#### 



**Global Terrorism Analysis**

Harini Sairajan, Nikita Bhandare, Nikita Pathak, Thudcharin Wongprayuk

San Diego State University

MIS 620: E-Business Infrastructure: Data Science and Big Data - Fall 2017

December 19, 2017

**Executive Summary**

Since 2004, the global terrorist attacks have increased exponentially (Figure 1). We cannot deny that they are obviously dangerous threats to civilians, governments, and societies (Figure 2). Therefore, we study the historical data from The Global Terrorism Database (GTD) in order to understand the patterns of the terrorist attacks. Our study could potentially identify the terrorist groups based on their attack characteristics and determine success rate of their attacks in order to avoid or reduce the degree of destruction in the future.

|  |  |
| --- | --- |
|  |  |
| Figure 1: Yearly trend of terrorist attacks | Figure 2: Victims of terrorist attacks |

In the event of an attack, determining if an attack is successful could help governments take calculated measures against the attackers. Also, this could help them manage the Law and Order situation better. With our final model using random forest technique, we are able to predict whether an **attack would be successful or not with 80.7% accuracy**. According to our model, the top-five variables that are important to classify successful attacks are nwound (the number of people got injured by the attack), nkill (the number of people got killed by the attack), attack type, day of a month, and target type. We believe that if we did further studies on this topic and found the effective method to protect people from an attack, the rate of successful attacks would be decreased substantially. Finally, our pilot model is a starting point of the study to help governments and related organizations identify the important factors to protect civilians from terror attacks.

Another study is that how we could identify the terrorist group behind the attacks in a specific region. Our study was scoped down to South Asia region as a pilot study. In the **South Asia region, 49% of the terrorist attacks are unclaimed (Figure 3). So, implementing our model could reduce the rate of unclaimed attacks by 6.89%.** It could also provide an insight to Law Enforcement as to who can be the culprit of a terror attack. Likewise, this model could be used to study the attack patterns of terror activities in other specific regions or specific terror groups.

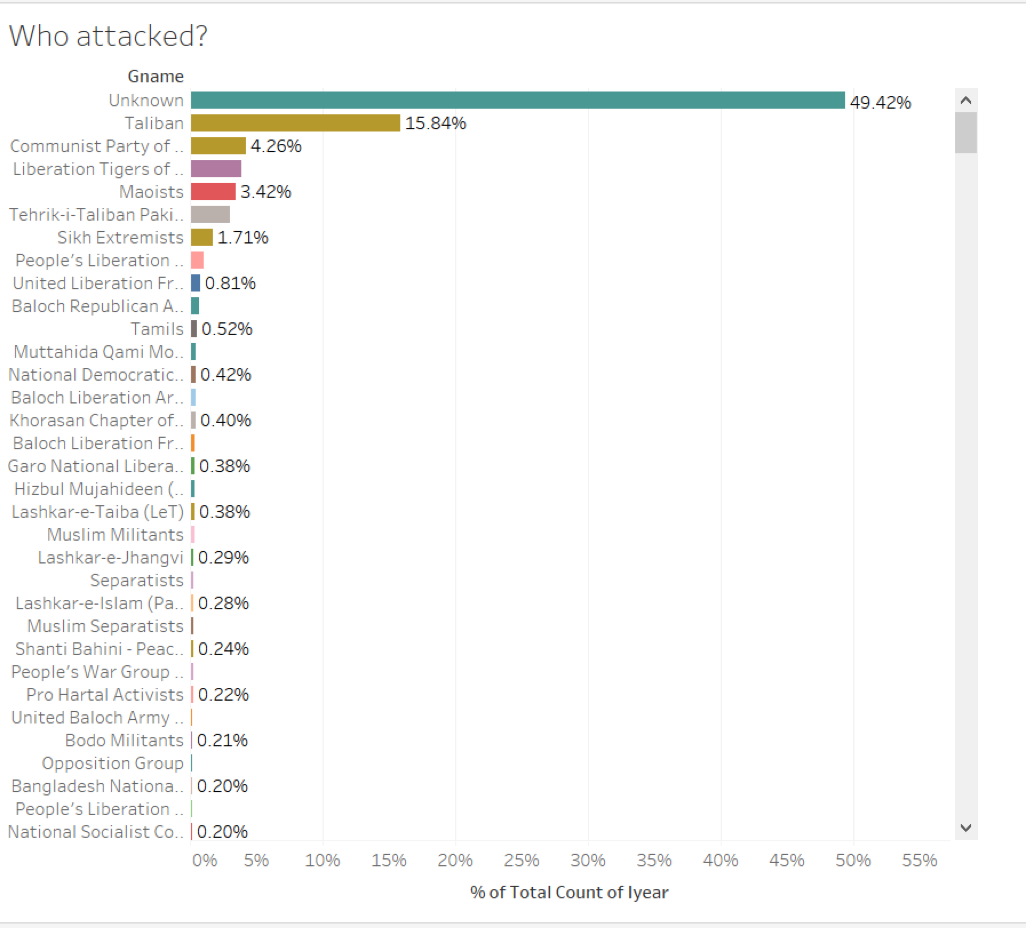


Figure 3: Terrorist Groups attacking in South Asia

**Discovery**

The Global Terrorism Database (GTD) is an open-source database including information on terrorist events around the world from 1970 through 2016 (with additional annual updates planned for the future). Unlike many other event databases, the GTD includes systematic data on domestic as well as transnational and international terrorist incidents that have occurred during this time period and now includes more than 170,000 cases. In addition, data for the incidents of terrorism for the year 1993 are not present in the GTD because they were lost prior to START’s compilation of the GTD from multiple data collection efforts.

For each GTD incident, information is available on the date and location of the incident, the weapons used and nature of the target, the number of casualties, and--when identifiable--the group or individual responsible.  
 Statistical information contained in the Global Terrorism Database is based on reports from a variety of open media sources. Information is not added to the GTD unless and until we have determined the sources are credible. Users should not infer any additional actions or results beyond what is presented in a GTD entry and specifically, users should not infer an individual associated with a particular incident was tried and convicted of terrorism or any other criminal offense. If new documentation about an event becomes available, an entry may be modified, as necessary and appropriate.  
 The National Consortium for the Study of Terrorism and Responses to Terrorism (START) makes the GTD available via this online interface in an effort to increase understanding of terrorist violence so that it can be more readily studied and defeated.

Characteristics of the GTD

|  |  |
| --- | --- |
| Duration of Attack | 1970 to 2016 (except 1993) |
| Number of Data Points | 135 |
| Number of Records | 170,351 |
| Geography | Worldwide |
| Variables (Data Point) | Data is collected for variables like location, tactics, success of attack, target group name, number of victims killed or wounded, number of perpetrators killed or wounded, targets and outcomes. |
| Size | 73.9 MB |

The GTD was designed to gather a wide variety of etiological and situational variables pertaining to each terrorist incident. Depending on availability of information, the database records up to 135 separate attributes of each incident, including approximately 75 coded variables that can be used for statistical analysis. These are collected under eight broad categories, as identified in the GTD Codebook, and include, whenever possible:

1) GTD ID and Date:

* Event Id: Incidents from the GTD follow a 12‐digit Event ID system.
* Year, Month, Day, Approximate Date
* Extended Incident: whether the duration of an incident extended more than 24 hours or not.

2) Incident Information:

* Incident Summary: A brief narrative summary of the incident, noting the “when, where, who, what, how, and why.”
* Inclusion Criteria

3) Incident Location:

* Country, region, state/province, city, vicinity, Location Description.
* Latitude and longitude
* Geocoding specificity

4) Attack Information:

* Attack Type: 8 categories + unknown.
  + Assassination, Hijacking, Kidnapping, Barricade Incident, Bombing/Explosion, Armed Assault, Unarmed Assault, Facility/Infrastructure Attack, and Unknown.
* Suicide Attack

5) Weapon Information:

* Weapon Type: 12 categories + unknown.
* Several sub weapon types.

6) Target Information:

* Target Type: 22 categories
* Several specific target/victim information, including names, nationalities, etc.

7) Perpetrator Information:

* Perpetrator Group Name: the name of the group that carried out the attack
* Several sub-group information, including number, claim, motive, etc.

8) Casualties and Consequences:

* Total Number of Fatalities, including Number of US Fatalities and Number of Perpetrator Fatalities
* Total Number of Injured, including Number of U.S. Injured and Number of Perpetrators Injured
* Property Damage, including damage extend, values and comments
* Total Number of Hostages/Kidnapping Victims, including US Hostages or Kidnapping Victims, kidnapping hours, countries, total ransom amount demanded, and number released/escaped/rescued

9) Additional Information and Sources:

* Additional relevant details about the attack, including International‐ Logistical, International‐ Miscellaneous, and sources.

**Data Preparation**

The raw data we retrieved from “National Consortium for the Study of Terrorism and Responses to Terrorism” was in csv format, which is comma-separated between attributes.

It can be opened by MS Excel in order to edit the content, or it can be read into a dataframe in R Studio.

There were few attribute in the dataset which were introduced after 1998. So the data for those attributes was collected after 1998. Since the data for those attributes was not missing at random we removed them from our analysis.   
 For few numerical variables we imputed the missing values in caret using bagimpute technique. The variables for which we imputed the missing values are as follows: Nkill, Nwound, Nkillter, Nwoundte, Nperps, and Nperpcap.

The Global Terrorism dataset consisted of unnecessary text fields. Since our task was focussed on classification we removed those unnecessary text fields. (Motive, Summary, latitude, longitude, provision state, city).  
 We removed variables from our analysis which had more than 50% missing values.  
Our dataset consisted of crit1,crit2,crit3. containing information about the motive behind the attack. To reduce the complexity we merged the data from these variables into a single variables. We replaced the missing values in the categorical variables(Nalty1, Weaponsubtype1,compclaim, Ransom ) with -99. The categorical features were encoded using dummy variables encoding (meaning they were transformed in the boolean variables).

After cleaning the dataset we performed feature selection using Boruta. Boruta is an all relevant feature selection wrapper algorithm, capable of working with any classification method that output variable importance measure (VIM); by default, Boruta uses Random Forest. The method performs a top-down search for relevant features by comparing original attributes' importance with importance achievable at random, estimated using their permuted copies, and progressively eliminating irrelevant features to stabilise that test.

i) Boruta iteratively compares importances of attributes with importances of shadow attributes, created by shuffling original ones.

ii) Attributes that have significantly worst importance than shadow ones are being consecutively dropped. On the other hand, attributes that are significantly better than shadows are admitted to be Confirmed. Shadows are re-created in each iteration.

iii) Algorithm stops when only Confirmed attributes are left, or when it reaches maxRuns importance source runs. If the second scenario occurs, some attributes may be left without a decision. They are claimed Tentative.

After performing feature selection we found 37 important variables. Figure 4 shows the variable importance graph.

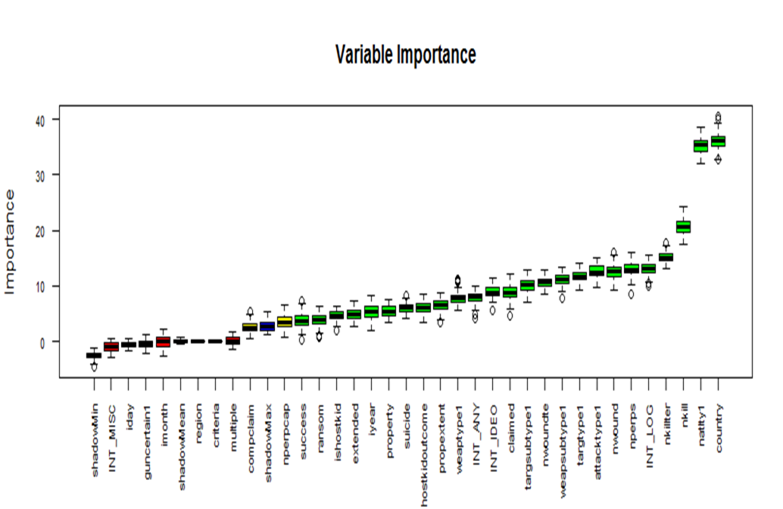


Figure 4: Variable Importance based on feature selections

**Model Planning**

Our study focuses on two main goals, which are the prediction of success or fail attacks of worldwide terrorist groups, and the prediction of terrorist group based on the attack characteristics in a specific region, such as South Asia. Therefore, the model planning stage would be adjusted to each specific goal since the predictors and dependent variables are different.

The prediction of success or fail attacks

The success or fail attacks are defined according to the tangible effects of the attack. The definition of a successful attack depends on the type of attack. For example, a bomb that exploded in a building would be counted as a success even if the building was not collapsed. Another example is an assassination would be successful only if the intended target is killed.

At this stage, we plan to build binary classification models to classify the success or fail attacks based on the selective attributes that we got from Global Terrorism Database. The models we built included the followings: Decision Tree (using “rpart”), Random Forest (using “rf”), Bagging (using “TreeBag”), Boosting (using “ada”), Artificial Neural Network (using “ada”), and Naive Bayes (using “nb”).

However, we faced some challenges with highly skewed data set because the data is imbalanced. We noticed that ‘Fail’ attack case had only 12,064 records, but ‘Success’ attack case had 90,850 records from our data set. The imbalanced data would cause the poor performance of the model, saying that the model could not classify the ‘Fail’ case because the training data was dominated by ‘Success’ case. Then, we applied the method to deal with imbalanced data using several techniques such as down sampling, up sampling, SMOTE algorithm, and ROSE algorithm. Figure 5 shows the performance result of each method developed based on the naive bayes classifier.

As a result, we decided to use ROSE algorithm to manage imbalanced data since it achieved the highest ROC. We will discuss more on the development of different models using ROSE training data in the next section, Model Building.

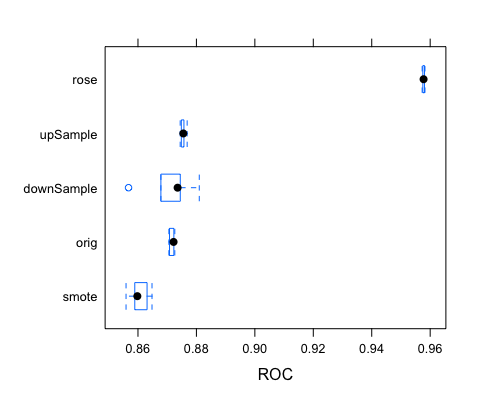


Figure 5: Box Plot comparing the performance of each methodology to handle imbalanced data

The prediction of terrorist group based on the attack characteristics

We decided to perform predictive analysis on terrorist group because when a terrorist attack occurs, sometimes there may be no groups claiming this attack. Besides, it can also happen that a terrorist groups falsely claim an attack, and in this case, law and order can check on the probability of the claim made by the terrorist group to be true.

While analyzing the Global Terrorism Dataset, we found that Middle East & North Africa and South Asia regions have the highest number of incidents from 1970 to 2016.

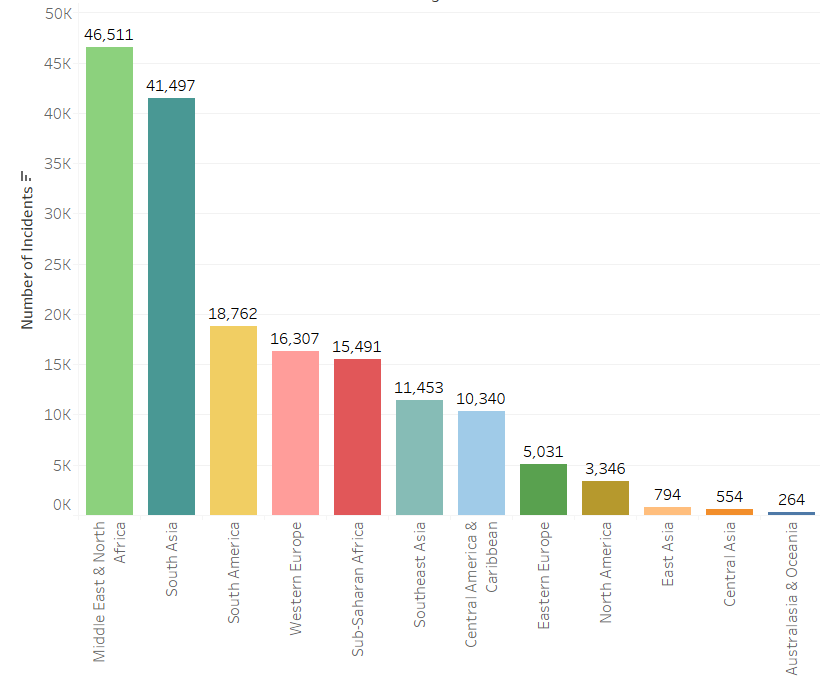


Figure 6: The number of incidents by region

We have noticed that there are 48% of the terrorist attacks are not claimed by any terrorist group and it would be interesting to predict terrorist groups behind the attack.

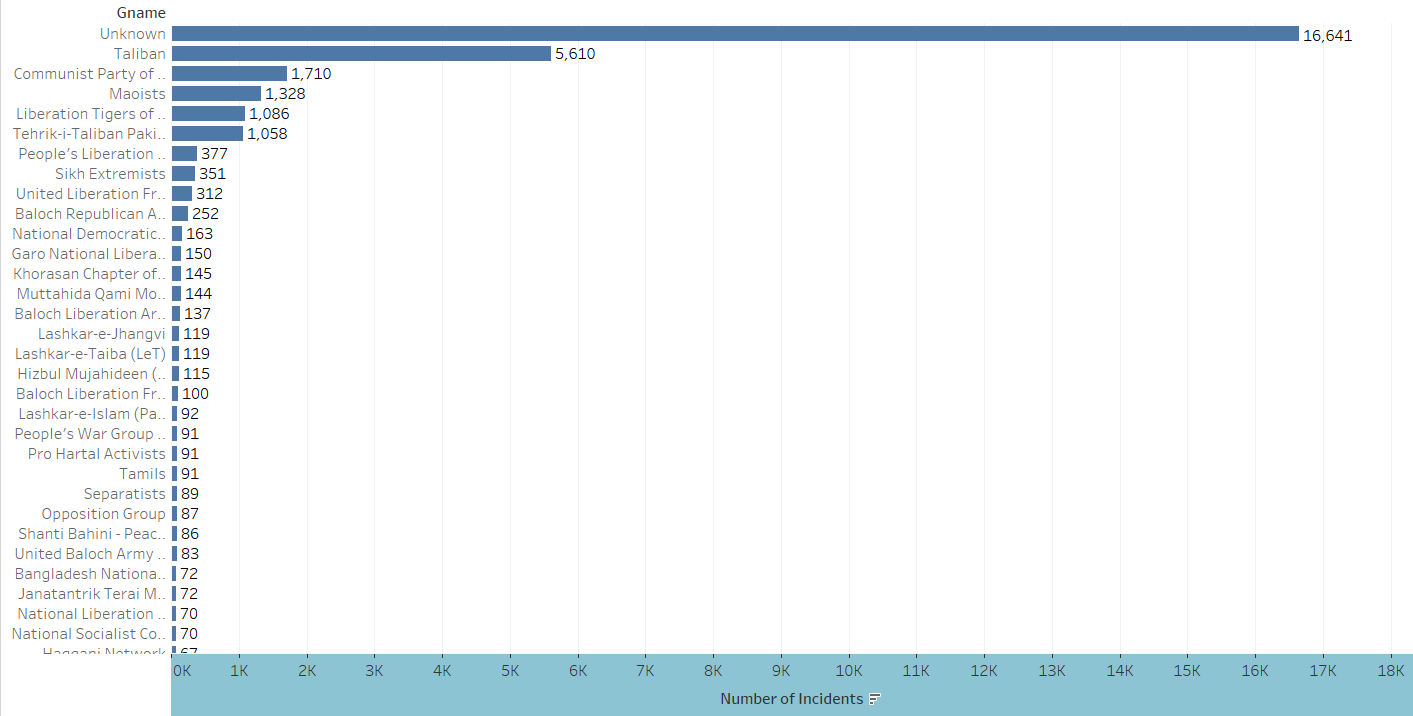
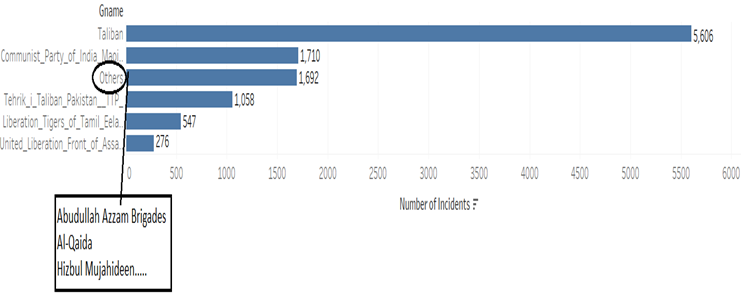


Figure 7: The number of incidents by terrorist group name

For this prediction goal, we have decided to focus our analysis on South Asia region. We have dropped all attacks that were marked as doubtful as whether they should be considered terrorist attacks or not and all the attacks which were not affiliated to the terrorist groups. Also, we have manual selected the “real terrorist groups’ by using search engine (for example, “Maoist” or “Extremists” should not be considered as terrorist groups. Since we have more than 1000 terrorist groups in South Asia region and it was taking huge amount of time to process with such huge number of classes, we decided to model the problem as a multi-classification problem by building a model for South Asia region sub-dataset and consider 5 different classes as the 5 terrorist groups with more incidents in each region, plus one class that includes all the other groups in South Asia region ("Other").



During this prediction task we used the following features:

* Year of the attack (Numeric)
* Existence of multiple attacks (Boolean)
* Success or not of the attack (Boolean)
* Existence of suicide regarding one or more perpetrators (Boolean)
* Number of perpetrators (Numeric)
* Number of perpetrators captured (Numeric)
* Number of killings (Numeric)
* Number of wounded (Numeric)
* Number of hostages kidnapped (Numeric)
* Asked ransom amount (Numeric)
* Paid ransom amount (Numeric)
* Number of released hostages (Numeric)
* Type of the attack (Categorical)
* Type of the target (Categorical)
* Nationality of the victims (Categorical)
* Type of the weapon (Categorical)
* Subtype of the weapon (Categorical)

The categorical features were encoded using dummy variables encoding (meaning they were transformed in the boolean variables).

We have decided to train following models using a One-vs-All Multiclass Classifier over our dataset: Decision Tree, Random Forest, Bagging, Boosting, Artificial Neural Network, Naive Bayes, Support Vector Machine

**Model Building**

The prediction of success or fail attacks

After the model planning stage, we splitted the terrorist attack records into 70% training set and 30% testing set. We started to build the binary classification models, according to Model Planning, from the training set. The positive class in our case is ‘Fail’ label, and the negative class is ‘Success’ label since our goal is to find the way to increase the number of fail attacks. In other words, we want to reduce the percentage of success attacks that are harmful to people and societies.

Figure 8 shows the training performance of each model using ROC as the metric. TreeBag and RandomForest models performed very well on training data, while Decision Tree model using rpart performed worse than other models.

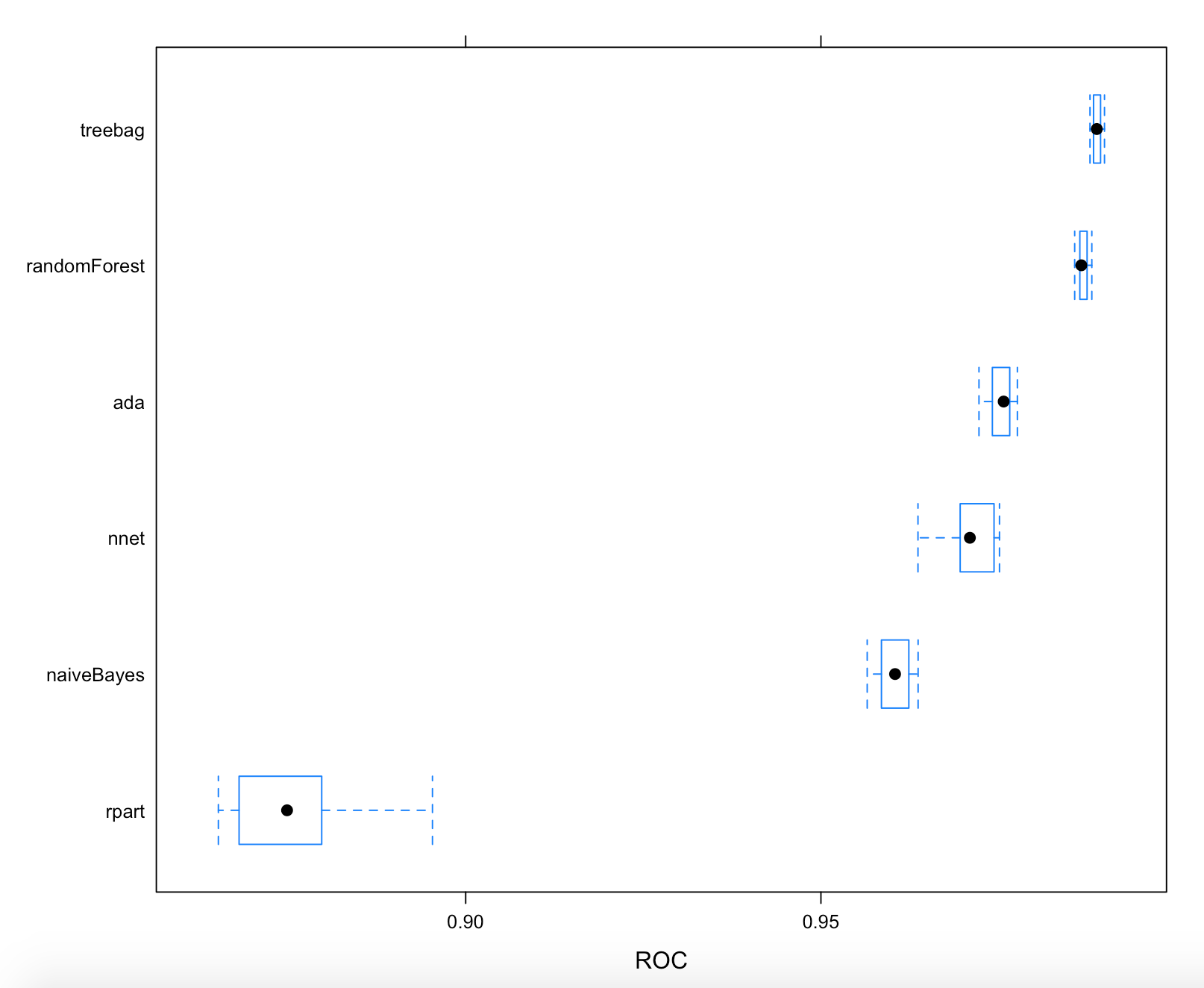


Figure 8: Box Plot comparing the training performance for predicting success/fail attacks

The prediction of terrorist group behind the attacks

First, it's important to analyze the frequencies of the different classes in South Asia region after 1998, which is presented below:

|  |  |
| --- | --- |
| **Terrorist group** | **Incidents** |
| Taliban | 5606 |
| Other | 1692 |
| Communist Party of India - Maoist (CPI-Maoist) | 1710 |
| Liberation Tigers of Tamil Eelam (LTTE) | 547 |
| Tehrik-i-Taliban Pakistan (TTP) | 1058 |
| United National Liberation Front (UNLF) | 276 |

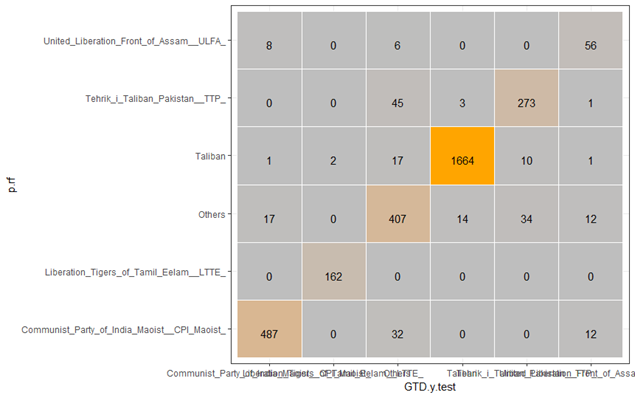
It's noticeable from this table that the classes "Taliban" and "Other" are the most dominant ones. An analysis of the confusion matrix on Figure 9 reveals that the model is balanced even with the class unbalance here. 

Figure 9: The analysis of confusion matrix

**Results and Performance**

The prediction of success or fail attacks

Figure 10 shows the ROC curves of testing data set on the models we developed. The ROC curves confirmed the performance of training data set since “Random Forest” and “TreeBag” models are still the top performers, while “rpart” result is disappointing.

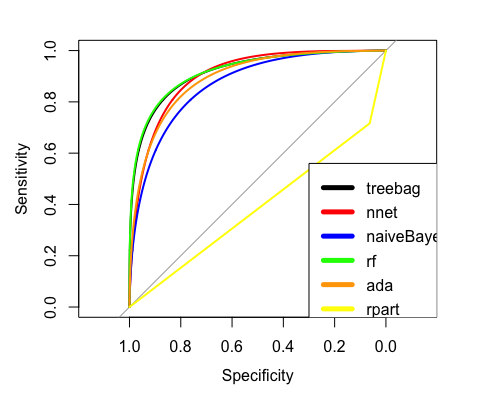


Figure 10: ROC Curve on test datasets for predicting success/fail attacks

According to the table below, the model that achieved the highest performance on test data is “Random Forest” Model. Based on test data which was 30% of the total, the accuracy is 80.7%. The sensitivity, or true positive rate, is 89.64%, and the specificity, or true negative rate is 79.52%.

|  |  |  |
| --- | --- | --- |
| **Model Name** | **AUC** | **Accuracy** |
| TreeBag | 0.9095 | 0.7203 |
| nnet | 0.9079 | 0.6782 |
| RandomForest | **0.9202** | **0.807** |
| ada | 0.8955 | 0.5358 |
| rpart | 0.3897 | 0.3603 |
| naiveBayes | 0.8678 | 0.6263 |

Lastly, we identified the variable importance based on the “Random Forest” classification model. The results is shown in Figure 11, indicating that nwound, nkill, attacktype1, iday, and targtype1 are the top-5 important variables to classify between “Fail” or “Success” attacks.

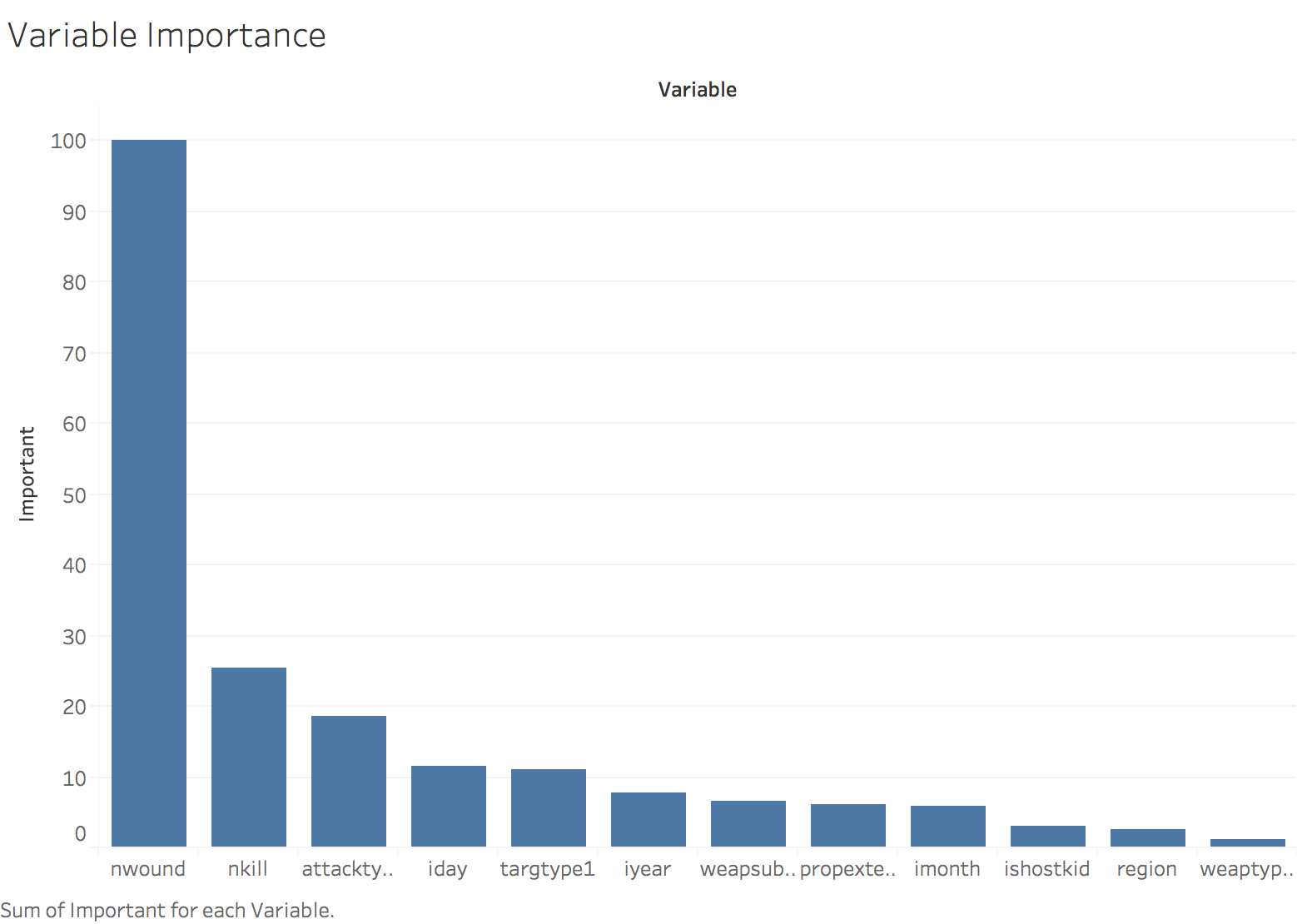
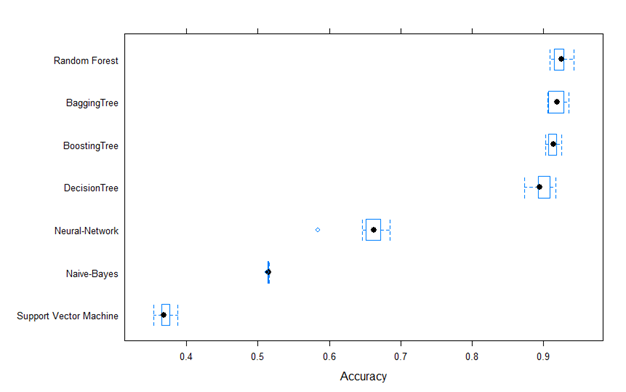


Figure 11: Variable Importance from the random forest model for predicting success/fail attacks

The prediction of terrorist group behind the attacks

Figure 12 shows the test performance of each model using Accuracy and Mean\_Sensitivity curve as the metric. Random Forest model has performed very well on test data, while Naive Bayes model using performed worse than other models.



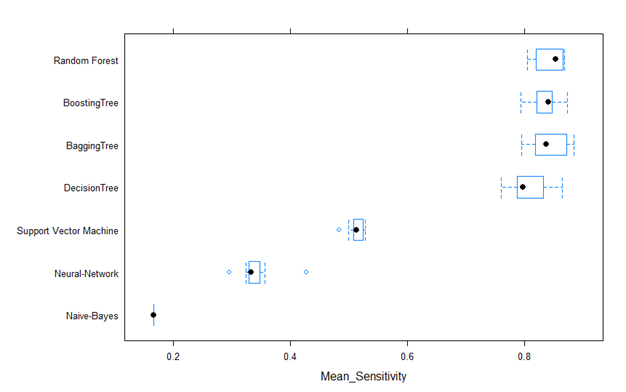


Figure 12: Box plot of Accuracy and Mean\_Sensitivity

**Discussion and Recommendations**

* Predictive model for determining the success of the attack:
  + Not all terror attacks can be declared successful. Using our model a terror attack can be categorized as a successful or not.
  + Law enforcement could use this to decide on the direction , plan their resources and money to counter a possible successful attack.
* Predictive model for determining the terrorist group name:
  + This model could help predict the Terrorist group name for **unclaimed** attacks.
  + Some terror attacks could be **falsely claimed** to mislead investigation or for propaganda purposes. This model could help catch potential perpetrators.
* We recommend Implementing the model as a pilot first to learn on performance and precision of the model.
* This predictive model could be extended for all the remaining regions of the world.
* The model could be altered in a way that it can predict the probability of success rate for a particular terror group.

**Appendix**

Appendix 1: R source code for predicting the success or fail attacks

library("rpart")

library("rpart.plot")

library("dplyr")

library("e1071")

library("caret")

library("ROCR")

library("plyr")

library("doParallel")

library("DMwR")

library("ROSE")

library("pROC")

#Download the dataset and place the file into working directory

#sample <- read.csv("globalterrorismdb\_0617dist.csv",header=TRUE,sep=",")

#saveRDS(sample, "globalterrorism.rds")

sample <- readRDS("globalterrorism.rds")

sample <- sample[,c("iyear","imonth","iday",

"country","region",

"extended", "crit1", "crit2", "crit3", "doubtterr", "multiple", "suicide", "ishostkid", "ransom", "INT\_ANY",

"attacktype1","weaptype1","weapsubtype1",

"targtype1","natlty1","propextent",

"nperps","nkill","nwound",

"gname","success")]

str(sample)

# analyze data in 1998 and after to avoid non-recorded data.

sample <- sample[sample$iyear >= 1998, ]

sample$iyear <- as.factor(sample$iyear)

sample$imonth <- as.factor(sample$imonth)

sample$iday <- as.factor(sample$iday)

sample$country <- as.factor(sample$country)

sample$region <- as.factor(sample$region)

sample$attacktype1 <- as.factor(sample$attacktype1)

sample$weaptype1 <- as.factor(sample$weaptype1)

sample$weapsubtype1 <- as.factor(sample$weapsubtype1) #NA

sample$targtype1 <- as.factor(sample$targtype1)

sample$natlty1 <- as.factor(sample$natlty1) #NA

sample$propextent <- as.factor(sample$propextent) #NA

sample$success <- as.factor(sample$success)

sample$extended <- as.factor(sample$extended)

sample$crit1 <- as.factor(sample$crit1)

sample$crit2 <- as.factor(sample$crit2)

sample$crit3 <- as.factor(sample$crit3)

sample$doubtterr <- as.factor(sample$doubtterr)

sample$multiple <- as.factor(sample$multiple)

sample$suicide <- as.factor(sample$suicide)

sample$ishostkid <- as.factor(sample$ishostkid) #NA

sample$ransom <- as.factor(sample$ransom) #NA

sample$INT\_ANY <- as.factor(sample$INT\_ANY)

sample$gname <- as.character(sample$gname)

# replace NA Value

sample$propextent[which(is.na(sample$propextent))] <- 4 #replace NA with 4 = Unknown

levels(sample$natlty1) <- c(levels(sample$natlty1),"-99")

sample$natlty1[which(is.na(sample$natlty1))] <- -99 #replace NA with -99

levels(sample$weapsubtype1) <- c(levels(sample$weapsubtype1),"-99")

sample$weapsubtype1[which(is.na(sample$weapsubtype1))] <- -99 #replace NA with -99

sample$ishostkid[which(is.na(sample$ishostkid))] <- -9 #replace NA with -9 = Unknown

sample$ransom[which(is.na(sample$ransom))] <- -9 #replace NA with -9 = Unknown

summary(sample$nkill)

sample$nkill[which(is.na(sample$nkill))] <- 1 # median = 1 assigned to NA

summary(sample$nwound)

sample$nwound[which(is.na(sample$nwound))] <- 0 # median = 0 assigned to NA

########## Predict success/fail ##########

levels(sample$success)[levels(sample$success)=="1"] <- "SUCCESS"

levels(sample$success)[levels(sample$success)=="0"] <- "FAIL"

drops <- c("country", "nperps", "gname", "natlty1")

sample <- sample[,!(names(sample) %in% drops)]

table(sample$success)

set.seed(1000)

inTrain <- createDataPartition(y=sample$success, p=.70, list=FALSE)

trainSplit <- sample[inTrain,]

length(trainSplit$success)

testSplit <- sample[-inTrain,]

length(testSplit$success)

#cl <- makeCluster(detectCores())

#registerDoParallel(cl)

#getDoParWorkers()

#set.seed(1000)

#smote\_train <- SMOTE(success ~ ., data = trainSplit)

#stopCluster(cl)

#table(smote\_train$success)

cl <- makeCluster(detectCores())

registerDoParallel(cl)

getDoParWorkers()

set.seed(1000)

rose\_train <- ROSE(success ~ ., data = trainSplit)$data

stopCluster(cl)

table(rose\_train$success)

y.train <- trainSplit$success

x.train <- trainSplit[,!(names(trainSplit) %in% c("success"))]

#y.train.up <- up\_train$success

#x.train.up <- up\_train[,!(names(up\_train) %in% c("success"))]

#y.train.down <- down\_train$success

#x.train.down <- down\_train[,!(names(down\_train) %in% c("success"))]

#y.train.smote <- smote\_train$success

#x.train.smote <- smote\_train[,!(names(smote\_train) %in% c("success"))]

y.train.rose <- rose\_train$success

x.train.rose <- rose\_train[,!(names(rose\_train) %in% c("success"))]

y.test <- testSplit$success

x.test <- testSplit[,!(names(testSplit) %in% c("success"))]

ctrl <- trainControl(method="cv", number=10,

classProbs=TRUE,

summaryFunction = twoClassSummary, # twoClassSummary for binary

allowParallel = TRUE)

##### NaiveBayes #####

cl <- makeCluster(detectCores())

registerDoParallel(cl)

getDoParWorkers()

set.seed(100)

m.nb.rose <- train(y=y.train.rose, x=x.train.rose,

trControl = ctrl,

metric = "ROC",

method = "nb")

stopCluster(cl)

m.nb.rose

getTrainPerf(m.nb.rose)

varImp(m.nb.rose)

##### Decision Tree #####

cl <- makeCluster(detectCores())

registerDoParallel(cl)

getDoParWorkers()

set.seed(100)

m.rpart.rose <- train(y=y.train.rose, x=x.train.rose,

trControl = ctrl,

metric = "ROC",

method = "rpart")

stopCluster(cl)

m.rpart.rose

getTrainPerf(m.rpart.rose)

varImp(m.rpart.rose)

##### Random Forest #####

cl <- makeCluster(detectCores())

registerDoParallel(cl)

getDoParWorkers()

set.seed(100)

m.rf.rose <- train(y=y.train.rose, x=x.train.rose,

trControl = ctrl,

metric = "ROC",

method = "rf")

stopCluster(cl)

m.rf.rose

getTrainPerf(m.rf.rose)

varImp(m.rf.rose)

##### Tree Bag #####

cl <- makeCluster(detectCores())

registerDoParallel(cl)

getDoParWorkers()

set.seed(100)

m.treebag.rose <- train(y=y.train.rose, x=x.train.rose,

trControl = ctrl,

metric = "ROC",

method = "treebag")

stopCluster(cl)

m.treebag.rose

getTrainPerf(m.treebag.rose)

varImp(m.treebag.rose)

##### Neural Network #####

cl <- makeCluster(detectCores())

registerDoParallel(cl)

getDoParWorkers()

set.seed(100)

m.nnet.rose <- train(y=y.train.rose, x=x.train.rose,

trControl = ctrl,

metric = "ROC",

method = "nnet")

stopCluster(cl)

m.nnet.rose

getTrainPerf(m.nnet.rose)

varImp(m.nnet.rose)

##### Boosting ##### xxx

cl <- makeCluster(detectCores())

registerDoParallel(cl)

getDoParWorkers()

set.seed(100)

m.ada.rose <- train(y=y.train.rose, x=x.train.rose,

trControl = ctrl,

metric = "ROC",

method = "ada")

stopCluster(cl)

m.ada.rose

getTrainPerf(m.ada.rose)

varImp(m.ada.rose)

#######################

##### PERFORMANCE #####

#######################

# Compare TRAINING performance of cross-validation runs

rValues <- resamples(list(naiveBayes=m.nb.rose, rpart=m.rpart.rose, randomForest=m.rf.rose, treebag=m.treebag.rose, ada=m.ada.rose, nnet=m.nnet.rose))

bwplot(rValues, metric="ROC")

bwplot(rValues, metric="Sens") #Sensitvity

bwplot(rValues, metric="Spec")

p.treebag <- predict(m.treebag.rose, x.test)

confusionMatrix(p.treebag, y.test) #0.7203

p.nnet <- predict(m.nnet.rose, x.test)

confusionMatrix(p.nnet, y.test) #0.6782

p.rpart <- predict(m.rpart.rose, x.test)

confusionMatrix(p.rpart, y.test) #0.3603

p.rf <- predict(m.rf.rose, x.test)

confusionMatrix(p.rf, y.test) #0.807

p.nb <- predict(m.nb.rose, x.test)

confusionMatrix(p.nb, y.test) #0.6263

p.ada <- predict(m.ada.rose, x.test)

confusionMatrix(p.ada, y.test) #0.5358

p.treebag.prob <- predict(m.treebag.rose, x.test, type = "prob")

p.nnet.prob <- predict(m.nnet.rose, x.test, type = "prob")

p.rf.prob <- predict(m.rf.rose, x.test, type = "prob")

p.ada.prob <- predict(m.ada.rose, x.test, type = "prob")

p.rpart.prob <- predict(m.rpart.rose, x.test, type = "prob")

p.nb.prob <- predict(m.nb.rose, x.test, type = "prob")

# using FAIL as positive class

p.treebag.roc <- roc(response = y.test, predictor = p.treebag.prob$FAIL)

p.nnet.roc <- roc(response = y.test, predictor = p.nnet.prob$FAIL)

p.rf.roc <- roc(response = y.test, predictor = p.rf.prob$FAIL)

p.ada.roc <- roc(response = y.test, predictor = p.ada.prob$FAIL)

p.rpart.roc <- roc(response = y.test, predictor = p.rpart.prob$FAIL)

p.nb.roc <- roc(response = y.test, predictor = p.nb.prob$FAIL)

auc(p.treebag.roc)

auc(p.nnet.roc)

auc(p.rf.roc)

auc(p.ada.roc)

auc(p.rpart.roc)

auc(p.nb.roc)

plot(smooth(p.treebag.roc), col="black")

plot(smooth(p.nnet.roc), add=T, col="red")

plot(smooth(p.nb.roc), add=T, col="blue")

plot(smooth(p.rf.roc), add=T, col="green")

plot(smooth(p.ada.roc), add=T, col="orange")

plot(p.rpart.roc, add=T, col="yellow")

legend(x=.3, y=.56, cex=1, legend=c("treebag","nnet", "naiveBayes", "rf", "ada", "rpart"), col=c("black", "red", "blue", "green", "orange", "yellow"), lwd=5)

varImp(m.rf.rose)

Appendix 2: R source code for predicting terrorist group in South Asia

install.packages(c("ISLR","rpart.plot","moments","PerformanceAnalytics","tidyr","dummies","gridExtra"))  
library(ISLR)  
library(caret)  
library(mice)  
library(psych)  
library(pROC)  
library(rpart)  
library(rpart.plot)  
library(cluster)  
library(tree)   
library(e1071)  
library(moments)  
library(corrplot)  
library(PerformanceAnalytics)  
library(lattice)  
library(doParallel)  
library(reshape2)  
library(tidyr)  
library(dummies)  
library(ggplot2)  
library(grid)  
library(gridExtra)  
library(MLmetrics)  
  
#setting the working directory  
  
getwd()  
setwd("/store/studenthome/mis620/2017Fall/adity2007@gmail.com/Data Crusaders/")  
  
#Download the dataset and Read data file in RDS  
  
GTD <-read.csv("GTD\_attempt 7.csv")  
saveRDS(GTD, "GTD.rds")  
  
GTD <- readRDS("GTD.rds")  
  
str(GTD)  
  
#convert the variables into factors  
convert <- c("INT\_LOG","INT\_IDEO","INT\_MISC","INT\_ANY","guncertain1","multiple","success","suicide",  
 "extended","country","region","attacktype1","targtype1","targsubtype1","natlty1","claimed","compclaim","weaptype1",  
 "weapsubtype1","propextent","property","ransom","ishostkid","hostkidoutcome")  
  
GTD[,convert] <- data.frame(apply(GTD[convert],2,as.factor))  
  
GTD <- GTD[,-c(9,10,11,13,27)]  
  
str(GTD)  
  
GTD$iyear <- as.factor(GTD$iyear)  
GTD$imonth <- as.factor(GTD$imonth)  
GTD$iday <- as.factor(GTD$iday)  
GTD$nperps <- as.factor(GTD$nperps)  
GTD$nperpcap <- as.factor(GTD$nperpcap)  
  
#will walk through basic imputing  
set.seed(192)  
  
#caret has preprocess function - we are imputing missing data with bag (ensemble decision trees), scaling and centering, and filtering out  
#highly correlated predictors  
GTD.prepmodel <- preProcess(GTD, method=c("bagImpute"))  
  
  
GTD.prepmodel$method  
  
#apply pre processing model to training/test data  
GTD.prepmodel <- predict(GTD.prepmodel, GTD)  
  
str(GTD.prepmodel)  
  
GTD.prepmodel$nkill <- ceiling(GTD.prepmodel$nkill)  
GTD.prepmodel$nkillter <- ceiling(GTD.prepmodel$nkillter)  
GTD.prepmodel$nwound <- ceiling(GTD.prepmodel$nwound)  
GTD.prepmodel$nwoundte <- ceiling(GTD.prepmodel$nwoundte)  
  
class(GTD.prepmodel$iyear)  
#GTD1<- subset(GTD, !(GTD$nperps %in% c(-99, -9)))  
GTD.prepmodel$iyear <- as.numeric(GTD.prepmodel$iyear)  
GTD.prepmodel$imonth <- as.numeric(GTD.prepmodel$imonth)  
GTD.prepmodel$iday <- as.numeric(GTD.prepmodel$iday)  
GTD.prepmodel$nperps <- as.numeric(GTD.prepmodel$nperps)  
GTD.prepmodel$nperpcap <- as.numeric(GTD.prepmodel$nperpcap)  
  
str(GTD.prepmodel)  
  
  
write.csv(GTD\_SA,file= "GTD\_AllRegions.csv")  
  
#GTD$nkillter, GTD$nwound, GTD$nperps  
GTD\_SA <- subset(GTD.prepmodel, (GTD$region %in% 6))  
  
#removing country\_txt, region\_txt, propextent\_txt, hostkidoutcome\_txt, attack\_type1\_txt, targtype1\_txt  
# targsubtype1\_txt, natlty1\_txt, weapsubtype1\_txt  
  
not\_reqd <- c("country\_txt", "region\_txt","attacktype1\_txt", "targtype1\_txt", "targsubtype1\_txt",  
 "natlty1\_txt", "weaptype1\_txt", "weapsubtype1\_txt", "propextent\_txt", "hostkidoutcome\_txt")  
  
GTD\_SA<- GTD\_SA[, !names(GTD\_SA) %in% not\_reqd]  
  
#converting gname to string for replacing the phaltu terrorist groups to others  
GTD\_SA$gname <- as.character(GTD\_SA$gname)  
  
  
#Subsetting terrorist group names by selecting top 5 from a region  
  
hai <- c("Abdullah Azzam Brigades","Al\_Badr","Al\_Qaida","Al\_Qaida in the Indian Subcontinent","Al\_Qaida Network for Southwestern Khulna Division",  
 "Al\_Umar Mujahideen","All Tripura Tiger Force \_ATTF\_","Ansarul Islam \_Pakistan\_","Babbar Khalsa International \_BKI\_","Baloch Liberation Army \_BLA\_",  
 "Baloch Liberation Front \_BLF\_","Baloch Liberation Tigers \_BLT\_","Baloch Republican Army \_BRA\_","Communist Party of India\_ Marxist",  
 "Communist Party of India\_Maoist \_CPI\_Maoist\_","Haqqani Network","Harakat ul\_Mujahidin \_HuM\_","Harakat ul\_Mujahidin Al\_Almi","Harkatul Jihad\_e\_Islami",  
 "Hizbul Mujahideen \_HM\_","Indian Mujahideen","Islamic Movement of Uzbekistan \_IMU\_","Jamaat\_E\_Islami \_Bangladesh\_","Jamaat\_E\_Islami \_India/Pakistan\_",  
 "Jama'atul Mujahideen Bangladesh \_JMB\_","Jamiat ul\_Mujahedin \_JuM\_","Jundallah \_Pakistan\_","Kanglei Yawol Kanna Lup \_KYKL\_",  
 "Kangleipak Communist Party \_KCP\_","Khorasan Chapter of the Islamic State","Lashkar\_e\_Jhangvi","Lashkar\_e\_Taiba \_LeT\_","Liberation Tigers of Tamil Eelam \_LTTE\_",  
 "Maoist Communist Center \_MCC\_","National Democratic Front of Bodoland \_NDFB\_","National Liberation Front of Tripura \_NLFT\_","New People's Army \_NPA\_",  
 "People's Liberation Army \_India\_","People's Revolutionary Party of Kangleipak \_PREPAK\_","Sipah\_e\_Sahaba/Pakistan \_SSP\_","Students Islamic Movement of India \_SIMI\_",  
 "Taliban","Tamil Nadu Liberation Army","Tehrik\_e\_Nafaz\_e\_Shariat\_e\_Mohammadi \_TNSM\_","Tehrik\_i\_Taliban Pakistan \_TTP\_","United Liberation Front of Assam \_ULFA\_",  
 "United National Liberation Front \_UNLF\_")  
  
GTD\_SA <- GTD\_SA[GTD\_SA$gname %in% hai,]  
  
table(GTD\_SA$gname)  
  
  
ExcludingTop5 <- GTD\_SA[GTD\_SA$gname %in% c("Abdullah Azzam Brigades","Al\_Badr","Al\_Qaida","Al\_Qaida in the Indian Subcontinent","Al\_Qaida Network for Southwestern Khulna Division",  
 "Al\_Umar Mujahideen","All Tripura Tiger Force \_ATTF\_","Ansarul Islam \_Pakistan\_","Babbar Khalsa International \_BKI\_","Baloch Liberation Army \_BLA\_",  
 "Baloch Liberation Front \_BLF\_","Baloch Liberation Tigers \_BLT\_","Baloch Republican Army \_BRA\_","Communist Party of India\_ Marxist",  
 "Haqqani Network","Harakat ul\_Mujahidin \_HuM\_","Harakat ul\_Mujahidin Al\_Almi","Harkatul Jihad\_e\_Islami",  
 "Hizbul Mujahideen \_HM\_","Indian Mujahideen","Islamic Movement of Uzbekistan \_IMU\_","Jamaat\_E\_Islami \_Bangladesh\_","Jamaat\_E\_Islami \_India/Pakistan\_",  
 "Jama'atul Mujahideen Bangladesh \_JMB\_","Jamiat ul\_Mujahedin \_JuM\_","Jundallah \_Pakistan\_","Kanglei Yawol Kanna Lup \_KYKL\_",  
 "Kangleipak Communist Party \_KCP\_","Khorasan Chapter of the Islamic State","Lashkar\_e\_Jhangvi","Lashkar\_e\_Taiba \_LeT\_",  
 "Maoist Communist Center \_MCC\_","National Democratic Front of Bodoland \_NDFB\_","National Liberation Front of Tripura \_NLFT\_","New People's Army \_NPA\_",  
 "People's Liberation Army \_India\_","People's Revolutionary Party of Kangleipak \_PREPAK\_","Sipah\_e\_Sahaba/Pakistan \_SSP\_","Students Islamic Movement of India \_SIMI\_",  
 "Tamil Nadu Liberation Army","Tehrik\_e\_Nafaz\_e\_Shariat\_e\_Mohammadi \_TNSM\_","United National Liberation Front \_UNLF\_"),]  
  
table(ExcludingTop5$gname)  
  
GTD\_SA$gname <- replace(GTD\_SA$gname, GTD\_SA$gname %in% ExcludingTop5$gname,"Others")  
  
  
table(GTD\_SA$gname)  
  
  
#phi coefficient > .7 than highly corelated then combine this  
  
GTD\_SA$gname <- gsub(" ", "\_",GTD\_SA$gname)  
  
#converting the gnames to factor  
GTD\_SA$gname <- as.factor(GTD\_SA$gname)  
  
write.csv(GTD\_SA,file= "GTD\_Final.csv")  
  
  
############\_\_\_\_\_\_\_\_\_\_\_\_FEATURE SELECTION USING BORUTA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_###################  
  
  
# selecting important variables using Boruta  
library(Boruta)  
# Decide if a variable is important or not using Boruta  
boruta\_output <- Boruta(GTD\_SA$gname ~ ., data=GTD\_SA, doTrace=2) # perform Boruta search  
# Confirmed 10 attributes: Humidity, Inversion\_base\_height, Inversion\_temperature, Month, Pressure\_gradient and 5 more.  
# Rejected 3 attributes: Day\_of\_month, Day\_of\_week, Wind\_speed.  
boruta\_signif <- names(boruta\_output$finalDecision[boruta\_output$finalDecision %in% c("Confirmed", "Tentative")]) # collect Confirmed and Tentative variables  
print(boruta\_signif) # significant variables  
  
# 100 Runs  
#=>[1] "iyear" "extended" "country" "success" "suicide"   
#=>[6] "attacktype1" "targtype1" "targsubtype1" "natlty1" "nperps"   
#=>[11] "nperpcap" "claimed" "compclaim" "weaptype1" "weapsubtype1"   
#=>[16] "nkill" "nkillter" "nwound" "nwoundte" "property"   
#=>[21] "propextent" "ishostkid" "ransom" "hostkidoutcome" "INT\_LOG"   
#=>[26] "INT\_IDEO" "INT\_ANY"   
  
### 50 runs  
#=>[1] "iyear" "imonth" "extended" "country" "multiple"   
#=>[6] "success" "suicide" "attacktype1" "targtype1" "targsubtype1"   
#=>[11] "natlty1" "guncertain1" "nperps" "nperpcap" "claimed"   
#=>[16] "compclaim" "weaptype1" "weapsubtype1" "nkill" "nkillter"   
#=>[21] "nwound" "nwoundte" "property" "propextent" "ishostkid"   
#=>[26] "ransom" "hostkidoutcome"  
plot(boruta\_output, cex.axis=.7, las=2, xlab="", main="Variable Importance") # plot variable importance  
  
######\_\_\_\_\_\_\_\_\_\_\_\_Creating Training and Test splits\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_###############

#=>[1] "iyear" "extended" "country" "success" "suicide"   
#=>[6] "attacktype1" "targtype1" "targsubtype1" "natlty1" "nperps"   
#=>[11] "nperpcap" "claimed" "compclaim" "weaptype1" "weapsubtype1"   
#=>[16] "nkill" "nkillter" "nwound" "nwoundte" "property"   
#=>[21] "propextent" "ishostkid" "ransom" "hostkidoutcome" "INT\_LOG"   
#=>[26] "INT\_IDEO" "INT\_ANY"   
  
#including only the important variables  
impvars <- c("iyear","extended","success","suicide","attacktype1","targtype1","targsubtype1",   
 "natlty1","nperps","nperpcap","claimed","compclaim","weaptype1","weapsubtype1",  
 "nkill","nkillter","nwound","nwoundte","property","propextent","ishostkid","ransom","hostkidoutcome",  
 "INT\_LOG","INT\_IDEO","INT\_ANY")  
  
table(GTD\_SA$gname)  
  
y <- GTD\_SA$gname  
x <- GTD\_SA[,names(GTD\_SA) %in% impvars]  
  
x.dummy <- dummyVars(~.,data = x)  
  
x<- as.data.frame(predict(x.dummy, x))  
  
set.seed(199)  
inTrain <- createDataPartition(GTD\_SA$gname,p=.7, list=F)  
  
GTD.x.train <- x[inTrain,]  
  
str(GTD.x.train)  
  
GTD.y.train <- y[inTrain]  
str(GTD.y.train)  
  
GTD.training <- cbind(GTD.x.train, GTD.y.train)  
  
GTD.y.test<- y[-inTrain]  
GTD.x.test <- x[-inTrain,]  
  
ctrl <- trainControl(method = "cv", number=10, summaryFunction=multiClassSummary,  
 classProbs=T, allowParallel = FALSE)  
  
  
#to see what parameters are to be tuned:  
set.seed(199)  
  
class(GTD\_SA$gname)  
  
  
############\_\_\_\_\_\_\_\_\_\_\_\_Random Forrest\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_###################  
#Random Forest  
set.seed(199)  
  
m.rf <- train(y= GTD.y.train, x= GTD.x.train,  
 trControl = ctrl,  
 method="rf",   
 metric="logLoss") #tuneLength=15, #mtry= floor(mtry.val), tuneGrid = data.frame(mtry = c(floor(mtry.val))))  
# ntree = 100)  
  
m.rf  
  
saveRDS(m.rf, "Randomforest\_model.rds")  
  
getTrainPerf(m.rf)  
impvars.rf<- varImp(m.rf)  
saveRDS(impvars.rf, "Imp Variables for random forest model.rds")  
  
#can plot the performance of different parameters affect on ROC  
plot(m.rf)  
  
#the best performing model trained on the full training set is saved   
##preprocessing using predict function with caret train object will be applied to new data  
p.rf <- predict(m.rf,GTD.x.test)  
cm.rf <- confusionMatrix(p.rf,GTD.y.test) #calc accuracies with confuction matrix on test set  
  
#confusion Matrix  
cm.rf  
  
  
#Table of predicted against actual values  
table(p.rf, GTD.y.test) #returns the confusion matrix  
  
  
df <- as.data.frame(table(p.rf, GTD.y.test))  
  
library(ggplot2)  
ggplot(data = df, mapping = aes(x = GTD.y.test, y = p.rf)) +  
 geom\_tile(aes(fill = df$Freq), colour = "white") +  
 geom\_text(aes(label = sprintf("%1.0f", df$Freq)), vjust = 1) +  
 scale\_fill\_gradient(low = "grey", high = "orange") +  
 theme\_bw() + theme(legend.position = "none")  
  
  
  
############\_\_\_\_\_\_\_\_\_\_\_\_DECISION TREE\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_###################  
  
m.rpart <- train(y=GTD.y.train, x=GTD.x.train,  
 trControl = ctrl,  
 tuneLength=15,  
 metric = "logLoss", #using AUC to find best performing parameters  
 method = "rpart")  
m.rpart  
getTrainPerf(m.rpart)  
varImp(m.rpart)  
  
#can plot the performance of different parameters affect on ROC  
plot(m.rpart)  
  
#the best performing model trained on the full training set is saved   
##preprocessing using predict function with caret train object will be applied to new data  
p.rpart <- predict(m.rpart,GTD.x.test)  
cm.rpart <- confusionMatrix(p.rpart,GTD.y.test) #calc accuracies with confuction matrix on test set  
  
#confusion Matrix  
cm.rpart  
  
#Table of predicted against actual values  
table(p.rpart, GTD.y.test) #returns the confusion matrix  
  
df.rpart <- as.data.frame(table(p.rpart, GTD.y.test))  
  
library(ggplot2)  
ggplot(data = df.rpart, mapping = aes(x = GTD.y.test, y = p.rpart)) +  
 geom\_tile(aes(fill = df.rpart$Freq), colour = "white") +  
 geom\_text(aes(label = sprintf("%1.0f", df.rpart$Freq)), vjust = 1) +  
 scale\_fill\_gradient(low = "grey", high = "red") +  
 theme\_bw() + theme(legend.position = "none")  
  
############\_\_\_\_\_\_\_\_\_\_\_\_NAIVE BAYES\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_###################  
#naive Bayes  
set.seed(199)  
  
class(GTD\_SA$gname)  
  
grid <- data.frame(fL = c(0,0.5,1), usekernel = TRUE, adjust = c(0,0.5,1))  
m.nb <- train(y=GTD.y.train, x=GTD.x.train,  
 trControl = ctrl,  
 tuneGrid = grid,  
 metric = "logLoss", #using AUC to find best performing parameters  
 method = "nb",  
 importance = TRUE)  
m.nb  
getTrainPerf(m.nb)  
varImp(m.nb)  
  
  
#can plot the performance of different parameters affect on ROC  
plot(m.nb)  
  
#the best performing model trained on the full training set is saved   
##preprocessing using predict function with caret train object will be applied to new data  
p.nb <- predict(m.nb,GTD.x.test)  
cm.nb <- confusionMatrix(p.nb,GTD.y.test) #calc accuracies with confuction matrix on test set  
  
#confusion Matrix  
cm.nb  
  
#Table of predicted against actual values  
table(p.nb, GTD.y.test) #returns the confusion matrix  
  
df.nb <- as.data.frame(table(p.nb, GTD.y.test))  
  
library(ggplot2)  
ggplot(data = df.nb, mapping = aes(x = GTD.y.test, y = p.nb)) +  
 geom\_tile(aes(fill = df.nb$Freq), colour = "white") +  
 geom\_text(aes(label = sprintf("%1.0f", df.nb$Freq)), vjust = 1) +  
 scale\_fill\_gradient(low = "grey", high = "green") +  
 theme\_bw() + theme(legend.position = "none")  
  
  
############\_\_\_\_\_\_\_\_\_\_\_\_NEURAL NETWORKS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_###################  
set.seed(199)  
  
m.nn <- train(y=GTD.y.train, x=GTD.x.train,  
 trControl = ctrl,  
 preProc = c("scale"),  
 metric = "logLoss", #using AUC to find best performing parameters  
 method = "nnet")  
m.nn  
getTrainPerf(m.nn)  
  
#only slightly better, but better none the less :)  
plot(m.nn)  
getTrainPerf(m.nn)  
varImp(m.nn)  
  
  
#the best performing model trained on the full training set is saved   
##preprocessing using predict function with caret train object will be applied to new data  
p.nn <- predict(m.nn,GTD.x.test)  
cm.nn <- confusionMatrix(p.nn,GTD.y.test) #calc accuracies with confuction matrix on test set  
  
#confusion matrix  
cm.nn  
  
#Table of predicted against actual values  
table(p.nn, GTD.y.test) #returns the confusion matrix  
  
df.nn <- as.data.frame(table(p.nn, GTD.y.test))  
  
library(ggplot2)  
ggplot(data = df.nn, mapping = aes(x = GTD.y.test, y = p.nn)) +  
 geom\_tile(aes(fill = df.nn$Freq), colour = "white") +  
 geom\_text(aes(label = sprintf("%1.0f", df.nb$Freq)), vjust = 1) +  
 scale\_fill\_gradient(low = "grey", high = "yellow") +  
 theme\_bw() + theme(legend.position = "none")  
  
############\_\_\_\_\_\_\_\_\_\_\_\_BAGGING TREE\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_###################  
  
modelLookup("treebag") #we have some paramters to tune such as laplace correction  
set.seed(192)  
m.bag <- train(y=GTD.y.train, x=GTD.x.train,  
 trControl = ctrl,  
 metric = "logLoss", #using AUC to find best performing parameters  
 method = "treebag")  
m.bag  
  
getTrainPerf(m.bag)  
  
varImp(m.bag)  
p.bag<- predict(m.bag,GTD.x.test)  
  
cm.bag <- confusionMatrix(p.bag,GTD.y.test) #calc accuracies with confuction matrix on test set  
  
#confusion matrix  
cm.bag  
  
#Table of predicted against actual values  
table(p.bag, GTD.y.test) #returns the confusion matrix  
  
df.bag <- as.data.frame(table(p.bag, GTD.y.test))  
  
library(ggplot2)  
ggplot(data = df.bag, mapping = aes(x = GTD.y.test, y = p.bag)) +  
 geom\_tile(aes(fill = df.bag$Freq), colour = "white") +  
 geom\_text(aes(label = sprintf("%1.0f", df.nb$Freq)), vjust = 1) +  
 scale\_fill\_gradient(low = "grey", high = "purple") +  
 theme\_bw() + theme(legend.position = "none")  
  
  
############\_\_\_\_\_\_\_\_\_\_\_\_BOOSTING\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_###################  
  
install.packages("maboost")  
  
modelLookup("maboost") #we have some paramters to tune such as laplace correction  
set.seed(192)  
m.boost <- train(y=GTD.y.train, x=GTD.x.train,  
 trControl = ctrl,  
 metric = "logLoss", #using AUC to find best performing parameters  
 method = "gbm")  
m.boost  
getTrainPerf(m.boost)  
  
varImp(m.boost)  
p.boost<- predict(m.boost,GTD.x.test)  
cm.boost <- confusionMatrix(p.boost,GTD.y.test) #calc accuracies with confuction matrix on test set  
  
#confusion matrix  
cm.boost  
  
#Table of predicted against actual values  
table(p.boost, GTD.y.test) #returns the confusion matrix  
  
df.boost <- as.data.frame(table(p.boost, GTD.y.test))  
  
library(ggplot2)  
ggplot(data = df.boost, mapping = aes(x = GTD.y.test, y = p.boost)) +  
 geom\_tile(aes(fill = df.boost$Freq), colour = "white") +  
 geom\_text(aes(label = sprintf("%1.0f", df.boost$Freq)), vjust = 1) +  
 scale\_fill\_gradient(low = "grey", high = "dark green") +  
 theme\_bw() + theme(legend.position = "none")  
  
############\_\_\_\_\_\_\_\_\_\_\_\_SVM\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_###################  
  
modelLookup("svmRadial") #we have some paramters to tune such as laplace correction  
set.seed(192)  
m.svm <- train(y=GTD.y.train, x=GTD.x.train,  
 trControl = ctrl,  
 preProc = c("scale"), #scale variables  
 metric = "logLoss", #using AUC to find best performing parameters  
 method = "svmLinear")  
m.svm  
getTrainPerf(m.svm)  
  
varImp(m.svm)  
p.svm<- predict(m.svm,GTD.x.test)  
cm.svm <- confusionMatrix(p.svm,GTD.y.test) #calc accuracies with confuction matrix on test set  
  
#confusion matrix  
cm.svm  
  
#Table of predicted against actual values  
table(p.svm, GTD.y.test) #returns the confusion matrix  
  
df.svm <- as.data.frame(table(p.svm, GTD.y.test))  
  
library(ggplot2)  
ggplot(data = df.svm, mapping = aes(x = GTD.y.test, y = p.svm)) +  
 geom\_tile(aes(fill = df.svm$Freq), colour = "white") +  
 geom\_text(aes(label = sprintf("%1.0f", df.nb$Freq)), vjust = 1) +  
 scale\_fill\_gradient(low = "grey", high = "pink") +  
 theme\_bw() + theme(legend.position = "none")  
  
#lets compare all resampling approaches  
GTD.models <- list("Neural-Network"=m.nn,  
 "Naive-Bayes"=m.nb,"DecisionTree" = m.rpart,   
 "BaggingTree" = m.bag,"BoostingTree" = m.boost,  
 "Support Vector Machine"= m.svm, "Random Forest" = m.rf)  
  
GTD.models  
  
GTD.resamples = resamples(GTD.models)  
  
#plot performance comparisons  
bwplot(GTD.resamples, metric="Accuracy")  
   
bwplot(GTD.resamples, metric="Mean\_Sensitivity") #predicting default dependant on threshold  
bwplot(GTD.resamples, metric="Mean\_Specificity")  
  
bwplot(GTD.resamples, metric="logLoss")