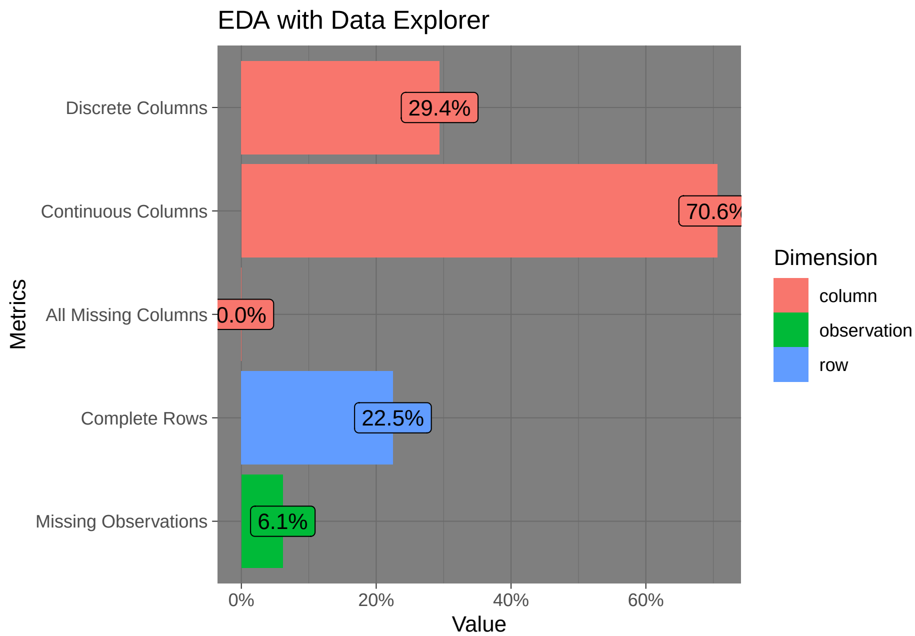


Figure 1: Data Distribution

2.3 Missing Columns

2.4 EDA for Continuous variables

2.5 Correlation Plot

2.6 EDA for Categorical

For more sophisticated graphs, that span over multiple pages, see function `ggarrange()` from `ggpubr` package (see link).

For good-looking colors, have a look at the Paul Tol's palette https://personal.sron.nl/~pault/.

2.7 Tables

To display a table, look at the `kable()` function from `knitr` package. Also, consider the `kableExtra` package for more sophisticated options (see link). In Table 1, we show an example that uses both `kable` and `kableExtra`.

You can reference a table by putting the code `\\label{tab:tblname}` inside the caption. See code below. Then, you see that the reference works (see Table 1).

```
# Prepare data to put in the table
dat2 <- mtcars %>%
  group_by(cyl) %>%
```

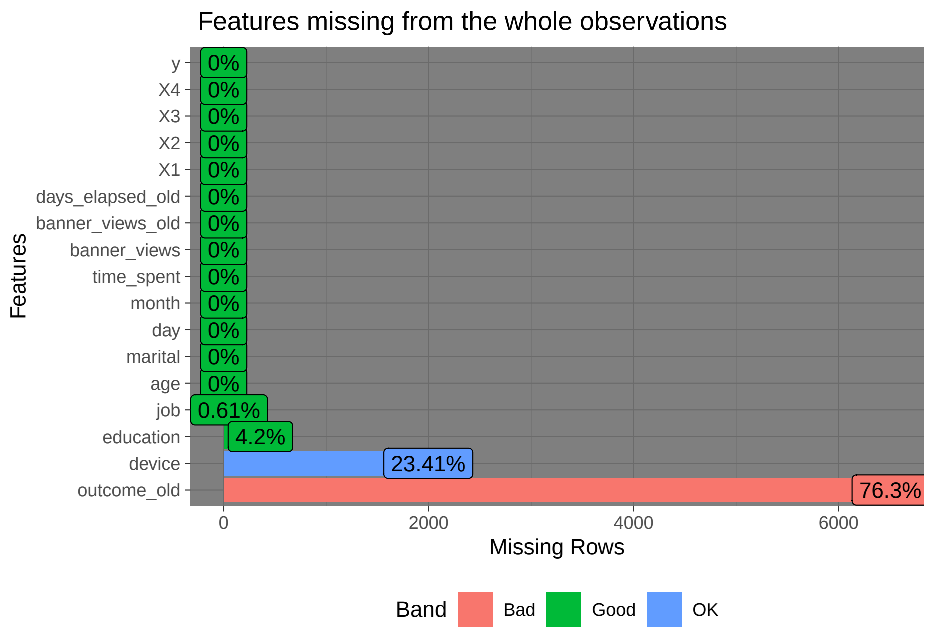


Figure 2: Missing Columns

```
  summarise(Average = mean(mpg), Max = max(mpg), Sqrt = sum(sqrt(mpg)))

# Print table
capt <- paste("\\label{tab:tblname}Average and ",
              "maximum miles per gallon for each number of cylindyers class.")
kable(dat2,
      format = "latex",
      longtable = F,
      booktabs = T,
      digits = 2,
      caption = capt) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

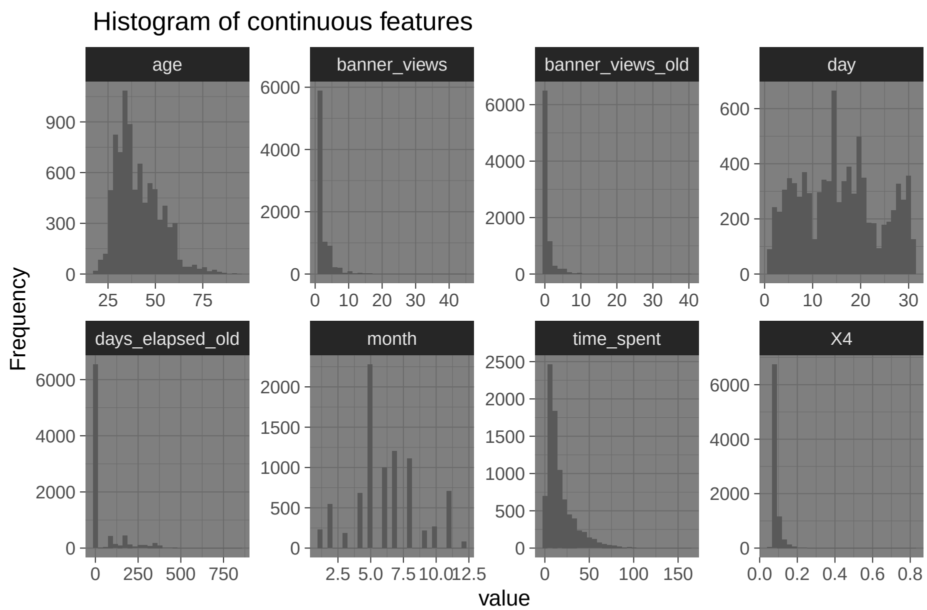
Table 1: Average and maximum miles per gallon for each number of cylindyers class.

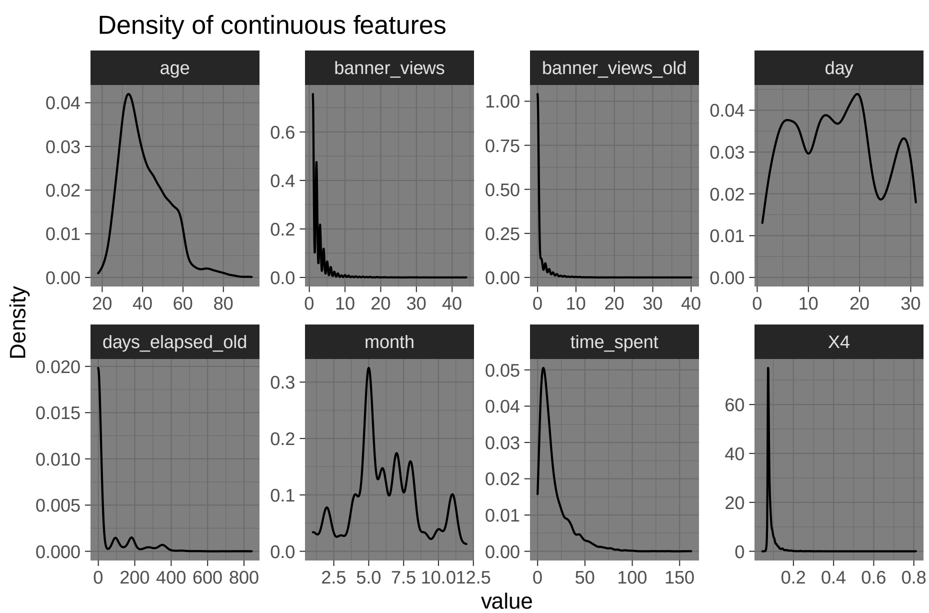
cyl	Average	Max	Sqrt
4	26.66	33.9	56.62
6	19.74	21.4	31.08
8	15.10	19.2	54.21

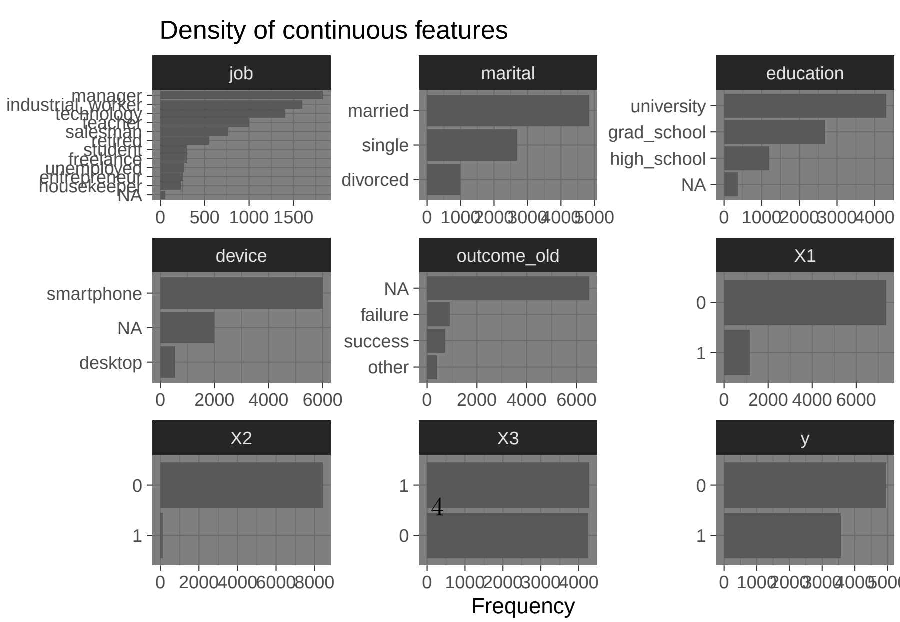
If you want to manually insert the values in the table, you can do it, too (see Table 2).

Table 2: Number of different levels and the number of predictors that have this amount of levels.

	Col 1	Col 2	Col 3	Col 4
Number of different values	2	4	12	> 300
Number of predictors	...	...	...	...


Histogram of continuous features
age
banner_views
banner_views_old
day
days_elapsed_old
month
time_spent
X4
Frequency
value


Density of continuous features
age
banner_views
banner_views_old
day
days_elapsed_old
month
time_spent
X4
Density
value


Density of continuous features
job
manager
industrial_worker
technology
teacher
salesman
retired
student
freelance
unemployed
entrepreneur
housekeeper
NA
marital
married
single
divorced
education
university
grad_school
high_school
NA
device
smartphone
NA
desktop
outcome_old
NA
failure
success
other
X1
X2
X3
y
Frequency

3 Models

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.

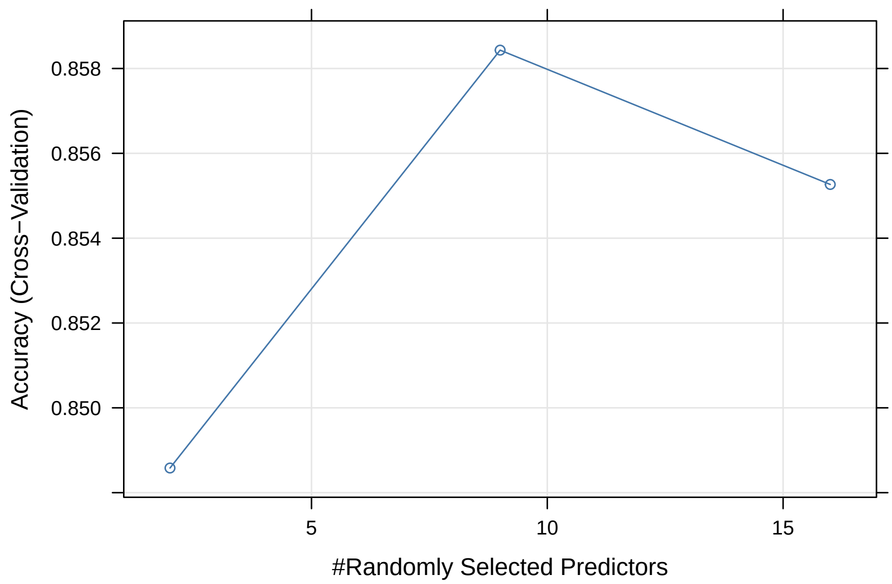
3.2 Implementation

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

```
## 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 depdendency 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.

3.3 Random Forest Plot



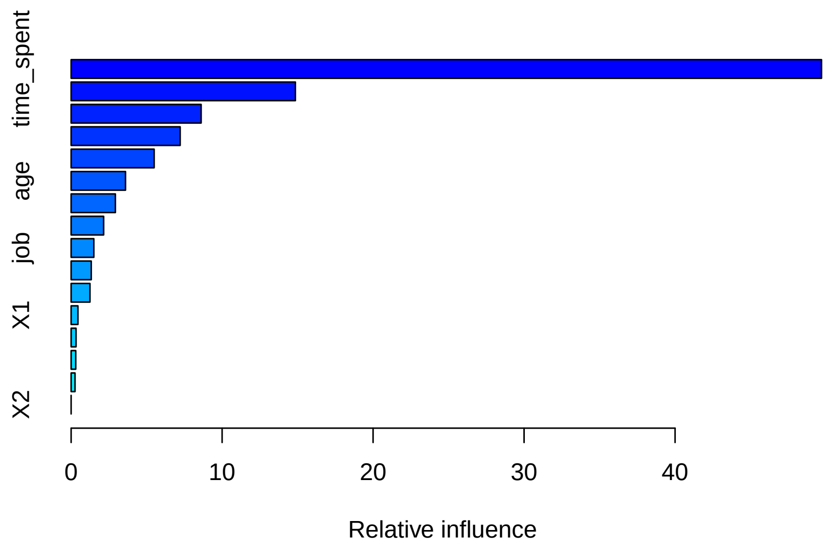
3.4 GBM Implementation

```
#=============================================================================================
rm(list=ls())
#GBM to find most relevant variables
library("gbm")
set.seed(77850)
dataset <- read.csv('train.csv')
gbm.fit <- gbm(y~.,
               distribution="bernoulli",
               data=dataset,
               n.trees=750,
               interaction.depth=4,
               shrinkage=0.01,cv.folds = 3)

summary(gbm.fit)
```

```
##                                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
```



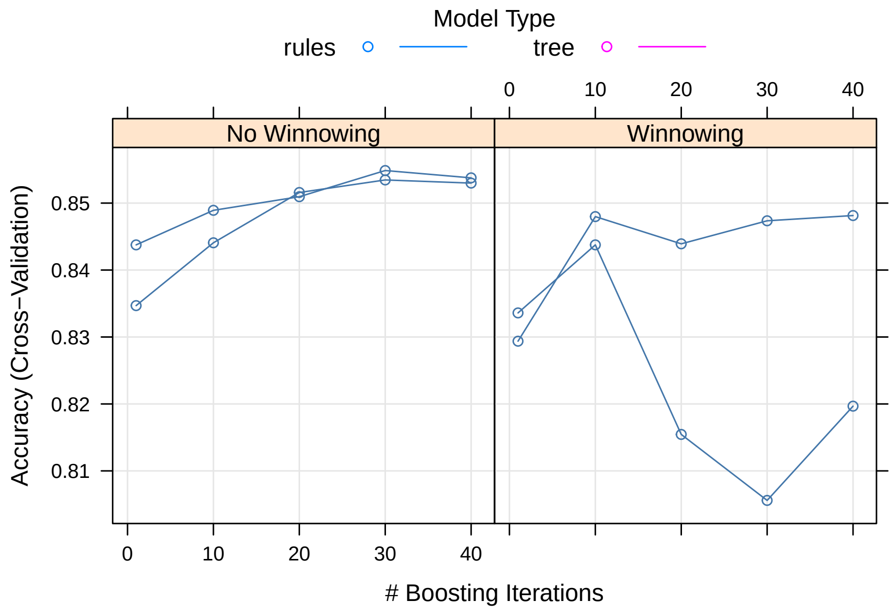
4 C5.0 Implementation

While implementing several models, a good prediction accuracy was also observed in case of C5.0 and got a prediction accuracy of 87.15 over the given dataset.

```
## [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
##
```



5 Results

We obtained the best prediction accuracy from the C5.0 model followed by Random forest where we carried out data manipulation by decreasing the number of factor levels.

6 Tests

6.1 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

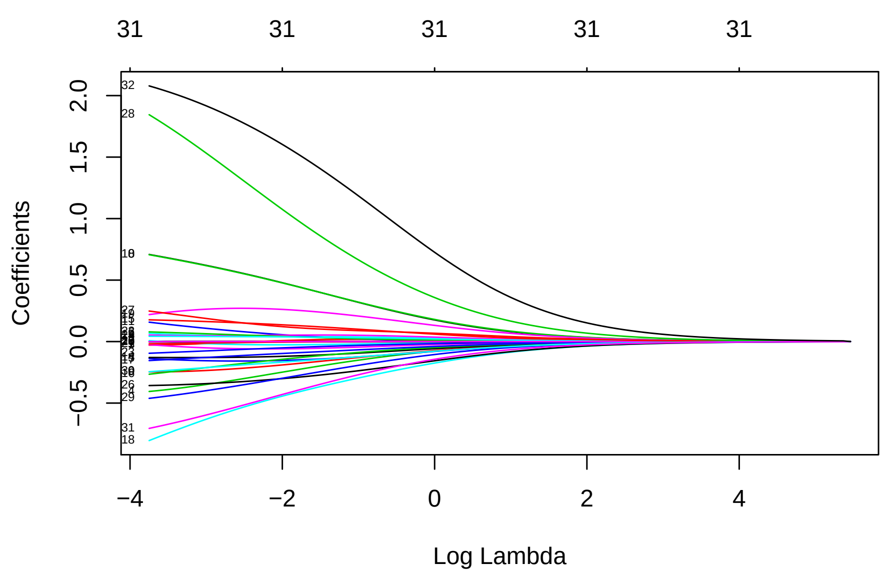
```
## [1] "====================================KNN===================================="
## Confusion Matrix and Statistics
##
##    y_pred
##        0    1
##   0 1107  133
##   1  282  610
##
##                Accuracy : 0.8053
##                  95% CI : (0.7879, 0.822)
##     No Information Rate : 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
```

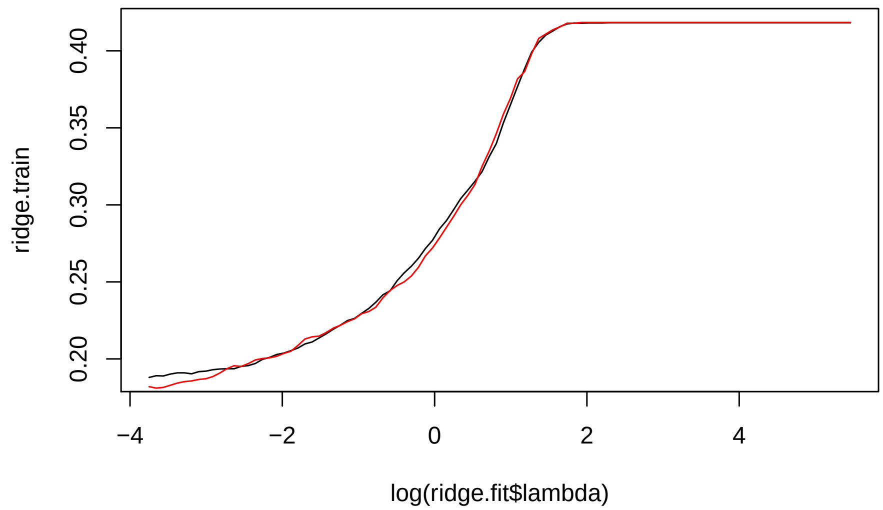
```
##    Detection Prevalence : 0.5816
##       Balanced Accuracy : 0.8090
##
##        'Positive' Class : 0
##
```

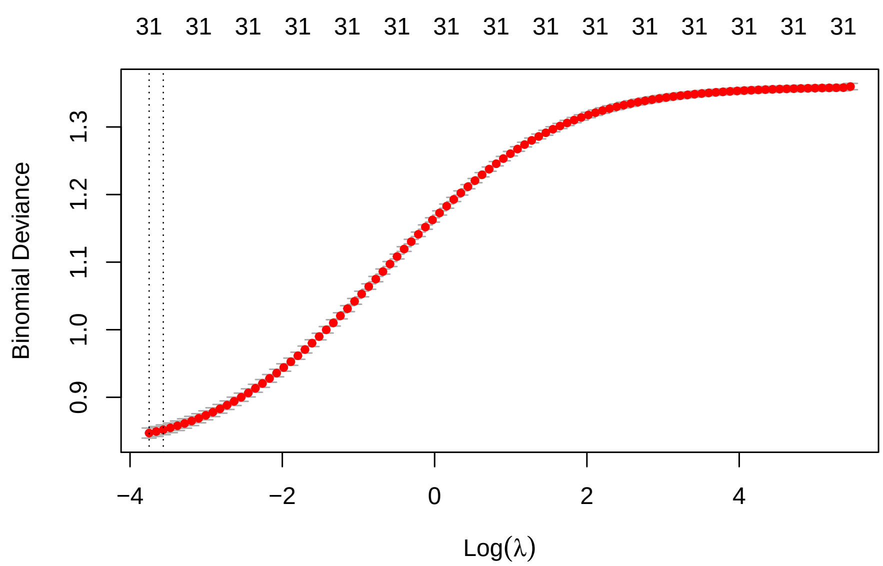
7 Lasso, Ridge and ElasticNet Implementation

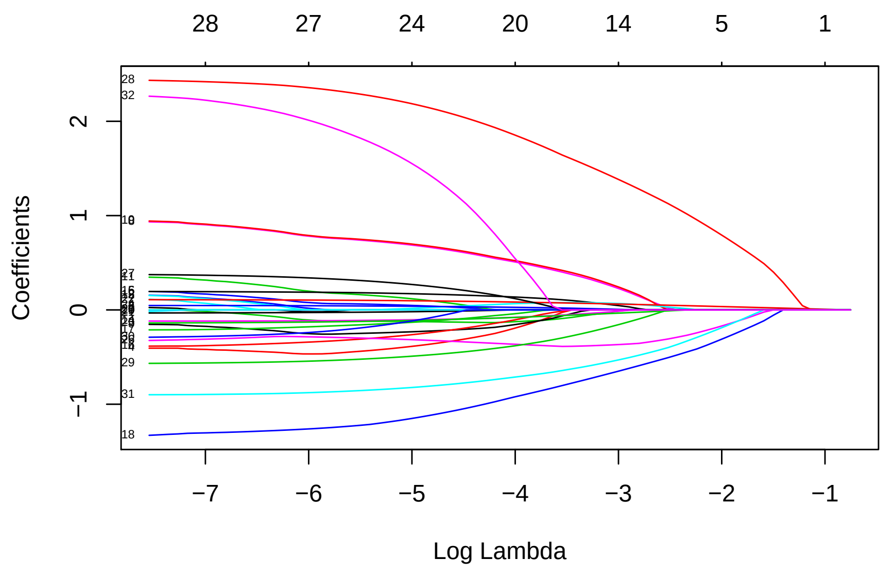
This is presently something we are carrying out in order to identify a good fit for the model. Here, $nfolds$=10 (Number of Folds)

```
## [1] "Ridge Implementation"
## [1] "Testing Accuracy"
## [1] 0.8184803
## [1] "Training Accuracy"
## [1] 0.8110729
## [1] "ElasticNet Implementation"
## [1] "Testing Accuracy"
## [1] 0.8245779
## [1] "Training Accuracy"
## [1] 0.8148264
## [1] "Lasso Implementation"
## [1] "Testing Accuracy"
## [1] 0.8236398
## [1] "Training Accuracy"
## [1] 0.8143572
```

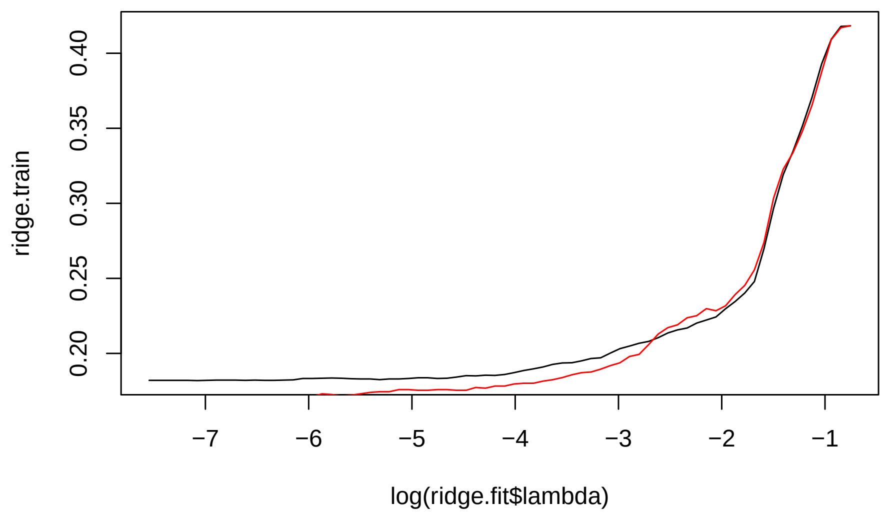



ridge.train
0.20
0.25
0.30
0.35
0.40
−4
−2
0
2
4
log(ridge.fit$lambda)

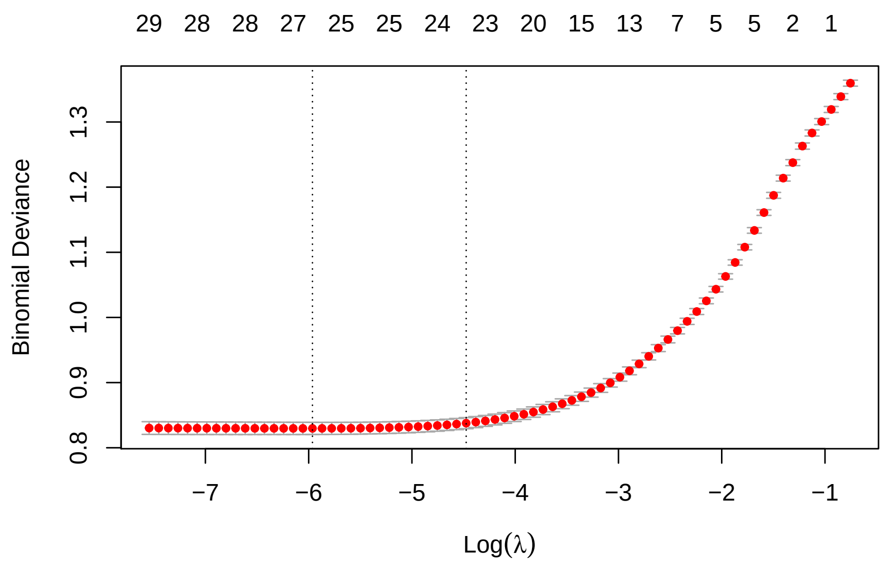

31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
Binomial Deviance
0.9
1.0
1.1
1.2
1.3
−4
−2
0
2
4
Log(λ)

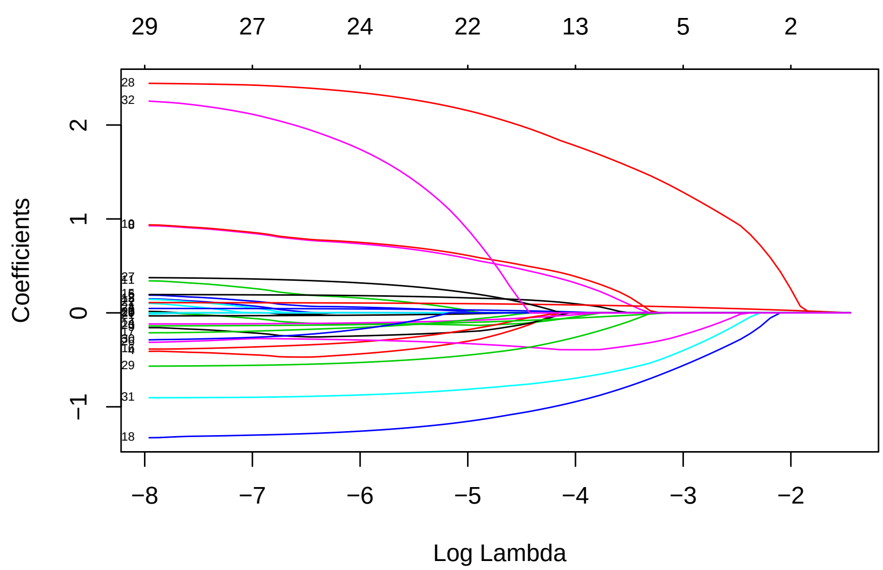


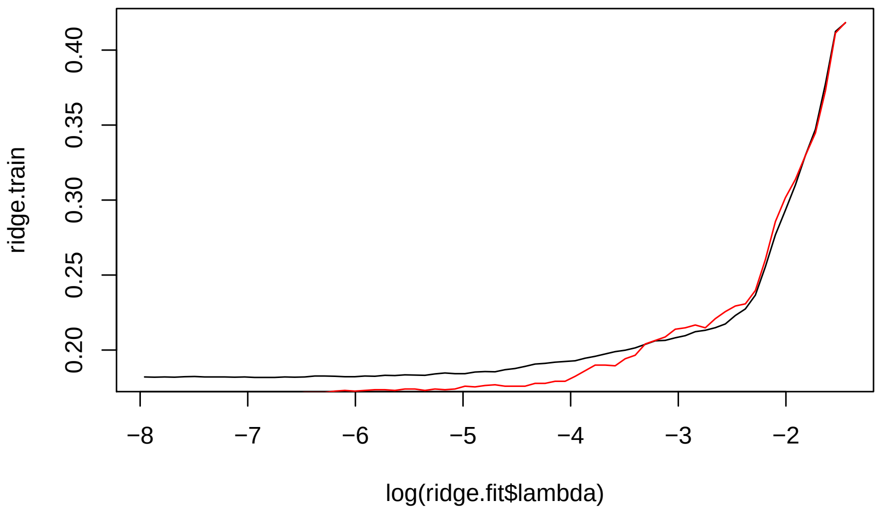
28
27
24
20
14
5
1
Coefficients
Log Lambda

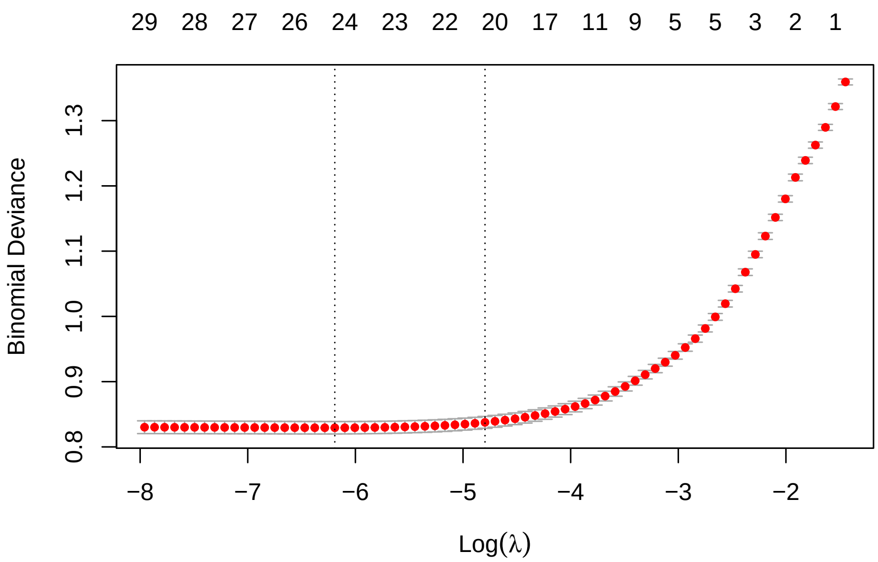


ridge.train
log(ridge.fit$lambda)


29 28 28 27 25 25 24 23 20 15 13 7 5 5 2 1
Binomial Deviance
0.8
0.9
1.0
1.1
1.2
1.3
−7 −6 −5 −4 −3 −2 −1
Log(λ)


29 27 24 22 13 5 2
Coefficients
−1
0
1
2
−8 −7 −6 −5 −4 −3 −2
Log Lambda





8 Detailed kNN implementation

```
#===============================================================================================
#Accuracy of the KNN model at different k values
rm(list=ls())
library(caret)
set.seed(123)
classSim <- read.csv('train.csv')
classSim$y = factor(classSim$y, levels = c(0, 1))

nfolds <- 10
```

	Training error	CV error	Public Leaderboard error (if available)
kNN	...	...	...
Ridge	...	...	...
lasso	...	...	...
ElasticNet	...	...	...
random forest	...	...	
SVM	...	...	
LDA	...	...	
QDA	...	...	
C5.0	...	...	

Table 3: Training and CV error of the different models.

```
trControl <- trainControl(method  = "cv",
                          number  = nfolds)
max_k <- 100

#Use this function to plot graphs for all the possible models used
fit <- train(form = y ~ .,
             data = classSim,
             method     = "knn", #can be changed here to 17 other configurations including random fores
             trControl  = trControl,
             metric     = "Accuracy")

palette = c(tolBlue = "#4477AA",
            tolRed = "#EE6677",
            tolGreen = "#228833",
            tolYellow = "#CCBB44",
            tolCyan = "#66CCEE",
            tolPurple = "#AA3377",
            tolGrey = "#BBBBBB") %>% unname()
plot(fit, col = palette[1])
```