Predicting the Hazardeous Nature of NEOs (Near Earth Objects)

Surya Ramesh

13/04/2021

Contents

1 Introduction

Near-Earth objects (NEOs) are asteroids or comets that orbit the Sun and whose orbits come close to that of Earth's. Of the more than 600,000 known asteroids in our Solar System, more than 20,000 are NEOs(The European Space Agency,2021). The aim of this coursework is to classify, potentially hazardous asteroids and comets, according to the risk of impacting earth.

The Target Variable: 'is_potentially_hazardous_asteroid'

Target Type: Logical (TRUE or FALSE)

Task: Classification

As the data complied consists of 53 attributes, a code book of variables is provided at this Onedrive link below , rather than printing it here.

https://liverguac-my.sharepoint.com/:w:/g/personal/s_ramesh1_rgu_ac_uk/EUJzGC7hE4tDushEcgVrN1YBdu4zIUYUxu3kSYWsS1KJew?e=cGFCLL.

Inspiration for this coursework came from a personal interest in space exploration programs and a Kaggle project from three years ago - link: https://www.kaggle.com/shrutimehta/nasa-asteroids-classification.

2 Preparation of Data and Exploratory Analysis

This section contains:

- Data Collection
- Load the data
- Data Preparation,cleaning and initial exploration

2.1 Data Collection

Data source:

The dataset was compiled using the NASA Near Earth Object Web Service (NeoWs) which is accessible from https://api.nasa.gov/ under the heading "Asteroids - NeoWs".

In order to obtain full access this API, one has to register with NASA and obtain a personalised API key. The query is limited to a 7-day date range and the number of queries that can be made is limited to 1000 queries per hour. A limited demo version can be accessed at:

https://api.nasa.gov/neo/rest/v1/feed?start_date=2021-02-12&end_date=2021-02-05&detailed=true&api_key=DEMO_KEY

The data:

The dataset was extracted from the content part of the JSON object that is received through the API for the Near Earth Objects (NEOs).

As this API is not accessible using the 'nasadata' package, it has to be compiled by calling queries. For this coursework, the dataset is going to be compiled for the last two years, using RStudio in 'Data_API_NASA.rmd' as shown in figure below.

The layout of entire coursework in terms of the two submitted files is in Figure 1.

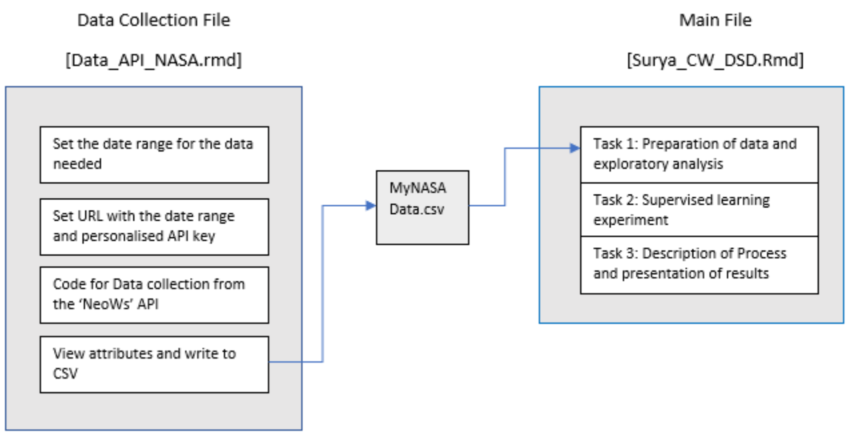

Data Collection File
[Data_API_NASA.rmd]
Set the date range for the data needed
Set URL with the date range and personalised API key
Code for Data collection from the 'NeoWs' API
View attributes and write to CSV
MyNASA Data.csv
Main File
[Surya_CW_DSD.Rmd]
Task 1: Preparation of data and exploratory analysis
Task 2: Supervised learning experiment
Task 3: Description of Process and presentation of results

Figure 1: The layout of the two rmd files

2.2 Load and view the files

```
# Loading the output from the data collection program,'MyNASAData.csv' into R

NASAoriginal<-read.csv(file= "../Data/MyNASAData.csv",stringsAsFactors = TRUE,
                          header = TRUE,sep = ",")

#"is_potentially_hazardous_asteroid" is read in as a boolean-change to factor

NASAoriginal$is_potentially_hazardous_asteroid<- as.factor(
                               NASAoriginal$is_potentially_hazardous_asteroid)
```

The data is loaded using read.csv and the class variable chosen for this data, 'is__potentially__hazardous__asteroid' is ensured to be a factor.

2.3 Data Preparation,cleaning and initial exploration

This section contains:

- Basic Checks: dimensions,column names
- Removal of unwanted columns that do not relate to the class
- Check the summary and remove other unnecessary columns
- Check the correlation matrix and check for further removals
- Check distribution of Class

2.3.1 Basic checks for the dataset

Table 1: Dimensions of Original NASA Dataset

Dimension	Number
Instances	12579
Attributes	53

The original Dataset has 53 attributes and 12579 instances.

```
# show all the column headings, output hidden for clarity

names(NASAoriginal)

# making a copy to work from

MyNASAdataset <- NASAoriginal
```

2.3.2 Remove unwanted columns

It can be seen that the there is duplication in the attributes . Some contain the same information with different units.There is also information that are not needed for the task at hand, like the url and IDs.These can be removed from the Dataset.

```
#remove name related information
MyNASAdataset$id <- NULL
MyNASAdataset$neo_reference_id <- NULL
MyNASAdataset$name <- NULL
MyNASAdataset$orbital_data.orbit_id <- NULL
MyNASAdataset$X <- NULL

# remove links
MyNASAdataset$nasa_jpl_url <- NULL
MyNASAdataset$links.self <- NULL
```

```
# remove duplicate information in different units
MyNASAdataset$estimated_diameter.meters.estimated_diameter_min <- NULL
MyNASAdataset$estimated_diameter.meters.estimated_diameter_max <- NULL

MyNASAdataset$estimated_diameter.miles.estimated_diameter_min <- NULL
MyNASAdataset$estimated_diameter.miles.estimated_diameter_max <- NULL

MyNASAdataset$estimated_diameter.feet.estimated_diameter_min <- NULL
MyNASAdataset$estimated_diameter.feet.estimated_diameter_max <- NULL

MyNASAdataset$relative_velocity.kilometers_per_hour <- NULL
MyNASAdataset$relative_velocity.miles_per_hour<- NULL

MyNASAdataset$miss_distance.astronomical <- NULL
MyNASAdataset$miss_distance.lunar <- NULL
MyNASAdataset$miss_distance.miles <- NULL

names(MyNASAdataset)            # show remaining column heading,output hidden
```

After removals, the number of attributes is now 35.

2.3.3 Check the summary and remove unnecessary columns

Looking at the summary , additional attributes can be removed:

- those containing class description
- those containing dates
- those containing the same information in all rows

```
summary(MyNASAdataset)               # ouput hidden for clarity of report
```

```
# remove attributes

MyNASAdataset$orbital_data.orbit_class.orbit_class_description <- NULL
MyNASAdataset$orbital_data.orbit_class.orbit_class_description <- NULL
MyNASAdataset$epoch_date_close_approach<- NULL
MyNASAdataset$close_approach_date_full<- NULL
MyNASAdataset$close_approach_date<- NULL

MyNASAdataset$orbital_data.orbit_class.orbit_class_type<- NULL
MyNASAdataset$orbital_data.first_observation_date<- NULL
MyNASAdataset$orbital_data.last_observation_date <- NULL
MyNASAdataset$orbital_data.orbit_determination_date <- NULL

MyNASAdataset$orbiting_body <- NULL        # always earth
MyNASAdataset$orbital_data.equinox <- NULL # always J2000:12579
MyNASAdataset$is_sentry_object <- NULL     #always FALSE

MyNASAdataset$orbital_data.orbit_class.orbit_class_range <- NULL
```

From summary again, missing values can be seen in orbital_data.data_arc_in_days.

```
# check for missing values

MyNASAdataset[!complete.cases(MyNASAdataset),] #output hidden for report clarity
```

It can be seen that there are 63 missing values and they are all are for orbital_data.data_arc_in_days.We could just remove these or look into replacing the missing value with the mean of orbital_data.data_arc_in_days. However, as the data arc in days does not seem to be of much relevance, they are going to be removed for now.

```
#Code to omit rows with NA

# nrow(MyNASAdataset)                      # check current no of rows

MyNASAdataset <- na.omit(MyNASAdataset)  # remove the NAs

# nrow(MyNASAdataset)                      # rows numbers after deletion

# check if any incomplete rows are left

cat(" The number of incomplete rows is: ",sum(!complete.cases(MyNASAdataset)))
```

```
##  The number of incomplete rows is:  0
```

Check was conducted for the number of rows before and after removals, to confirm. The total number of incomplete rows after removals are verified as well.

The final dimensions are shown in Table 2.

Table 2: Dimensions of NASA Dataset after Data preparation

Dimension	Number
Instances	12516
Attributes	18

As the names of the attributes are too long, they are renamed at this stage for ease of use in the code and for displaying in plots.

```
#rename the attributes

MyNASAdataset<- dplyr::rename(MyNASAdataset,c(
    ABS_M = absolute_magnitude_h,
    PHA = is_potentially_hazardous_asteroid,
    ED_MIN = estimated_diameter.kilometers.estimated_diameter_min,
    ED_MAX = estimated_diameter.kilometers.estimated_diameter_max,
    MOI = orbital_data.minimum_orbit_intersection ,
    JTI = orbital_data.jupiter_tisserand_invariant,
    SMA = orbital_data.semi_major_axis,
    INC =  orbital_data.inclination,
    ANL = orbital_data.ascending_node_longitude,
    OR_P = orbital_data.orbital_period ,
    PER_D = orbital_data.perihelion_distance ,
    PER_A = orbital_data.perihelion_argument,
    AP_D = orbital_data.aphelion_distance ,
    PER_T = orbital_data.perihelion_time,
    M_ANO = orbital_data.mean_anomaly,
    M_MO = orbital_data.mean_motion,
    RV =  relative_velocity.kilometers_per_second ,
    MIS =  miss_distance.kilometers
```

```
  ))
```

2.3.4 Check the correlation matrix and check for further removals

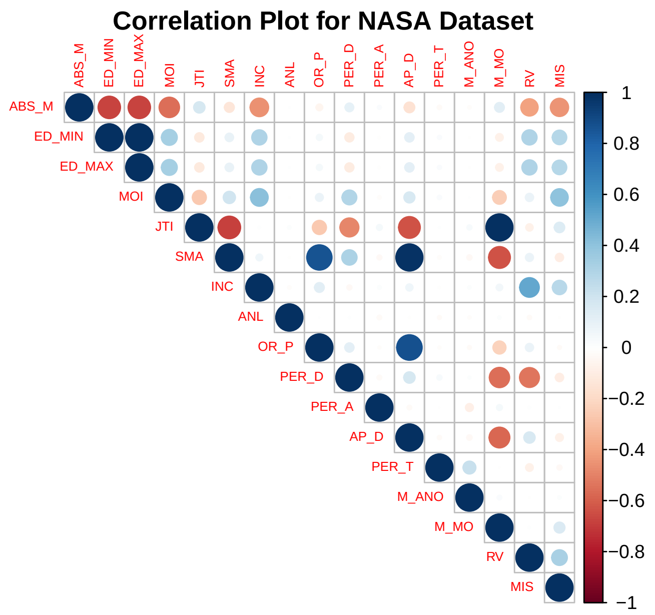


Figure 2: The Correlation chart for the Dataset

From Figure 2, there is very strong correlation shown between:

- EST_KM_MIN and EST_KM_MAX (ED_MIN) and (ED_MAX)
- OD_SMA and OD_AP_DIST (SMA) and (AP_D)
- OD_JTI and OD_M_MOT (JTI) and (M_MO)

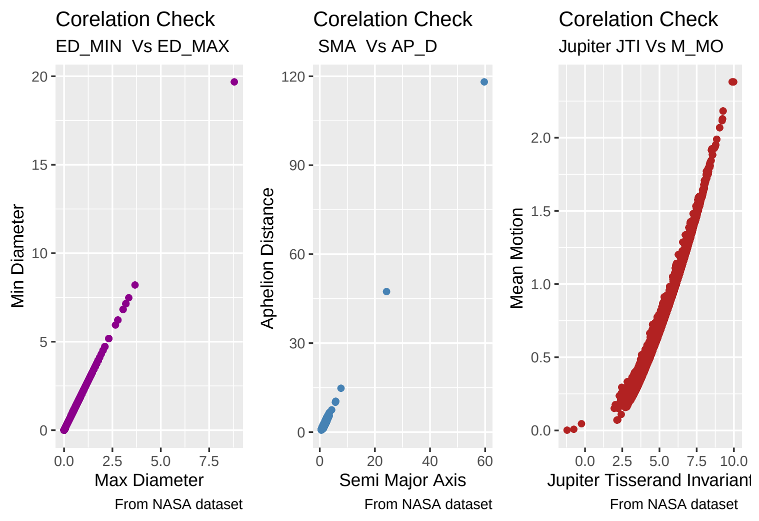


Figure 3: Additional check for highly correlated variables

From the plots in Figure 3 and the confidence intervals in the cor-test(code not shown),it can be seen that all three variable sets are highly correlated. Hence,estimated minimum,Aphelion Distance and mean motion is removed from the Dataset.

```
MyNASAdataset$ED_MIN <- NULL
MyNASAdataset$AP_D<- NULL
MyNASAdataset$M_MO<- NULL
```

2.3.5 Check the distribution of Class

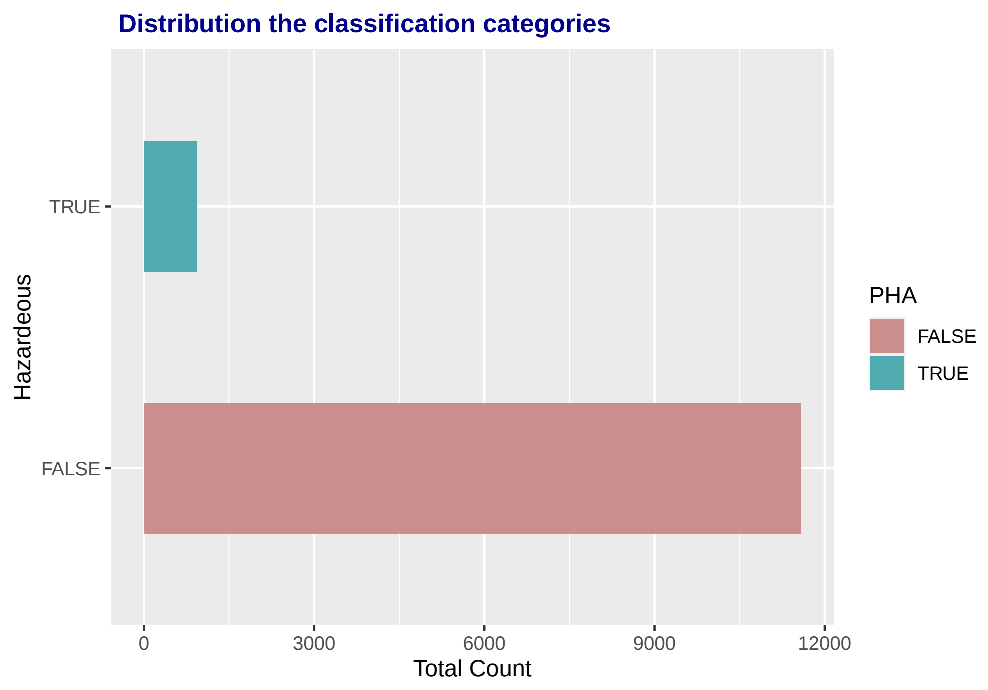


Figure 4: The Distribution of Class in the Dataset

Table 3: Distribution of Class in numbers

Var1	Freq
FALSE	11582
TRUE	934

As seen from Figure 4 and Table 3 , there are only 7.5% of True instances and 92.5% of False instances. The Dataset is heavily imbalanced.Balancing the data needs to be done after data exploration.

2.4 Exploratory Analysis

From this section onward, the PHA is renamed to Class.The attributes are checked with respect to the Class.The final list of Attributes are shown in Table 4.

```
# Rename PHA to Class and move it the last column

MyNASAdataset[16] <- MyNASAdataset$PHA   #copy PHA
colnames(MyNASAdataset)[16] <- "Class"   #rename column
MyNASAdataset <- MyNASAdataset[ ,-c(2)]  # PHA is in column 2
```

Table 4: The final list of attributes

Attributes
ABS_M
ED_MAX
MOI
JTI
SMA
INC
ANL
OR_P
PER_D
PER_A
PER_T
M_ANO
RV
MIS
Class

2.4.1 Check the the atrributes with respect to the Class

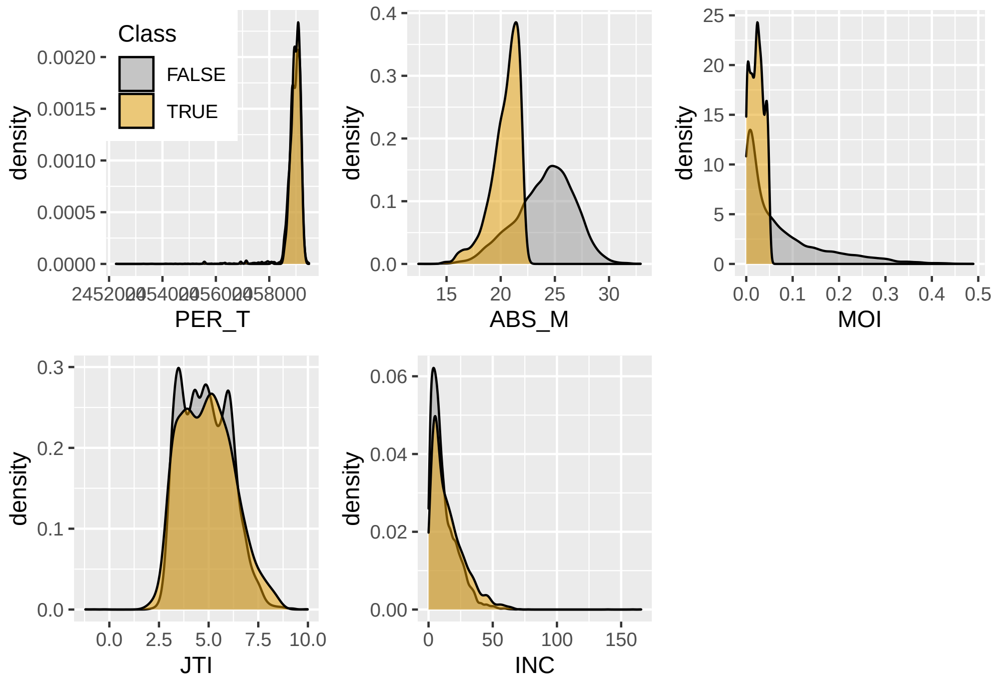


Figure 5: The Density plots w.r.t Class

Density plots are used to get a idea of shape of distribution for each variable. They are not limited by the number of bins like histograms and are plotted here by the 'Class'. From the density plots(Figures 5-7),only two attributes with a higher density for 'True' (yellow) is ABS_M and MOI. A distinction can be seen in ABS_M around 22.5 and for MOI around 0.05. RV,ED_MAX and PER_D show a degree of seperation as well.

Boxplots will be done for ABS_M , MOI and the attributes which were difficult to see clearly with the above plots(ED_MAX,OR_P).

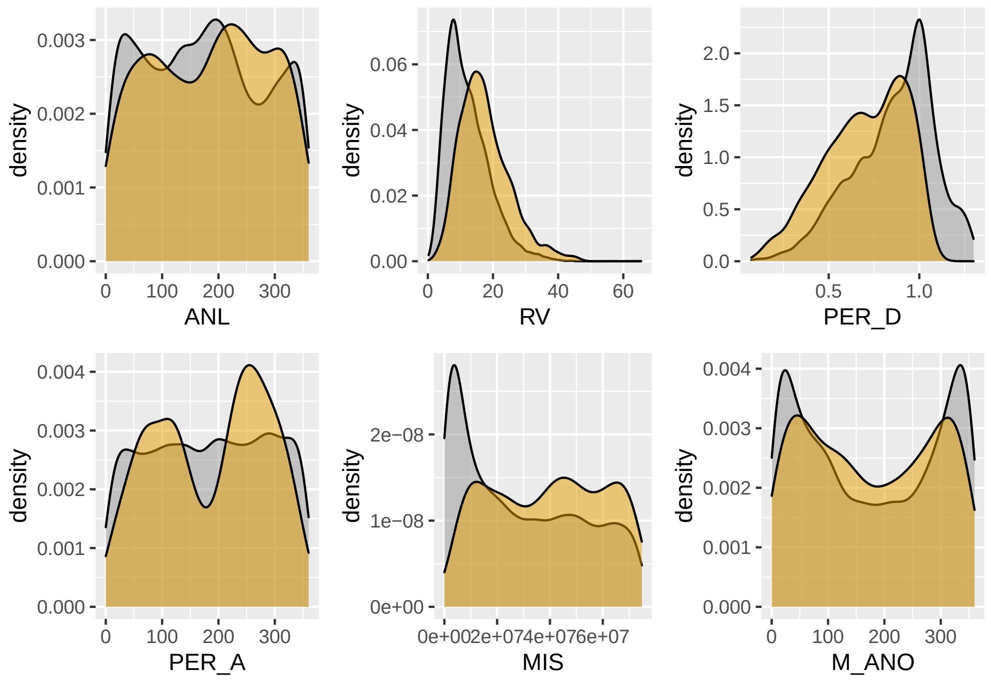


Figure 6: The Density plots w.r.t Class

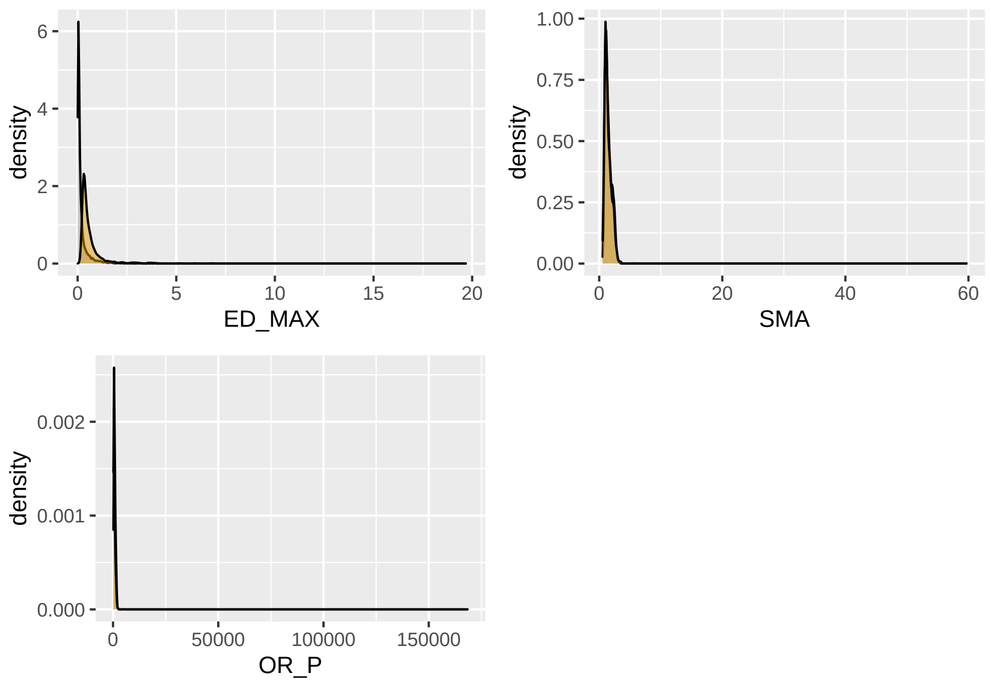
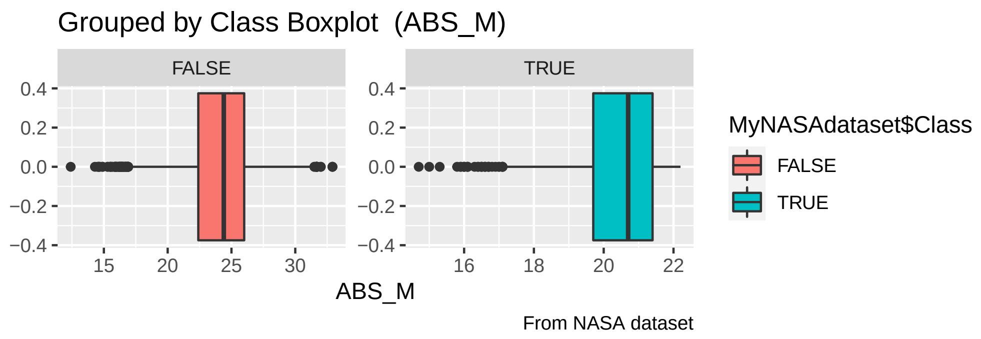


Figure 7: The Density plots w.r.t Class

2.4.2 Boxplots to check for outliers



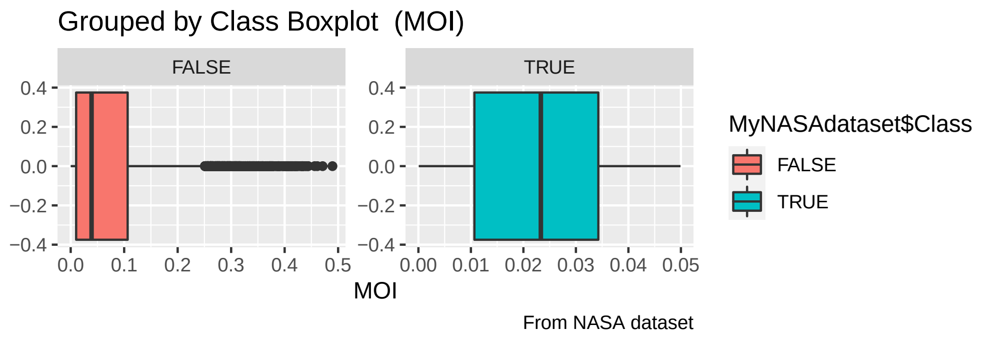


Figure 8: The Box plots for further attribute analysis

Figure 8 and 9 show the boxplots grouped by Class. ABS_M has outliers on both Class, with the True Class skewed. A distinction can be seen in the range clearly here. MOI has no outliers in its TRUE class, with the False class skewed and with outliers. Again , a clear distinction can be seen in the range between the classes.

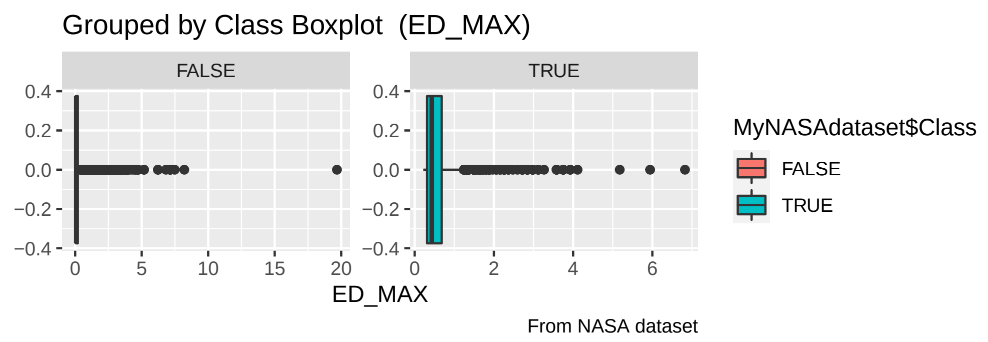
For ED_MAX, both classes are skewed and have multiple outliers, with overlapping ranges. For OR_P, the range for the FALSE class is still difficult to see due to the outliers.The true class has outliers as well and but has a lesser effect on range.

The outliers seen could be noise, labeling error or the most interesting sample.As this cannot be determined, they are not going to be modified here.

2.5 Split into train and test

The Dataset is going to be split into Train (80%) and Test(20%).Stratified split was used so that the train and test set has the same ratio of TRUE and FALSE variables as the full Dataset.This can be seen from Figure 10 and Table 5.

```r
seed <- 123

# Split data into partitions, stratified by variable Class
set.seed(seed)
inds <- partition(MyNASAdataset$Class, p = c(train = 0.8,test = 0.2))
#str(inds) # use for debugging the code

NASA_train <- MyNASAdataset[inds$train, ]
NASA_test  <- MyNASAdataset[inds$test, ]

percentage_training <- round(prop.table(table(NASA_train$Class)),4)
percentage_test     <- round(prop.table(table(NASA_test$Class)),4)
```



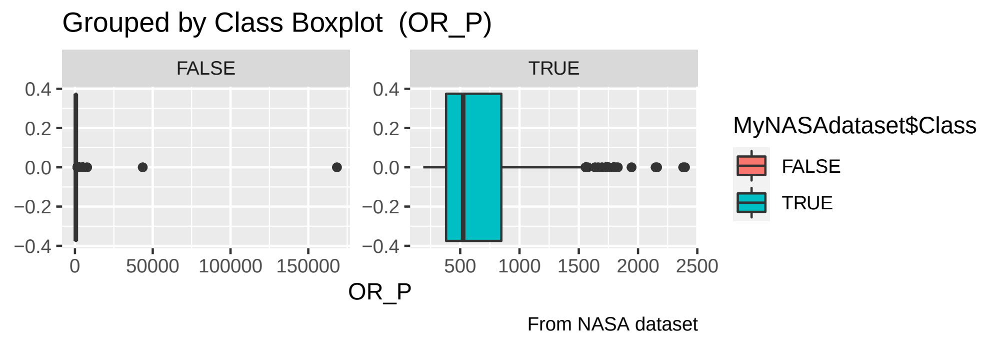
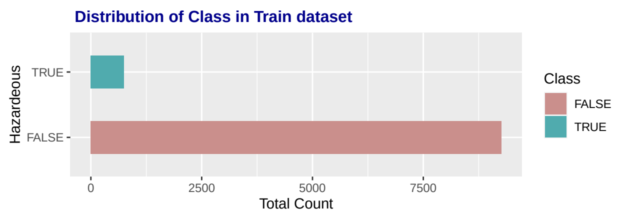


Figure 9: The Box plots for further attribute analysis

```
number_training      <- table(NASA_train$Class)
number_test          <- table(NASA_test$Class)
```



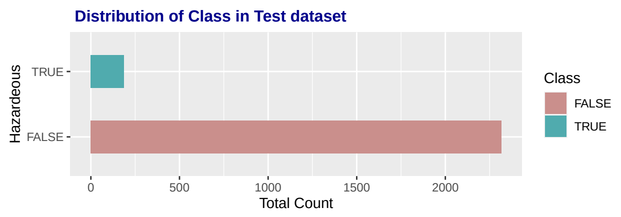


Figure 10: Distribution of Class

```
Distribution <- cbind(percentage_training,number_training,
                      percentage_test,number_test)
colnames(Distribution)<- c( "\t    Training Set Ratio"," Training Set Numbers",
                            "Test Set Ratio ", "  Test Set Numbers  ")
kable(Distribution,caption = " The current distribution of train and test set ",
      booktabs = TRUE,valign = 't')
```

Table 5: The current distribution of train and test set

	Training Set Ratio	Training Set Numbers	Test Set Ratio	Test Set Numbers
FALSE	0.9254	9265	0.9253	2317
TRUE	0.0746	747	0.0747	187

Next step is to improve the balance of the training set, while not disturbing the test set. To try out the difference processing can make for this Dataset, four repositories are going to be created as shown in Figure 11, from the training set. Appropriate processing is done for the test set as well.Functions have been created for the each of the processing needed so that it can be called as needed without repetition of code.

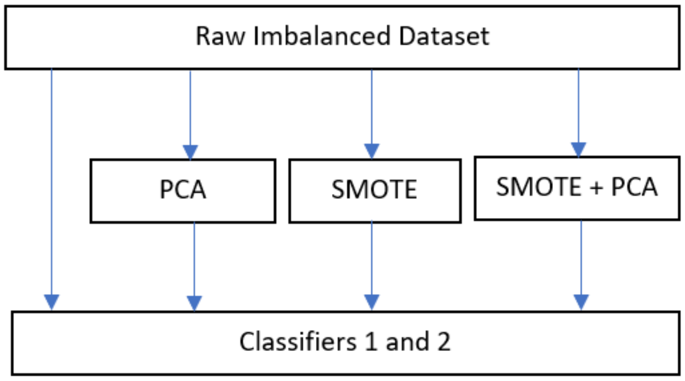


Figure 11: The Repositories created

2.6 Functions for the creating repositories(repos)

The functions created are for :

- Principal component Analysis (PCA)
- Synthetic Minority Over-sampling Technique (SMOTE)
- Scree plot for the PCA

PCA is used to transform a large set of variables into a smaller set of principal components,grouped according to their variance.PCA can also be used for dimentionality reduction by dropping some of the components.

SMOTE(Chawla etal.,2002) creates "synthetic" examples from the minority class by introducing new samples, near the minority class nearest neighbors.

Screeplot is one of the options used to determine the number of principal components selected after PCA.

The code for these are shown below.

2.6.1 PCA function

```
# function for Principal Component Analysis (PCA)

# PCA Works best with numerical data - as it is an unsupervised learning
#technique, hence Class variable must be removed.
```

```
pca <- function(data) {

  class <- data$Class
  data_withoutclass <- data[,-c(15)]# remove class

  #PCA

  pca.data <- prcomp(data_withoutclass,center = TRUE , scale = TRUE)


  #plot first two PCA

  PCAplot1 <-autoplot(pca.data , data=data, colour="Class",
                      alpha = 0.1,loadings = TRUE, loadings.colour = 'blue',
                      loadings.label.colour='black', loadings.label = TRUE,
                      loadings.label.size = 4,
                      loadings.label.repel=TRUE)+
                      stat_ellipse(type="norm",level=0.68,aes(color=data$Class))

  print(PCAplot1)


  return(pca.data)

}
```

2.6.2 SMOTE function

The function for SMOTE is shown below. The code for the calculation for the desired percentage of SMOTE was adapted from (Datacamp,2021).

```
# function for SMOTE
smote_custom <- function(data) {

  #Setting no of cases and the desired percentage
 L0 <- 9265
 L1 <- 747
 P0 <- 0.85 # choose ratio of generated data,0.5 will give almost equal

  # calculate the value for the dup_size parameter of SMOTE

  dup <- ((1 - P0) /P0) * (L0 / L1) - 1

  # create synthetic cases with SMOTE

  smote_output <- SMOTE(X = data[ , -c(15)], target = data$Class,
                        K = 5, dup_size = dup)

  data_smote <- smote_output$data
  data_smote$class <- as.factor(data_smote$class)

  colnames(data_smote)[15] <- "Class"

  #create table with the training set proportion and values
  train_proportion <- prop.table(table(data_smote $Class))
  train_values <-table(data_smote $Class)

  Dist2 <- cbind(train_proportion,train_values)
  colnames(Dist2)<- c( "\t\tTraining set Proportion","\tTraining set Values ")

  #plot with the training and test set
  gg<- ggplot(data_smote ,aes(x= Class,fill=Class)) + geom_bar(width = 0.5) +
  xlab("Hazardeous") + ylab("Total Count")+
  ggtitle(" Distribution of Class in the training set with SMOTE ") +
  theme(plot.title=element_text(color="dark blue",size=12, face="bold.italic"))+
  coord_flip()+ scale_fill_hue(c = 40)
  print(gg)
  return(data_smote)
}
```

2.6.3 Scree plot function

```
# function for screeplot test - created from factoextra package

screeplot <- function(pca.data) {

 scree1 <- fviz_screeplot(pca.data, ncp = 6,
  barfill ="cadetblue",linecolor = "#FC4E07")+ ggtitle("With 6 components")+
  theme(plot.title = element_text(color="dark blue",size=12,face="bold.italic"))

 scree2 <- fviz_screeplot(pca.data, ncp = 16,
  barfill = "cadetblue",linecolor= "#FC4E07")+ ggtitle("With all components")+
  theme(plot.title =element_text(color="dark blue",size=12,face="bold.italic"))

 # arrange plots in a grid
 print(grid.arrange(scree1,scree2, nrow = 1, ncol = 2,
              top=textGrob("Screeplot Test ",gp=gpar(fontsize=20,font=1))))
}
```

2.7 Assembling all repos

The ‘Original Repo’ is taken directly from the Dataset.

```
#create a list for repos
repos <- list()
original.repo <- list("Original Repo",NASA_train,NASA_test)
```

Function PCA is called for creating the Original with PCA repo.

```
# call pca
org_pca_train <- pca(NASA_train)
```

```
train_data_org_pca <- data.frame(org_pca_train$x,NASA_train$Class)
colnames(train_data_org_pca)[15] <- "Class"

### Test set - convert to data frame , with class added on

NASA_test1 <- NASA_test[-15]
```

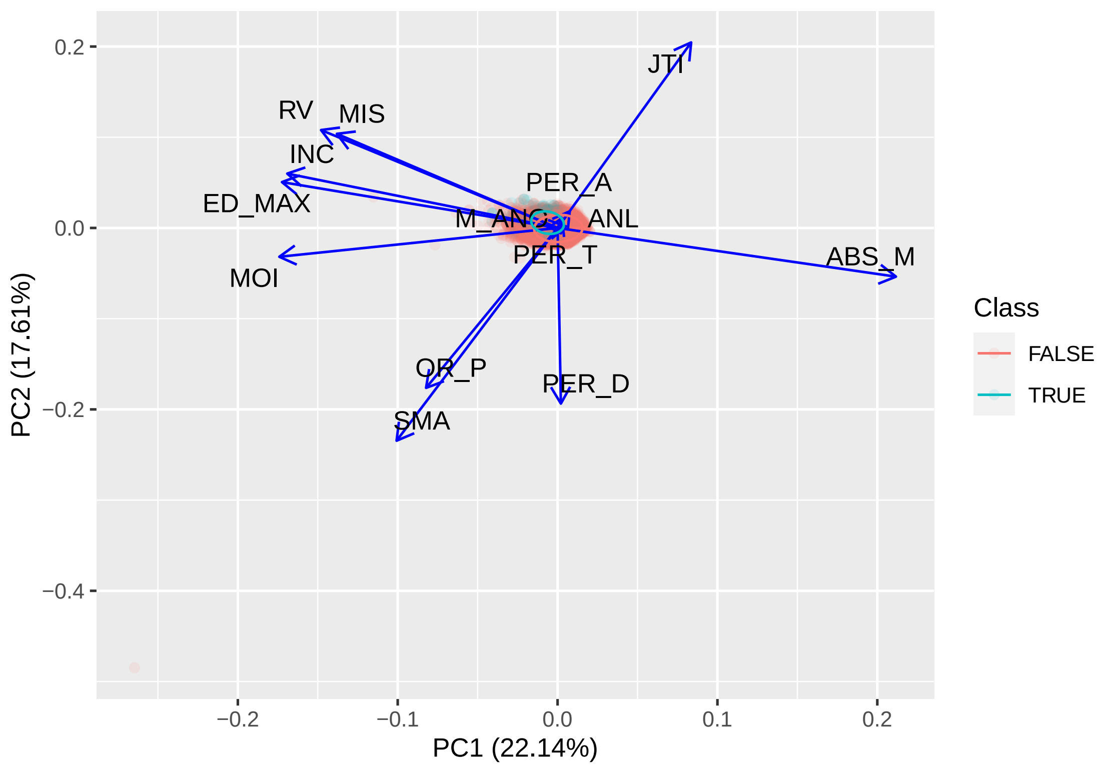


Figure 12: Relevant Figures for Original + PCA

```
test_data_org_pca <- predict(org_pca_train, newdata = NASA_test1)
test_data_org_pca1 <- data.frame(test_data_org_pca,NASA_test$Class)
colnames(test_data_org_pca1)[15] <- "Class"
original_pca.repo<-list("Original with PCA",train_data_org_pca,test_data_org_pca1)

#call screeplot
screeplot(org_pca_train)
```

PCA has been applied and Figure 12 shows the first two components of the PCA w.r.t the original variables. This plot helps identify the variables which have the most important influence on the components. ABS_M has a large positive effect on PC1,while ED_MAX,INC,RV,MIS,MOI have large negative effect on PC1.For PC2, PER_D has negative effect.Due to the large number of overlapping samples, the clustering is difficult to see, but zooming the image helps identify the clusters with the help of the ellipses. There is an outlier in the lower left hand corner in the FALSE class that could be investigated further.

The 14 PCA dimensions can be viewed in the Figure 13,in terms of their percentage of variance. To determine the number of PCAs to be retained:

- Use the screeplot above and choose the elbow of the curve as the cutoff for dimensions, in this case upto PC3.
- Select the upto PCA that gives a cumulative variance of around a cut off (99%),upto PC 12 here.
- check eigenvalue >1 from Kaiser-Guttman Rule, using paran(), which would give upto PCA 5.

However, as the aim here is to compare the best possible outcome for each repository and classifier combinations, all the PCAs have been retained so that dimentionality reduction does not potentially lead to a reduction in the metrics values been calculated.

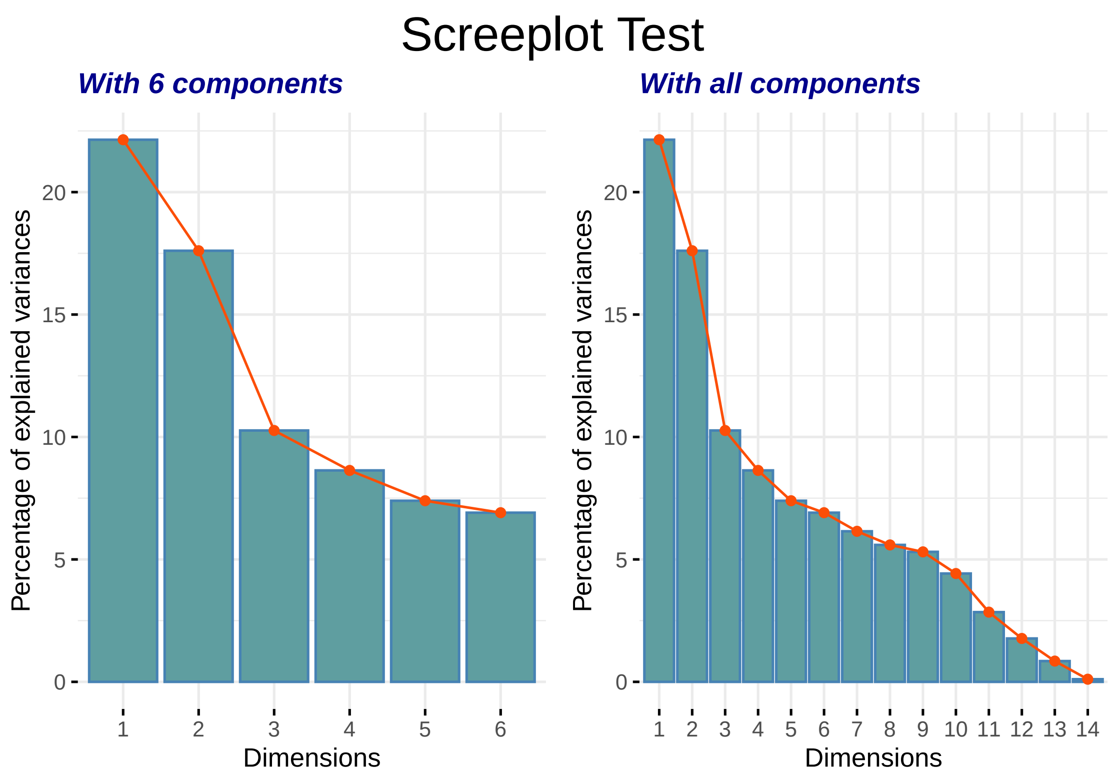


Figure 13: Relevant Figures for Original + PCA

```
#call smote
org_smote_train  <- smote_custom(NASA_train)
```

```
original_smote.repo <- list("Original with SMOTE",org_smote_train,NASA_test)
```

SMOTE function is created to increase the number of minority class by adding synthetically generated data. This can be used to balance Datasets. Here it has only been used to increase the minority class by 50 % as shown in the Figure 14.

```
#call smote + pca

org_smote_train  <- smote_custom(NASA_train)
smote_pca_train <- pca(org_smote_train)

train_data_smote_pca <- data.frame(smote_pca_train$x,org_smote_train$Class)
colnames(train_data_smote_pca)[15] <- "Class"


# Processing for Test set

NASA_test1 <- NASA_test[-15]
test_data_org_pca <- predict(smote_pca_train, newdata = NASA_test1)
test_data_org_pca1 <- data.frame(test_data_org_pca,NASA_test$Class)
colnames(test_data_org_pca1)[15] <- "Class"
smote_pca.repo <-list("SMOTE with PCA",
                        train_data_smote_pca,test_data_org_pca1)
```

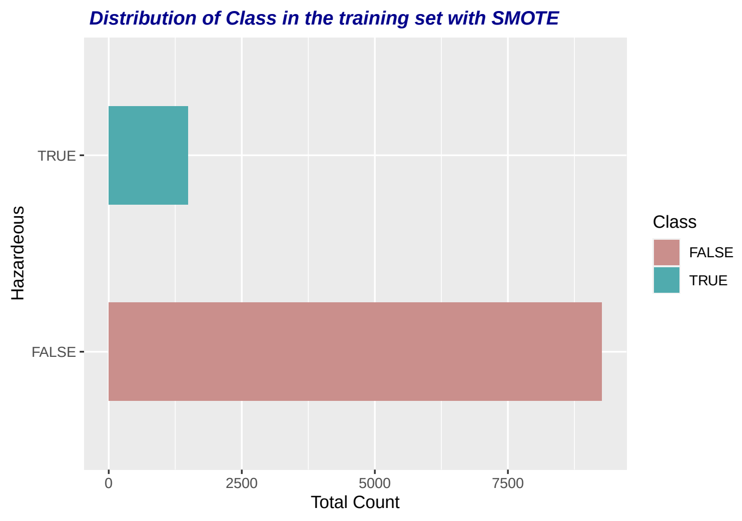


Figure 14: Relevant Figures for Original + SMOTE

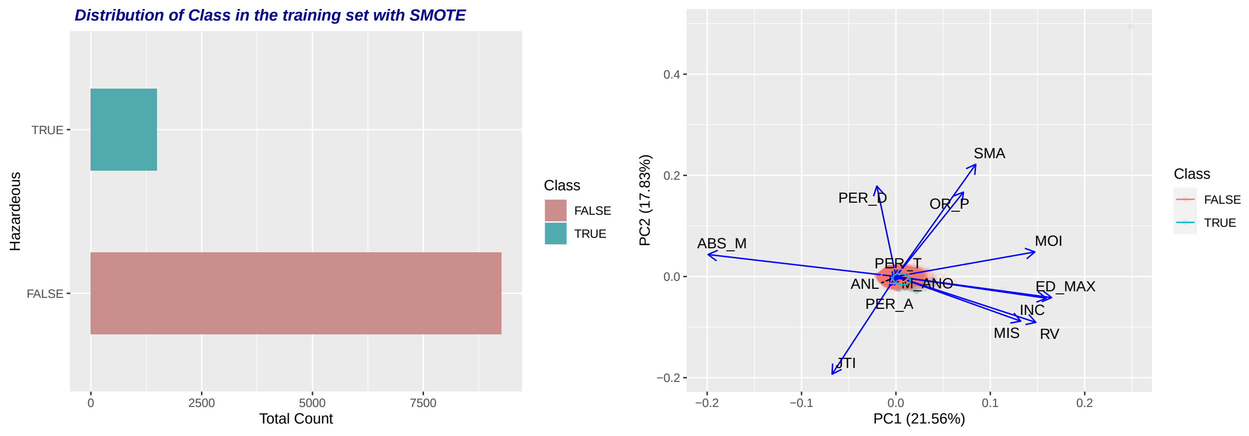


Figure 15: Relevant Figures for SMOTE + PCA

As the plots for SMOTE + PCA are (Figure 15) are similar to the ones for SMOTE and PCA separately they are not discussed again.

2.7.1 Assembling all repos as a list

```r
# all repos

repos <- list(original.repo,
              original_pca.repo,
              original_smote.repo,
              smote_pca.repo)

#bestplot <- list()
```

3 Supervised Learning experiment - Classification

The classifiers selected are:

- SVM(poly) with tuning for C, degree and scale
- RF with tuning for mtry

It can be see that Random Forest(RF) and Support Vector Machines (SVM) are one of the top performing classifiers from (Fernandez-Delgado etal.,2014).Hence they are chosen as the two classifiers to be compared here.

Both the classifiers will be tuned and the same seed is set for them to keep the experiment unbiased.The same cross validation parameters are used as well.

3.1 SVM classifier

The code for the function for SVM is shown below. It takes the repo as input and returns the data evaluated on the test set. The metrics will be derived from this in the evaluation stage.Tuning is done for Kappa. More information about the evaluation setup is shown in Figure 18 in the Evaluation section.

```r
# function for svm poly  model

svmp_tuning <- function(repo) {

  repo_name  <- repo[[1]]
  train_data <- repo[[2]]
  test_data  <- repo[[3]]

  set.seed(seed)

  #setting ctrl as 5 - fold Cross validation

  ctrl <- trainControl(method = "cv",number = 5,
               verboseIter=FALSE ) # run information is not outputted

  Cs <- c(0.5,1,2)
  degrees <- c(2,3)
  scales <- c(1,2)
  tunegrid <- expand.grid(C=Cs, degree = degrees, scale = scales)

  set.seed(seed)
  svmpolymod <- train(Class ~ ., data = train_data, method = "svmPoly",
             tuneLength = 10,     #  granularity of the tuning parameter grid
             metric = "Kappa",    # Accuracy is the metric chosen
             tuneGrid= tunegrid,
             preProcess = c("center","scale"),
             trControl = ctrl)

            # Variable Importance - commented out here
            # print(plot(varImp(svmpolymod, scale = FALSE,
            #                 main=(paste("Best plot for for SVM : ",
            # repo_name))))) #plot variable importance


  # plot for tuning

  print(plot(svmpolymod,main=(paste("Tuning for SVM : ",repo_name)),
        xlab="Cost",
        ylab="Kappa(Cross-Validation)",
        font.main=12, font.lab=8))


  # test set

  test_data$pred <- predict(svmpolymod, test_data[-15])

  return (test_data)

}
```

This section loops through the list of repositories created earlier and applies SVM function to each of them.The results are saved in test_results_svm.

```
test_results_svm <- list()
i<- 1;
for (repo in repos) {
  #call SVM poly  classifier
  test_results_svm[[i]] <- svmp_tuning(repo)
  i <- i+1
}
```

Figure 16 shows the plots for tuning of the SVM classifier. Variation of the Kappa for the variation in Cost ,degree and scale can be seen. For Original with PCA and Original with SMOTE, the plot for scale 2 is on an upward trend and should ideally be explored further.

3.2 RF Classifier

The code for the function for RF is shown below. It takes the repo as input and returns the data evaluated on the test set. The metrics will be derived from this in the evaluation stage.Tuning is done for Kappa.More information about the evaluation setup is shown in Figure 18 in the Evaluation section.

```
# function for tuning mtry

rf_tuning_mtry <- function(repo) {

  repo_name  <- repo[[1]]
```

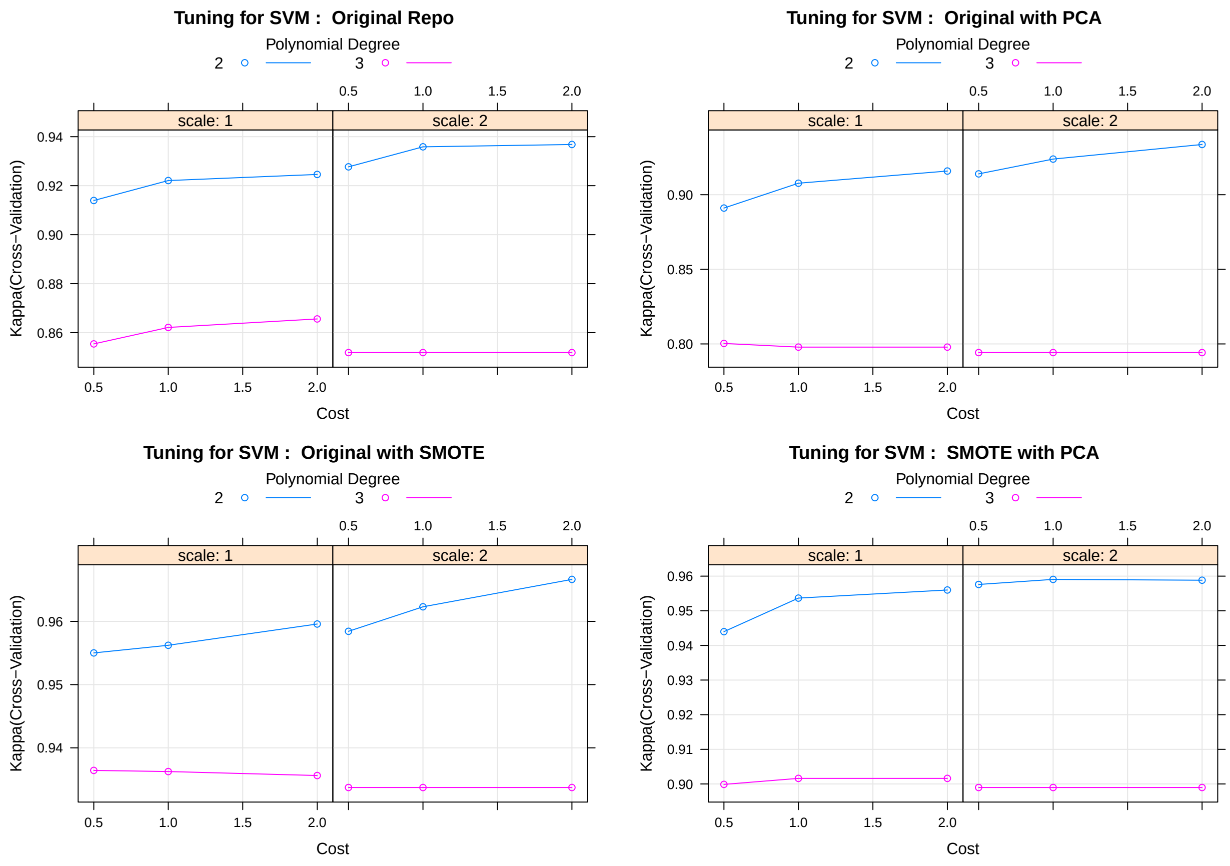


Figure 16: Plots for Tuning(SVM)

```
  train_data <- repo[[2]]
  test_data  <- repo[[3]]


  set.seed(seed)

  #setting ctrl as 5-fold  Cross validation

  ctrl <- trainControl(method = "cv",number = 5,
               verboseIter=FALSE )      # run information is not outputted

  # tuning section partially adapted from (Brownlee,2016)

  tunegrid <- expand.grid(.mtry=c(1:5))

  set.seed(seed)

  rf_mtry <- train(Class ~ ., data = train_data, method="rf",
                      tuneGrid=tunegrid,tuneLength = 10,  metric = "Kappa",
                      trControl = ctrl)

  # plot for mtry tuning decision

  rfplot<- ggplot(rf_mtry)+geom_line(color="#E69F00",size =1)+
  xlab("mtry Values") + ylab("Accuracy")+
  ggtitle(paste("Tuning for RF : ",repo_name)) +
  theme(plot.title=element_text(color="dark blue",size=12, face="bold.italic"))
  print(rfplot)


  # Variable Importance - commented out here
    # print(plot(varImp(rf_mtry, scale = FALSE,
    #    main=(paste("Best plot for for RF : ",repo_name)))))

  # predict on test set

  test_data$pred <- predict(rf_mtry, test_data[-15])
  return (test_data)

}
```

This section loops through the list of repositories created earlier and applies SVM function to each of them.The results are saved in test_results_tuned.

```
test_results_tuned <- list()
i<- 1;
for (repo in repos) {
  #call Random Forest classifier
  test_results_tuned[[i]] <- rf_tuning_mtry(repo)
  i <- i+1
}
```

In Figure 17, the tuning of Rf for mtry is visualised, best mtry value will be chosen as the highest point in the plot for each repo. ‘SMOTE with PCA’ is on an upward trend and ideally the range of mtry should be extended to check it the Kappa improves beyond the best value (currently at 3).

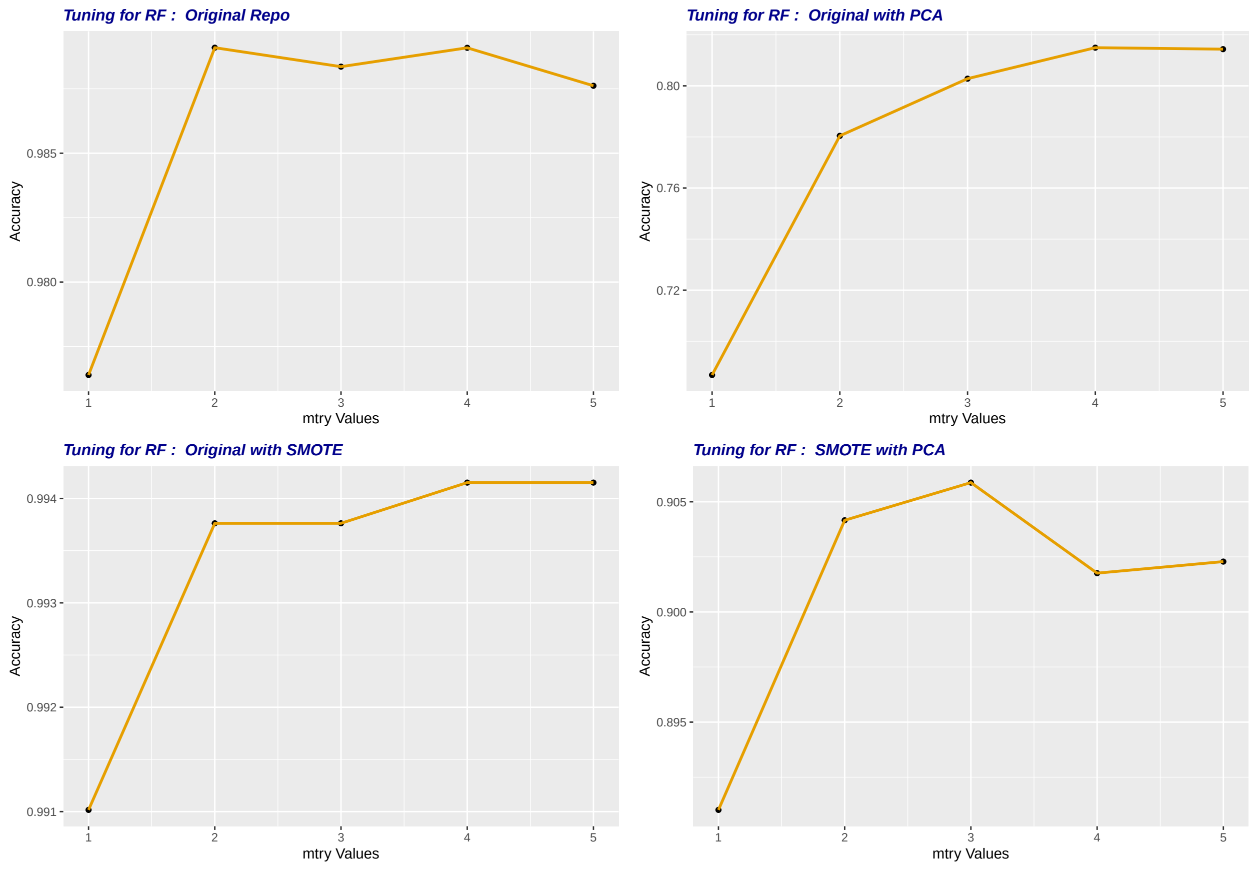


Figure 17: Tuning(RF)

4 Evaluation

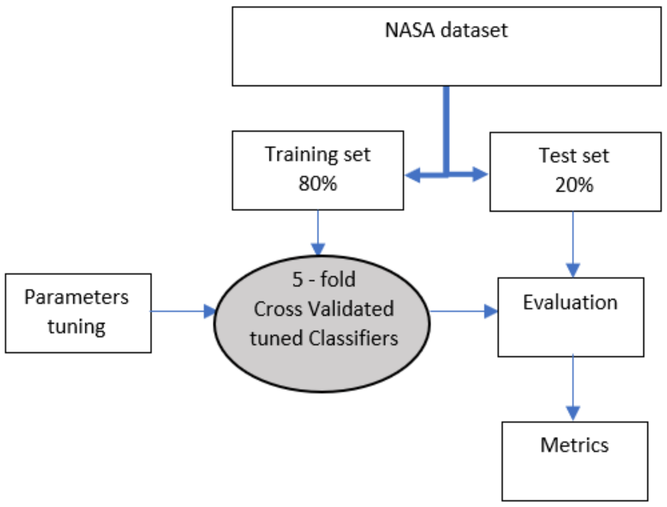


Figure 18: The Evaluation Setup

Figure 18 shows the Evaluation setup for this coursework.

4.1 Metrics

The comparison is done using :

- Confusion Matrix
- Precision
- Recall
- AUC
- Accuracy

This function displays the confusion matrix in a graphical format and calculates the values for the other metrics mentioned above.The code for the confusion matrix part of the function was adapted from (Stackflow, 2021).

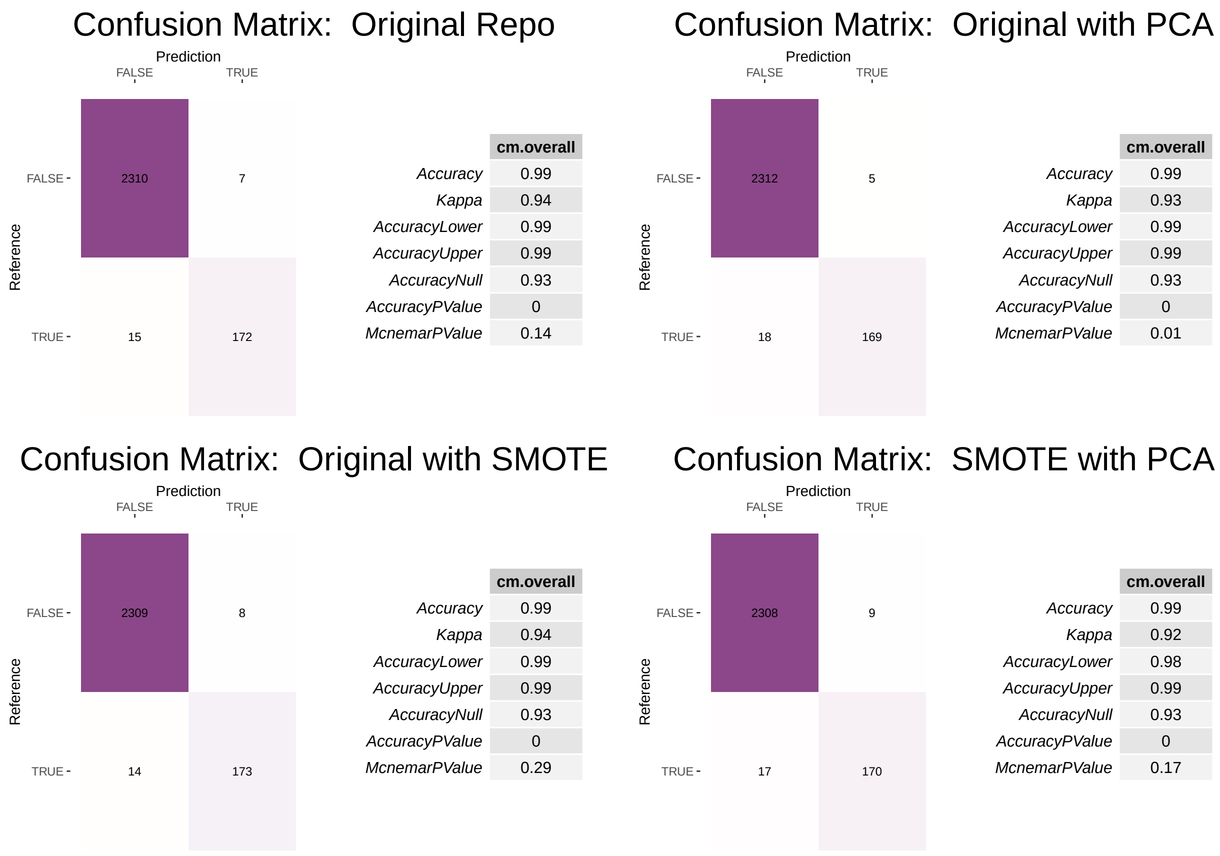
```
#function to calculate performance metrics-Precision,Recall,AUC and Accuracy

get_perf_metrics <- function(test_data,name) {

  #------ code for plotting confusion matrix from ref matrix ------------

  cm<- caret:: confusionMatrix(test_data$pred,test_data$Class)
  cm_d <- as.data.frame(cm$table)# confusion matrix values as data.frame
  cm_st <-data.frame(cm$overall) # confusion matrix statistics as data.frame
  cm_st$cm.overall <- round(cm_st$cm.overall,2) # round the values
  cm_d$diag <- cm_d$Prediction == cm_d$Reference # Get the Diagonal
  cm_d$ndiag <- cm_d$Prediction != cm_d$Reference # Off Diagonal
  cm_d[cm_d == 0] <- NA # Replace 0 with NA for white tiles
  cm_d$Reference <- reverse.levels(cm_d$Reference)# diagonal starts at top left
  cm_d$ref_freq <- cm_d$Freq * ifelse(is.na(cm_d$diag),-1,1)
  plt1 <-  ggplot(data = cm_d,aes(x = Prediction,y =Reference, fill = Freq))+
    scale_x_discrete(position = "top") +
    geom_tile( data = cm_d,aes(fill = ref_freq)) +
    scale_fill_gradient2(guide = FALSE ,low="red3",high="orchid4",
                        midpoint = 0,na.value = 'white') +
                        geom_text(aes(label = Freq), color = 'black',size=3)+
                        theme_bw() +
    theme(panel.grid.major = element_blank(),panel.grid.minor=element_blank(),
          legend.position = "none",
          panel.border = element_blank(),
          plot.background = element_blank(),
          axis.line = element_blank(),
          )

  plt2 <-  tableGrob(cm_st)
  print(grid.arrange(plt1, plt2, nrow = 1, ncol = 2,
               top=textGrob((paste("Confusion Matrix: ",name)),gp=gpar(fontsize=25,font=1))))

  #----------- End of confusion Matrix Plot section ----------------------

  precision <-  round(precision(as.logical(test_data$Class),
                         as.logical(test_data$pred)),digits=4)
  recall <- round(recall(as.logical(test_data$Class),
                         as.logical(test_data$pred)),digits=4)
  AUC <- round(auc(as.logical(test_data$Class),
                         as.logical(test_data$pred)),digits=4)
  accuracy <- round(accuracy(test_data$Class,test_data$pred),digits=4)

  perf_metrics <- cbind(precision,recall,AUC,accuracy)

  return(perf_metrics)
}
```

This section gets the metrics for all the repos from tuned SVM classifier and displays the confusion matrix from the earlier function.

```
all_perf_metrics_svm <- data.frame()

for (i in 1:length(test_results_svm)) {


  perf_metrics_t <- get_perf_metrics(test_results_svm[[i]],repos[[i]][[1]])

  allt<- cbind("SVM Poly",repos[[i]][[1]],perf_metrics_t)
  all_perf_metrics_svm<- rbind(all_perf_metrics_svm,allt)

}
names(all_perf_metrics_svm) <- c("Classifier","Repo", "Precision", "Recall",
                              "AUC", "Accuracy")
```

The table below shows the other metrics for the SVM.

```
knitr::kable(all_perf_metrics_svm,
      caption = " All Performance Metrics for SVM Evaluation")
```



Original Repo

	cm.overall
Accuracy	0.99
Kappa	0.94
AccuracyLower	0.99
AccuracyUpper	0.99
AccuracyNull	0.93
AccuracyPValue	0
McnemarPValue	0.14

Original with PCA

	cm.overall
Accuracy	0.99
Kappa	0.93
AccuracyLower	0.99
AccuracyUpper	0.99
AccuracyNull	0.93
AccuracyPValue	0
McnemarPValue	0.01

Original with SMOTE

	cm.overall
Accuracy	0.99
Kappa	0.94
AccuracyLower	0.99
AccuracyUpper	0.99
AccuracyNull	0.93
AccuracyPValue	0
McnemarPValue	0.29

SMOTE with PCA

	cm.overall
Accuracy	0.99
Kappa	0.92
AccuracyLower	0.98
AccuracyUpper	0.99
AccuracyNull	0.93
AccuracyPValue	0
McnemarPValue	0.17

Figure 19: Metrics(SVM)

Table 6: All Performance Metrics for SVM Evaluation

Classifier	Repo	Precision	Recall	AUC	Accuracy
SVM Poly	Original Repo	0.9609	0.9198	0.9584	0.9912
SVM Poly	Original with PCA	0.9713	0.9037	0.9508	0.9908
SVM Poly	Original with SMOTE	0.9558	0.9251	0.9608	0.9912
SVM Poly	SMOTE with PCA	0.9497	0.9091	0.9526	0.9896

Table 7: All Performance Metrics for RF Evaluation

Classifier	Repo	Precision	Recall	AUC	Accuracy
RF	Original Repo	1	0.9572	0.9786	0.9968
RF	Original with PCA	0.9091	0.6952	0.8448	0.972
RF	Original with SMOTE	1	0.9679	0.984	0.9976
RF	SMOTE with PCA	0.882	0.7594	0.8756	0.9744

In Table 6, It can be seen that all repositories show good accuracy for SVM Poly (all above 99%) even though the Precision ,Recall and AUC scores slightly lower.

Of the four repos with SVM, the 'Original Repo' and the 'Original Repo with SMOTE',show the best results as they have high precision(around 95%) and very high recall, with the very high AUC as well.This can be seen from the Confusion Matrix in Figure 19 as well with much lesser amount of wrongly classified instances in the 'Original Repo' and 'Original with SMOTE', compared to the rest.

As 'Original + SMOTE' has slightly better metrics overall, it will chosen as the Best for SVM and will be compared with the best output for RF.

This section below gets the metrics for all the repos from tuned RF classifier and displays the confusion matrix from the earlier get_perf_metric function.

```
all_perf_metrics_tuned <- data.frame()

for (i in 1:length(test_results_tuned)) {

  perf_metrics_t <- get_perf_metrics(test_results_tuned[[i]],repos[[i]][[1]])
  allt<- cbind("RF",repos[[i]][[1]],perf_metrics_t)
  all_perf_metrics_tuned<- rbind(all_perf_metrics_tuned,allt)

}

names(all_perf_metrics_tuned) <- c("Classifier","Repo", "Precision", "Recall",
                              "AUC", "Accuracy")
```

```
kable(all_perf_metrics_tuned,
      caption = " All Performance Metrics for RF Evaluation")
```

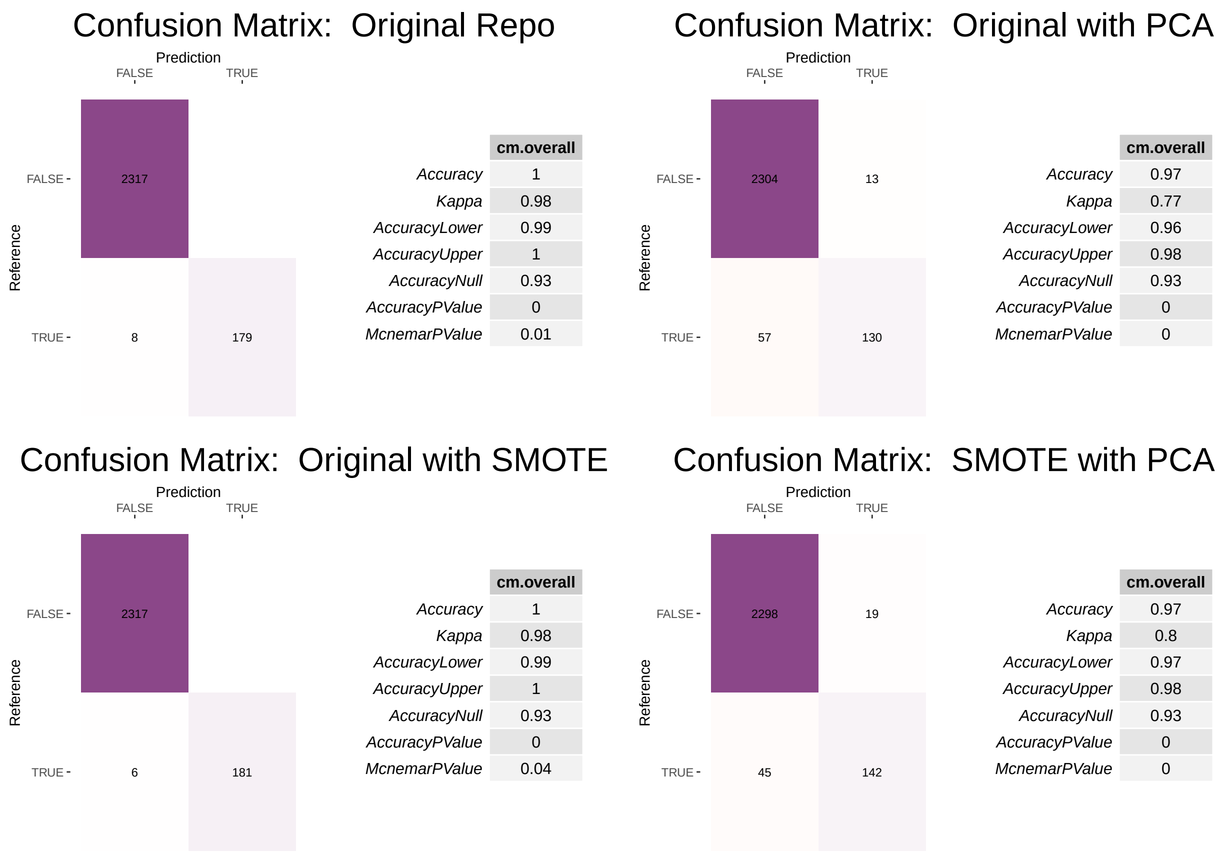
In Table 7, all repositories show high accuracy for RF (all above 97%),though the Precision , Recall and AUC scores differ. Both repositories with PCA have shown worse results for recall overall. Precision is slightly better for 'Original with PCA', but with worse recall.

Of the four repos with RF, the 'Original Repo' and the 'Original Repo with SMOTE',show the best results as they have perfect precision and very high recall, with the very high AUC as well.This can be seen from the Confusion Matrix in Figure 20 as well.SMOTE also has more possibility of improvement.

Due to this, 'Original + SMOTE' has been chosen as the best for RF and will be compared with the best output for SVM.

5 Conclusion

Accuracy is not the best metric for comparison in this imbalanced Dataset as it will provide a very high accuracy even if the minority class is rarely predicted correctly.For this Dataset,the recall is just as important as precision due to the nature of the classification outcome.



Original Repo:

	cm.overall
Accuracy	1
Kappa	0.98
AccuracyLower	0.99
AccuracyUpper	1
AccuracyNull	0.93
AccuracyPValue	0
McnemarPValue	0.01

Original with PCA:

	cm.overall
Accuracy	0.97
Kappa	0.77
AccuracyLower	0.96
AccuracyUpper	0.98
AccuracyNull	0.93
AccuracyPValue	0
McnemarPValue	0

Original with SMOTE:

	cm.overall
Accuracy	1
Kappa	0.98
AccuracyLower	0.99
AccuracyUpper	1
AccuracyNull	0.93
AccuracyPValue	0
McnemarPValue	0.04

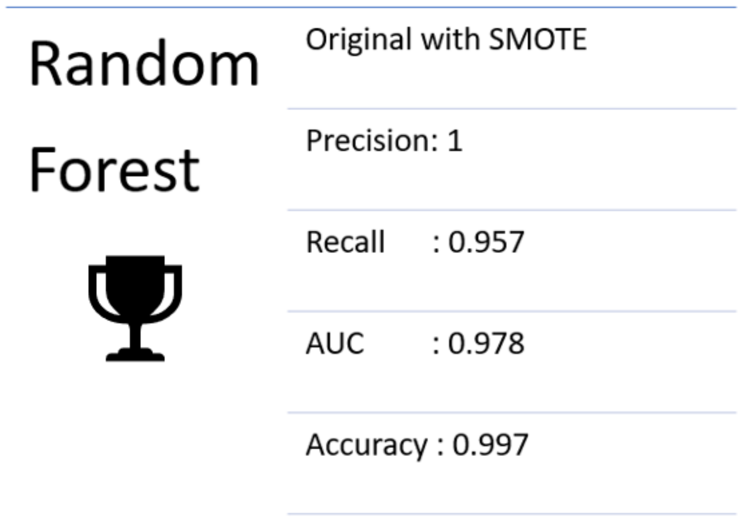
SMOTE with PCA:

	cm.overall
Accuracy	0.97
Kappa	0.8
AccuracyLower	0.97
AccuracyUpper	0.98
AccuracyNull	0.93
AccuracyPValue	0
McnemarPValue	0

Figure 20: Metrics(RF)

- Both RF and SVM Poly have shown good results on the test set for the ‘Original repo’ and ’Original with SMOTE’with high recall, precision , AUC and Accuracy.
- The precision and recall for repos with PCA for both SVM and RF perform slightly worse than repos without PCA. This could be due to the number of outliers seen during the exploration of Dataset. It should be noted that PCA was not used for dimentionality reduction in this coursework as all the PCAs were used.Hence these results could only be improved with further tuning of classifiers or exploring the effect of the outliers further.
- The only tuning done for Rf was through ‘mtry’ and the results could improve further with tuning of ‘ntree’ as well.
- Comparing the two best repos from each of the classifiers , it can be seen that ‘Original with SMOTE’ performs better when comparing all the metrics.

As mentioned above, recall is just as important as precision, hence the best classifier repo combination is: RF and ‘Original with SMOTE’ as highlighted in Figure 21.



Random Forest

Original with SMOTE
Precision: 1
Recall : 0.957
AUC : 0.978
Accuracy : 0.997

Figure 21: The Best Classifier and Repo combination

The Importance of variables can be seen in Figure 22. The two most importance variable identified are ED_MAX and ABS_M, which were also identified as possible important variables during the exploration stage.

Other ways to potentially improve the results further include:

- balancing the data further with SMOTE for minority class.
- using undersampling of majority class along with SMOTE for minority class.
- Applying class decomposition with SMOTE (CD-SMOTE).

Future work would include implementing the above while keeping the classifier as RF.

6 References

BROWNLEE,J.,2016.Tune Machine Learning Algorithms in R(random forest case study) .[online]. Available from: https://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/ [Accessed 19 Apr 2021].

CHAWLA,N.etal,2002.SMOTE: Synthetic Minority Over-sampling Technique.Journal of Artificial Intelligence Research 16 (2002) 321–357.Available from:https://arxiv.org/pdf/1106.1813.pdf%5BAccessed 22 Apr 2021].

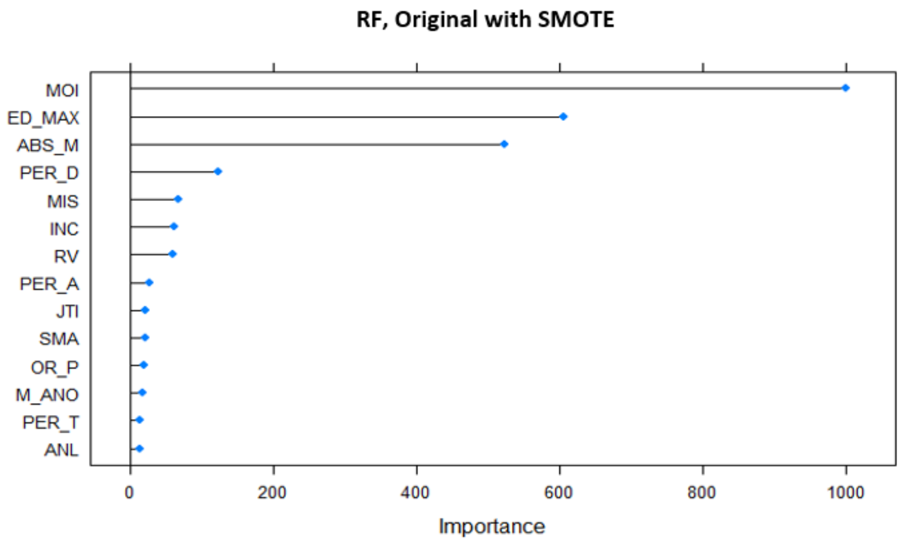


Figure 22: Variable Importance for best RF-repo combination

DATACAMP, 2021. SMOTE | R. [online]. Available from: https://campus.datacamp.com/courses/fraud-detection-in-r/imbalanced-class-distributions?ex=10 [Accessed 22 Apr 2021].

FERNANDEZ-DELGADO,M.etal,2014.Do we Need Hundreds of Classifiers to Solve Real World Classification.[online].Available from:https://jmlr.csail.mit.edu/papers/v15/delgado14a.html [Accessed 16 Apr 2021].

LATE, E., 2018. An intuitive introduction to support vector machines using R – Part 1. [online]. Available from: https://eight2late.wordpress.com/2018/06/06/an-intuitive-introduction-to-support-vector-machines-using-r-part-1/ [Accessed 19 Apr 2021].

STACKOVERFLOW,2021. How to plot confusion matrix in R with caret package. [online].Available from: https://stackoverflow.com/questions/61309446/how-to-plot-confusion-matrix-in-r-with-caret-package [Accessed 19 Apr 2021].

THE EUROPEAN SPACE AGENCY.,2021.Near-Earth Objects - NEO Segment. [online].Available from: https://www.esa.int/Safety_Security/Near-Earth_Objects_-_NEO_Segment [Accessed 19 Apr 2021].