Seattle Police Crime Markdown

Eren Uslu | usle00@vse.cz

Overview & Description of dataset

This data represents crime reported to the Seattle Police Department (SPD). Each row contains the record of a unique event where at least one criminal offense was reported by a member of the community or detected by an officer in the field. The Dataset is used in meetings for strategic planning, accountability and performance managment. This updated process includes all records of crime reports logged in the Departments Records Management System (RMS) since 2008, which are tracked as part of the SeaStat process. Records are evolved daily and are continually refreshed.

The dataset was provided by OpenML. The author is the City of Seattle.

• For more information visit: https://data.seattle.gov/Public-Safety/Crime-Data/4fs7-3vj5

• Access to the OpenML project: https://www.openml.org/d/41960

Problem statement & Goal

The goal is to build a ML model that will predict the accuracy for an Primary Offense by a given of all other feature variables (e.g. Neighborhood, Beat, etc.)

• **Target** is Primary Offense Description

• **Features** are the other remaining 8 variables

- Report Number (1)
- Occured Time (2)
- Reported Time (3)
- Crime Subcategory (4)
- Precinct (5)
- Sector (6)
- Beat (7)
- Neighborhood (8)

Structure

The structure and steps are similar to the one of the Python notebook. Tips and suggestion of the received review report are included in this project.

```
install.packages("corrplot", repos = "http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/erenu/Documents/R/win-library/3.6'
## (as 'lib' is unspecified)
```

```
## package 'corrplot' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\erenu\AppData\Local\Temp\RtmpqQ4iTH\downloaded_packages
```

```
install.packages("Hmisc", repos = "http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/erenu/Documents/R/win-library/3.6'
## (as 'lib' is unspecified)
```

```
## package 'Hmisc' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\erenu\AppData\Local\Temp\RtmpqQ4iTH\downloaded_packages
```

```
install.packages("corrgram", repos ="http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/erenu/Documents/R/win-library/3.6'
## (as 'lib' is unspecified)
```

```
## package 'corrgram' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\erenu\AppData\Local\Temp\RtmpqQ4iTH\downloaded_packages
```

Setup & Prerequisites

In order to to build the model and to the classification all necessary R packages have to be installed & loaded. **Note:** If one of the packaged don't load use require(package)

```
# Setup and loading needful & necessary libraries
library(caret)
library(dplyr)
library(e1071)
library(Hmisc)
library(ggplot2)
library(corrplot)
library(corrgram)
library(knitr)
library(MASS)
library(ranger)
library(tidyverse)
```

Exploratory data analysis

Importing the dataset and displaying the data with its structure to perform the analysis and classification task

Loading Seattle Police Crime Dataset

The Seattle Police Crime Dataset is been loaded and saved into a variable. The dataset is been directly imported from the OpenML website with its provided .arff file

```
crime_data <- read.csv("https://www.openml.org/data/get_csv/21379024/Seattle_Crime_Data_06-23-2019-4.arff")

# If any problem is occuring during the loading process, you have the option the load the dataset from the p
rovided excel sheet. Therefore just switch use your active working directoy and read the .csv file --> setwd
() --> crime_data <- read.csv(file = "Seattle_Crime_Data_06-23-2019-4.csv")

head(crime_data, n=10) # head displays the first 10 rows of the dataset
```

```
##    Report_Number Occurred_Time Reported_Time     Crime_Subcategory
## 1      1.975e+12           900          1500  BURGLARY-RESIDENTIAL
## 2      1.976e+12             1          2359   'SEX OFFENSE-OTHER'
## 3      1.979e+12          1600          1430           'CAR PROWL'
## 4      1.981e+13          2029          2030              HOMICIDE
## 5      1.981e+12          2000           435  BURGLARY-RESIDENTIAL
## 6      1.988e+13           155           155 'MOTOR VEHICLE THEFT'
## 7      1.993e+13          2213          2213              HOMICIDE
## 8      1.994e+13             0           844     'THEFT-ALL OTHER'
## 9      1.996e+13          1130          1700           'CAR PROWL'
## 10     1.999e+13             ?             ?        THEFT-SHOPLIFT
##     Primary_Offense_Description  Precinct Sector Beat
## 1            BURGLARY-FORCE-RES     SOUTH      R   R3
## 2   'SEXOFF-INDECENT LIBERTIES'   UNKNOWN      ?    ?
## 3               THEFT-CARPROWL      EAST      G   G2
## 4  HOMICIDE-PREMEDITATED-WEAPON     SOUTH      S   S2
## 5            BURGLARY-FORCE-RES SOUTHWEST      W   W3
## 6               VEH-THEFT-AUTO      WEST      M   M2
## 7     HOMICIDE-PREMEDITATED-GUN     SOUTH      R   R2
## 8                    THEFT-OTH SOUTHWEST      F   F1
## 9               THEFT-CARPROWL     SOUTH      O   O1
## 10              THEFT-SHOPLIFT   UNKNOWN      ?    ?
##                    Neighborhood
## 1          'LAKEWOOD/SEWARD PARK'
## 2                         UNKNOWN
## 3       'CENTRAL AREA/SQUIRE PARK'
## 4                 BRIGHTON/DUNLAP
## 5  'ROXHILL/WESTWOOD/ARBOR HEIGHTS'
## 6                     SLU/CASCADE
## 7        'CLAREMONT/RAINIER VISTA'
## 8                    'HIGH POINT'
## 9                            SODO
## 10                        UNKNOWN
```

Each row is a **unique crime report record column in the dataset**

• **Report Number** = Unique ID

• **Occured Time** = Time the offense occured on the 24 hour clokc

• **Reported Time** = Time the offense has been reported on the 24 hour clock

• **Crime Subcategory** = More detailed description of the Primary Offense

• **Primary Offense** = Description of the occured offense

• **Precinct** = District where offense occured

• **Sector** = Sector of the district

• **Beats** = Granular unit of management for patrol deployment

• **Neighborhood** = Location of the occured offense

Note: The Dataset doesn't provide any dates

```
str(crime_data) # Display structure and shape of the dataset
```

```
## 'data.frame':    523590 obs. of  9 variables:
##  $ Report_Number              : num  1.98e+12 1.98e+12 1.98e+12 1.98e+13 1.98e+12 ...
##  $ Occurred_Time              : Factor w/ 1441 levels "?","0","1","10",..: 1382 3 431 708 677 415 833 2
110 1 ...
##  $ Reported_Time              : Factor w/ 1441 levels "?","0","1","10",..: 361 954 323 710 1096 415 833 1
365 492 1 ...
##  $ Crime_Subcategory          : Factor w/ 31 levels "'AGGRAVATED ASSAULT'",..: 15 10 5 18 15 9 18 11 5 29
...
##  $ Primary_Offense_Description: Factor w/ 144 levels "'ADULT-VULNERABLE-PHYSICAL ABUSE'",..: 40 16 127 60
40 135 59 130 127 133 ...
##  $ Precinct                   : Factor w/ 7 levels "?","EAST","NORTH",..: 4 6 2 4 5 7 4 5 4 6 ...
##  $ Sector                     : Factor w/ 24 levels "?","6804","9512",..: 18 1 9 19 23 14 18 8 16 1 ...
##  $ Beat                       : Factor w/ 65 levels "?","B1","B2",..: 51 1 21 54 63 37 50 17 43 1 ...
##  $ Neighborhood               : Factor w/ 59 levels "'ALASKA JUNCTION'",..: 20 58 5 43 36 55 7 16 56 58 .
..
```

• 523590 = Total number of data

• 9 = Number of columns (1 target and 8 feature variables)

The head of the dataset doesn't show any **NaN (Not a Number) or NA (Not available)** values. Although it seems like no data is missing from the columns, it is clear from the first few rows that there are values corresponding to missing **'?' and 'UNKNOWN'**

Before converting the missing values **'?' and 'UNKNOWN'** lets check if there is any *NA* value at all in the dataset.

```
colSums(is.na(crime_data)) # Cumulate the sum of NA values for each of the nine columns
```

```
##             Report_Number             Occurred_Time
##                         0                         0
##             Reported_Time         Crime_Subcategory
##                         0                         0
## Primary_Offense_Description                  Precinct
##                         0                         0
##                    Sector                      Beat
##                         0                         0
##              Neighborhood
##                         0
```

The resuls show that no column has **NA** values

Preprocessing

Data preperation - Handling missing data

Before using the dataset for classification purposes missing data has to be handled and either removed or replaced.

Since no **NA** values where found let's see how many missing values in form of **?** and **UNKNOWN** occur in the dataset

```
# Count the number of '?' and 'UNKNOWN' entries with the sum() operation
sum(length(which(crime_data == "?")), length(which(crime_data == "UNKNOWN")))
```

```
## [1] 13628
```

13628 values where found which have **? or UNKNOWN** entries

In order to treat mis sing data properly, the missing values have to replaced using **NA** to find and handle missing values faster

```
crime_data[crime_data == "?"] <- NA # All '?' values are being replaced with 'NA'
crime_data[crime_data == "UNKNOWN"] <- NA # All 'UNKNOWN' are being replaced with 'NA'
```

Now check the number of **NA** values for each column again

```
colSums(is.na(crime_data)) # Cumulate the sum of NA values for each of the nine columns
```

```
##             Report_Number             Occurred_Time
##                         0                         2
##             Reported_Time         Crime_Subcategory
##                         2                       262
## Primary_Offense_Description                  Precinct
##                         0                      3352
##                    Sector                      Beat
##                      3346                      3298
##              Neighborhood
##                      3366
```

It is clearly visible that the column **Sector, Crime Subcategory, and Beat** have to the most missing values while the others have none or just a few

Finally, the missing **NA** values are being dropped. Since the amount of dropped values is not critical and replacing/inmuting the **NA** values makes no sense due to reason of having multiple, labeled entries for each column the data handling for missing values ends here.

```
crime_data <- na.omit(crime_data) # Rows with 'NA' values are being dropped
```

```
colSums(is.na(crime_data)) # Cumulate the sum of NA values for each of the nine columns
```

```
##                 Report_Number                  Occurred_Time
##                             0                              0
##                 Reported_Time             Crime_Subcategory
##                             0                              0
## Primary_Offense_Description                        Precinct
##                             0                              0
##                        Sector                           Beat
##                             0                              0
##                  Neighborhood
##                             0
```

```
sum(length(which(crime_data == "?")), length(which(crime_data == "UNKNOWN")), is.na(crime_data)) # Cumulate the number of missing values for '?', 'UNKNOWN', and 'NA'
```

```
## [1] 0
```

Data preperation - Label sizing

Data transformation

Since the dataset has more than 500.000 entries it more convenient and beneficial to transform and group the entries from the target variable **Primary Description**

Firstly, the number of unique value needs to be computated

```
length(unique(crime_data$Primary_Offense_Description)) # Computate the number of unique values
```

```
## [1] 142
```

In total the target variable has 142 unique values

Create of new column **Response Time**

To better understand the dataset we decided to create a new column "Response time". This column should have demonstrated the time between the time of crime and the time of the reporting of that crime. Unfortunately, the dataset does not contain any dates. So we don't know whether the crime was reported on the same day or maybe a day after. Therefore sometimes negative Response Times are being displayed.

```
# Create new Column 'Response Time' by subtracting 'Reportied Time - Occured Time'
crime_data$Response_Time <- as.numeric(as.character(crime_data$Occurred_Time)) - as.numeric(as.character(crime_data$Reported_Time))
#  Note: Negative Values appear due to the lack of dates
```

```
head(crime_data, n=10) # display first ten rows with new column
```

```
##    Report_Number Occurred_Time Reported_Time      Crime_Subcategory
## 1      1.975e+12           900          1500   BURGLARY-RESIDENTIAL
## 3      1.979e+12          1600          1430            'CAR PROWL'
## 4      1.981e+13          2029          2030               HOMICIDE
## 5      1.981e+12          2000           435   BURGLARY-RESIDENTIAL
## 6      1.988e+13           155           155 'MOTOR VEHICLE THEFT'
## 7      1.993e+13          2213          2213               HOMICIDE
## 8      1.994e+13             0           844      'THEFT-ALL OTHER'
## 9      1.996e+13          1130          1700            'CAR PROWL'
## 11     2.000e+13          2330          1055            'CAR PROWL'
## 12     2.001e+12          2310          2310               HOMICIDE
##     Primary_Offense_Description  Precinct Sector Beat
## 1             BURGLARY-FORCE-RES     SOUTH      R   R3
## 3                 THEFT-CARPROWL      EAST      G   G2
## 4  HOMICIDE-PREMEDITATED-WEAPON     SOUTH      S   S2
## 5             BURGLARY-FORCE-RES SOUTHWEST      W   W3
## 6                 VEH-THEFT-AUTO      WEST      M   M2
## 7     HOMICIDE-PREMEDITATED-GUN     SOUTH      R   R2
## 8                      THEFT-OTH SOUTHWEST      F   F1
## 9                 THEFT-CARPROWL     SOUTH      O   O1
## 11                THEFT-CARPROWL      WEST      Q   Q3
## 12    HOMICIDE-PREMEDITATED-GUN      WEST      K   K3
##                       Neighborhood Response_Time
## 1          'LAKEWOOD/SEWARD PARK'          -600
## 3      'CENTRAL AREA/SQUIRE PARK'           170
## 4               BRIGHTON/DUNLAP            -1
## 5  'ROXHILL/WESTWOOD/ARBOR HEIGHTS'          1565
## 6                   SLU/CASCADE             0
## 7      'CLAREMONT/RAINIER VISTA'             0
## 8                  'HIGH POINT'          -844
## 9                          SODO          -570
## 11                  SLU/CASCADE          1275
## 12         'DOWNTOWN COMMERCIAL'             0
```

Sub-categories/Group Primary Offense

Before grouping the target variable **Primary Offense** into the three sub categories **THEFT, DRUGS AND GUNS, AND OTHERS** a subset is being created

```
# Create subset and replace 'Primary Offense Description' label with 'Sub Crime'
crime_data_sub = crime_data
crime_data_sub["Sub_Crime"] = crime_data["Primary_Offense_Description"]
```

```
crime_data_sub$Sub_Crime <- sub(".*THEFT.+|.*BURGLARY.+|.*ROBBERY.+", "THEFT", crime_data_sub$Sub_Crime) # C
reate first subcategory 'THEFT'
crime_data_sub$Sub_Crime <- sub(".*THEFT.+|.*NARC.+|.*DRUG.+,|.*WEAPON.+", "DRUGS_AND_GUNS", crime_data_sub$
Sub_Crime) # Create second subcategory 'DRUGS AND GUNS'
crime_data_sub$Sub_Crime[!grepl("THEFT|DRUGS_AND_GUNS", crime_data_sub$Sub_Crime)] <- "OTHERS" # Create thir
d subcategory 'OTHERS'
```

The 142 unique values from the Primary Offense were grouped in the three sub-caregories **THEFT, DRUGS AND GUNS, and OTHERS**.

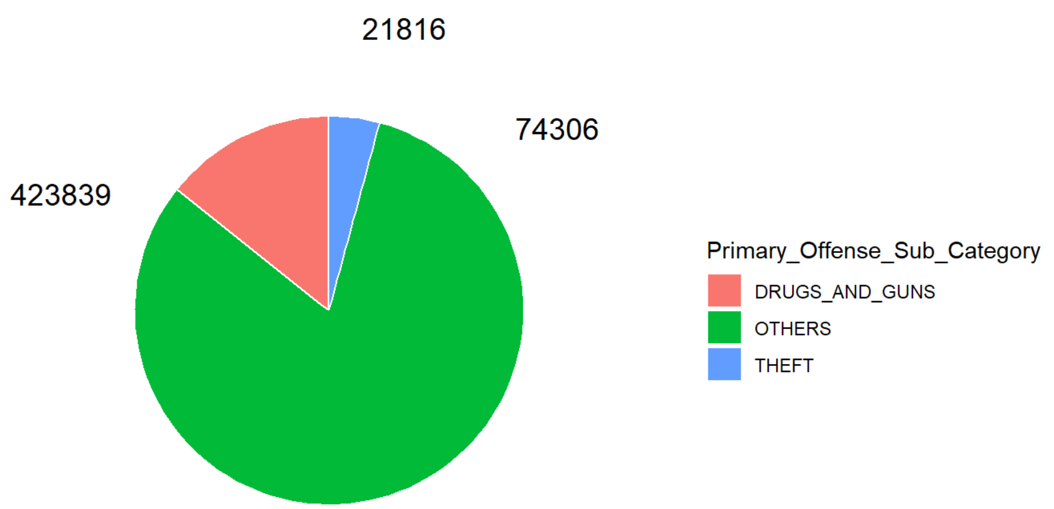
The frequency of each sub-category is being computated.

```
sub_crime_category <- as.data.frame(table(crime_data_sub$Sub_Crime))
sub_crime_category <- data.frame(c("THEFT","DRUGS_AND_GUNS","OTHERS"), sub_crime_category$Freq)
colnames(sub_crime_category) <-c("Primary_Offense_Sub_Category", "Frequency")
sub_crime_category
```

```
##   Primary_Offense_Sub_Category Frequency
## 1                        THEFT     21816
## 2               DRUGS_AND_GUNS     74306
## 3                       OTHERS    423839
```

The frequency of the sub-categories are being visualized

```r
ggplot(sub_crime_category, aes(x="Primary Offense Sub Category", y=Frequency, fill=Primary_Offense_Sub_Categ
ory)) +
  geom_bar(stat="identity", width=1, color="white") +
  coord_polar("y", start=0) +
  theme_void() +
  geom_text(aes(label = Frequency, x =2), color="black", size=5)
```



Not all features are necessary for the classification task. Not needed columns are being dropped

```r
crime_class <- crime_data_sub[c("Occurred_Time","Crime_Subcategory", "Precinct", "Sector", "Beat", "Neighbor
hood", "Sub_Crime")]
head(crime_class, n=10)
```

```
##    Occurred_Time      Crime_Subcategory  Precinct Sector Beat
## 1            900   BURGLARY-RESIDENTIAL     SOUTH      R   R3
## 3           1600            'CAR PROWL'      EAST      G   G2
## 4           2029               HOMICIDE     SOUTH      S   S2
## 5           2000   BURGLARY-RESIDENTIAL SOUTHWEST      W   W3
## 6            155 'MOTOR VEHICLE THEFT'       WEST      M   M2
## 7           2213               HOMICIDE     SOUTH      R   R2
## 8              0     'THEFT-ALL OTHER' SOUTHWEST      F   F1
## 9           1130            'CAR PROWL'     SOUTH      O   O1
## 11          2330            'CAR PROWL'      WEST      Q   Q3
## 12          2310               HOMICIDE      WEST      K   K3
##                          Neighborhood Sub_Crime
## 1             'LAKEWOOD/SEWARD PARK'     THEFT
## 3         'CENTRAL AREA/SQUIRE PARK'     THEFT
## 4                   BRIGHTON/DUNLAP    OTHERS
## 5  'ROXHILL/WESTWOOD/ARBOR HEIGHTS'     THEFT
## 6                       SLU/CASCADE     THEFT
## 7          'CLAREMONT/RAINIER VISTA'    OTHERS
## 8                      'HIGH POINT'     THEFT
## 9                              SODO     THEFT
## 11                      SLU/CASCADE     THEFT
## 12             'DOWNTOWN COMMERCIAL'    OTHERS
```

Label Encoding

Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. This is a better way on how those non numeric columns must be operated.

For the the classification task only float or integer data types are being accepted. Therefore, object data must be encode to integer datatypes.

Categorical text data is being converted into readable numeric data using Label Encoding class

```
crime_class$Occurred_Time<-as.numeric(as.factor(crime_class$Occurred_Time))
crime_class$Crime_Subcategory<-as.numeric(as.factor(crime_class$Crime_Subcategory))
crime_class$Precinct<-as.numeric(as.factor(crime_class$Precinct))
crime_class$Sector<-as.numeric(as.factor(crime_class$Sector))
crime_class$Beat<-as.numeric(as.factor(crime_class$Beat))
crime_class$Neighborhood<-as.numeric(as.factor(crime_class$Neighborhood))
crime_class$Sub_Crime<-as.factor(as.character(crime_class$Sub_Crime))
```

Random Sampling

Random sampling is a sampling technique where every item in the population has an even chance and likelihood of being selected in the sample. In this dataset this means running over more than 500.000 takes to much time and ressources. Sampling data reduces computation time and reduce the observed data size. **10% (50.000 rows)** of the data is randomly being sampled

```
crime_random_sample <- crime_class[sample(nrow(crime_class), 50000, replace = FALSE, prob=NULL),]
```

Data shuffling

Shuffling data serves the purpose of reducing variance and making sure that models remain general and overfit less. Data shuffling helps to improve ML model quality and improve the predictive performance

```
data_shuffle <- nrow(crime_random_sample)
p_rows <- sample(data_shuffle)
crime_shuffle <- crime_class[p_rows,]
```

Data splitting

Data splitting is the act of partitioning available data into. two portions, usually for cross-validatory purposes. One. portion of the data is used to develop a predictive model. and the other to evaluate the model’s performance.

The dataset is splitted into a train dataset with **75%** and a test dataset with **25%** of all sampled observations

```
# Model fitting
crime_split <- round(data_shuffle * 0.75)
crime_train <- crime_shuffle[1:crime_split,]
crime_test <- crime_shuffle[(crime_split+1):nrow(crime_shuffle),]
```

Model Building - Classification Task

A recommendation to use which algorithm can be found here: https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

1. KNN-Algorithm

First a seed needs to be set in order to make results replicable

```
set.seed(7)
```

- Train Control

trainControl is being set up to run the KNN-Algorithm

```
knn_train_control <- trainControl(method="repeatedcv", number = 10, repeats = 3)
```

- KNN Model Fitting

The model is being fit with the crime train data

```
knn_fit <- train(
  Sub_Crime ~.,
  data = crime_train,
  method = "knn",
  tuneLength = 20,
  trControl = knn_train_control)
```

- Faults prediction

The fault of the test dataset is being predicted

```
knn_prediction <- predict(knn_fit, newdata = crime_test)
```

- Confusion Matrix

Executing Confustion Matrix for KNN

```
confusionMatrix(knn_prediction, crime_test$Sub_Crime)
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction      DRUGS_AND_GUNS OTHERS THEFT
##   DRUGS_AND_GUNS            308     84   123
##   OTHERS                     56    121   162
##   THEFT                     508   1323  9815
##
## Overall Statistics
##
##                Accuracy : 0.8195
##                  95% CI : (0.8127, 0.8262)
##     No Information Rate : 0.808
##     P-Value [Acc > NIR] : 0.0005158
##
##                   Kappa : 0.2512
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: DRUGS_AND_GUNS Class: OTHERS Class: THEFT
## Sensitivity                        0.35321       0.07919       0.9718
## Specificity                        0.98220       0.98013       0.2371
## Pos Pred Value                     0.59806       0.35693       0.8428
## Neg Pred Value                     0.95294       0.88430       0.6663
## Prevalence                         0.06976       0.12224       0.8080
## Detection Rate                     0.02464       0.00968       0.7852
## Detection Prevalence               0.04120       0.02712       0.9317
## Balanced Accuracy                  0.66770       0.52966       0.6044
```

- Accuracy of KNN-Classification

The Accuracy for KNN is being computated

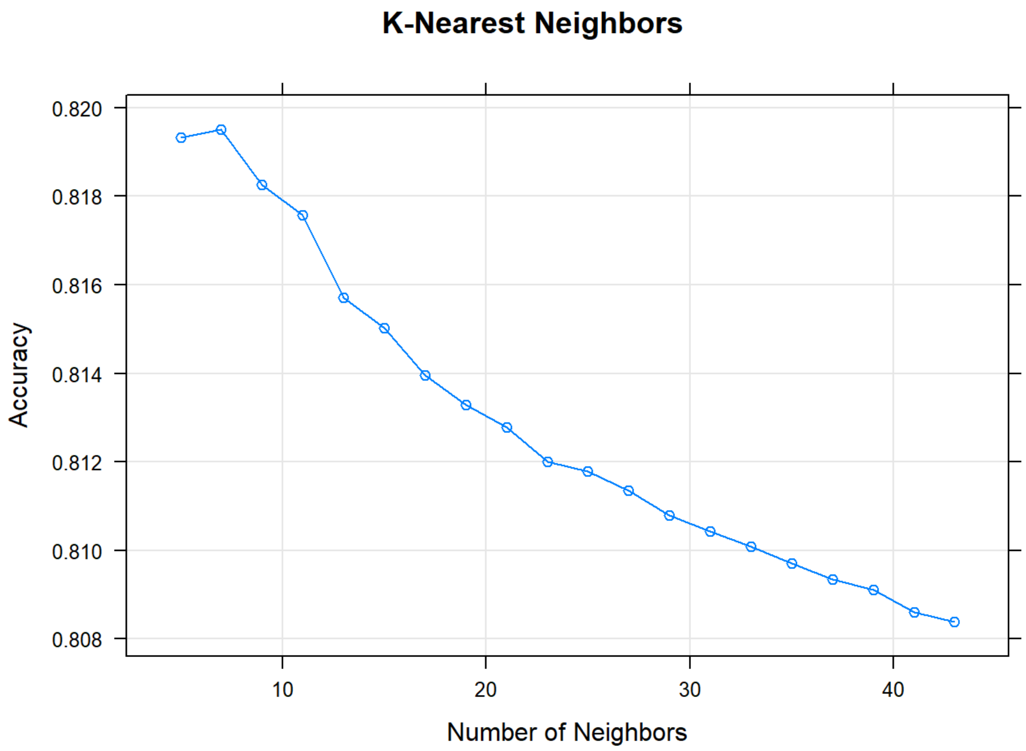
```
knn_accuracy <- mean(knn_prediction == crime_test$Sub_Crime)
knn_accuracy
```

```
## [1] 0.81952
```

- Accuracy plotting

The accuracy with a different number of neigbors is being plot. The variance of the accurancy is being displayed

```
plot(knn_fit, xlab = "Number of Neighbors", ylab = "Accuracy", main = "K-Nearest Neighbors")
```



** The higher the number of neighbors the slightly higher is the accuracy (positive correlation)**

2. Random Forest Classifier

First a seed needs to be set in order to make results replicable

```
set.seed(12)
```

- Random Forest model fitting

The Random model is being fit with the crime train data by using the 'ranger' method

```
rf_fit <- train(Sub_Crime ~ .,
                data = crime_train,
                method = "ranger")
```

```
## Growing trees.. Progress: 70%. Estimated remaining time: 3 minutes, 38 seconds.
```

- Faults prediction

The prediction of the fault is being predicted

```
rf_prediction <- predict(rf_fit, crime_test)
```

- Confusion Matrix

Executing the confusion matrix

```
confusionMatrix(rf_prediction, crime_test$Sub_Crime)
```

```
## Confusion Matrix and Statistics
##
##                  Reference
## Prediction       DRUGS_AND_GUNS OTHERS THEFT
##   DRUGS_AND_GUNS            872      0     0
##   OTHERS                      0   1528     0
##   THEFT                       0      0 10100
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.808
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: DRUGS_AND_GUNS Class: OTHERS Class: THEFT
## Sensitivity                        1.00000        1.0000        1.000
## Specificity                        1.00000        1.0000        1.000
## Pos Pred Value                     1.00000        1.0000        1.000
## Neg Pred Value                     1.00000        1.0000        1.000
## Prevalence                         0.06976        0.1222        0.808
## Detection Rate                     0.06976        0.1222        0.808
## Detection Prevalence               0.06976        0.1222        0.808
## Balanced Accuracy                  1.00000        1.0000        1.000
```

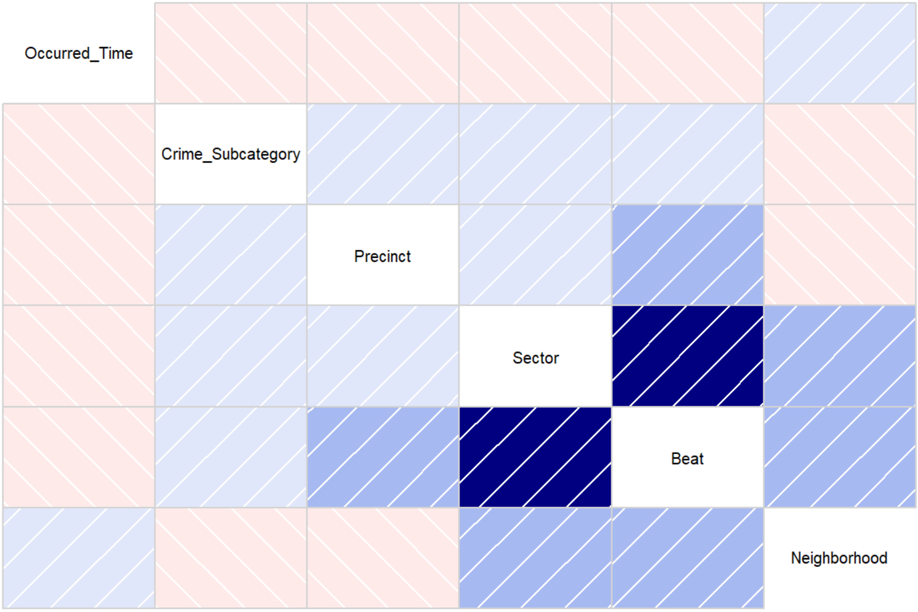
- Correlation Matrix with its plotting

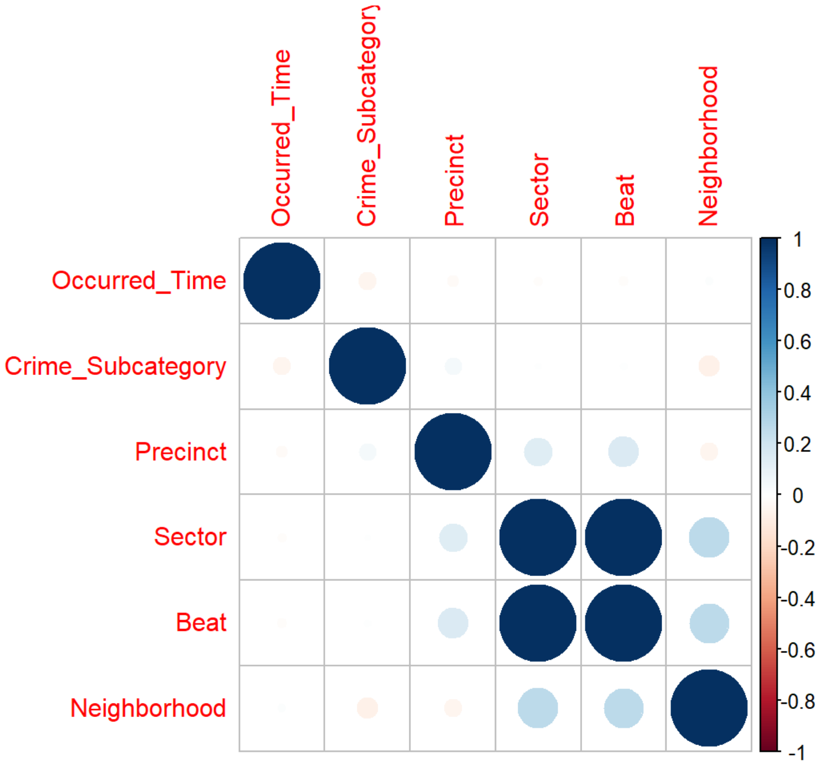
Executing the correlation matrix with rcorr and plotting the result to disply correlation between variables

```
corr_matrix <- crime_class[, sapply(crime_class, is.numeric)]
cor(corr_matrix, use ="complete.obs", method="pearson")
```

```
##                    Occurred_Time Crime_Subcategory    Precinct       Sector
## Occurred_Time         1.00000000      -0.053323027 -0.02191947 -0.014201714
## Crime_Subcategory    -0.05332303       1.000000000  0.04763651  0.004771488
## Precinct             -0.02191947       0.047636514  1.00000000  0.134757156
## Sector               -0.01420171       0.004771488  0.13475716  1.000000000
## Beat                 -0.01438178       0.009312654  0.15841369  0.995201682
## Neighborhood          0.01050857      -0.073000865 -0.05261363  0.269278658
##                           Beat Neighborhood
## Occurred_Time     -0.014381781   0.01050857
## Crime_Subcategory  0.009312654  -0.07300087
## Precinct           0.158413687  -0.05261363
## Sector             0.995201682   0.26927866
## Beat               1.000000000   0.26154329
## Neighborhood       0.261543289   1.00000000
```

```
corrplot(corrgram(corr_matrix, method="number"))
```





- Accuracy of Random Forest Classifier

The accuracy of the Random Forrest Classifier is being computated

```
rf_accuracy <- mean(rf_prediction == crime_test$Sub_Crime)
rf_accuracy
```

```
## [1] 1
```

Hyperparameter tuning

Before starting with the hyperparameter tuning the classifier with the highest score has to be selected. Therefore, the accuracy between the two classifiers **KNN and Random Forest** are being compared

```
comparison_accuracy <- data.frame(c("KNN", "Random Forest"), c(knn_accuracy, rf_accuracy))
comparison_accuracy
```

```
##   c..KNN....Random.Forest.. c.knn_accuracy..rf_accuracy.
## 1                       KNN                       0.81952
## 2             Random Forest                       1.00000
```

As a result the Random Forest classifier has an higher accuracy score. Thus, the Hyperparamater tuning is applied to this classifier.

For the Hyperparameter tuning Random Search and Grid Search CV have been selected

1. Random Search

*First a seed needs to be set in order to make results replicable

```
set.seed(70)
```

*Set trainControl

```
random_control <- trainControl(method='repeatedcv', number=5, repeats=2, search = 'random')
```

*Fit Random Ranger

10 random values of mtry at each time tuning

```
fit_random_ranger <- train(Sub_Crime ~ .,
                   data = crime_train,
                   method = 'ranger',
                   metric = 'Accuracy',
                   tuneLength  = 10,
                   trControl = random_control)
print(fit_random_ranger)
```

```
## Random Forest
##
## 37500 samples
##     6 predictor
##     3 classes: 'DRUGS_AND_GUNS', 'OTHERS', 'THEFT'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 2 times)
## Summary of sample sizes: 30002, 29999, 30000, 30000, 29999, 30000, ...
## Resampling results across tuning parameters:
##
##   min.node.size  mtry  splitrule   Accuracy   Kappa
##    1             3     extratrees  0.9986933  0.9959796
##    2             4     gini        0.9999600  0.9998773
##    4             3     gini        0.9989600  0.9968020
##    7             2     gini        0.9840933  0.9495510
##    8             5     gini        1.0000000  1.0000000
##   14             3     extratrees  0.9984267  0.9951568
##   15             1     gini        0.8474935  0.2992850
##   15             5     gini        1.0000000  1.0000000
##   17             3     extratrees  0.9980667  0.9940412
##   20             2     extratrees  0.9374932  0.7789889
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 5, splitrule = gini
##  and min.node.size = 8.
```

2. Grid Search CV

*First a seed needs to be set in order to make results replicable

```
set.seed(70)
```

- Definition of tune grid parameters

```
tune_grid <- expand.grid(mtry = c(2, 3, 4, 5), splitrule = c("extratrees", "gini"), min.node.size= c(1,2))
```

```
fit_tunegrid_ranger <- train(
  as.factor(Sub_Crime) ~.,
  tuneLength = 1,
  data = crime_train,
  method = "ranger",
  trControl = trainControl(
    method = "cv",
    number = 5,
    verboseIter = TRUE),
  tuneGrid = tune_grid
)
```

```
## + Fold1: mtry=2, splitrule=extratrees, min.node.size=1
## - Fold1: mtry=2, splitrule=extratrees, min.node.size=1
## + Fold1: mtry=3, splitrule=extratrees, min.node.size=1
## - Fold1: mtry=3, splitrule=extratrees, min.node.size=1
## + Fold1: mtry=4, splitrule=extratrees, min.node.size=1
## - Fold1: mtry=4, splitrule=extratrees, min.node.size=1
## + Fold1: mtry=5, splitrule=extratrees, min.node.size=1
## - Fold1: mtry=5, splitrule=extratrees, min.node.size=1
## + Fold1: mtry=2, splitrule=gini, min.node.size=1
## - Fold1: mtry=2, splitrule=gini, min.node.size=1
## + Fold1: mtry=3, splitrule=gini, min.node.size=1
## - Fold1: mtry=3, splitrule=gini, min.node.size=1
## + Fold1: mtry=4, splitrule=gini, min.node.size=1
## - Fold1: mtry=4, splitrule=gini, min.node.size=1
## + Fold1: mtry=5, splitrule=gini, min.node.size=1
## - Fold1: mtry=5, splitrule=gini, min.node.size=1
## + Fold1: mtry=2, splitrule=extratrees, min.node.size=2
## - Fold1: mtry=2, splitrule=extratrees, min.node.size=2
## + Fold1: mtry=3, splitrule=extratrees, min.node.size=2
## - Fold1: mtry=3, splitrule=extratrees, min.node.size=2
## + Fold1: mtry=4, splitrule=extratrees, min.node.size=2
## - Fold1: mtry=4, splitrule=extratrees, min.node.size=2
## + Fold1: mtry=5, splitrule=extratrees, min.node.size=2
## - Fold1: mtry=5, splitrule=extratrees, min.node.size=2
## + Fold1: mtry=2, splitrule=gini, min.node.size=2
## - Fold1: mtry=2, splitrule=gini, min.node.size=2
## + Fold1: mtry=3, splitrule=gini, min.node.size=2
## - Fold1: mtry=3, splitrule=gini, min.node.size=2
## + Fold1: mtry=4, splitrule=gini, min.node.size=2
## - Fold1: mtry=4, splitrule=gini, min.node.size=2
## + Fold1: mtry=5, splitrule=gini, min.node.size=2
## - Fold1: mtry=5, splitrule=gini, min.node.size=2
## + Fold2: mtry=2, splitrule=extratrees, min.node.size=1
## - Fold2: mtry=2, splitrule=extratrees, min.node.size=1
## + Fold2: mtry=3, splitrule=extratrees, min.node.size=1
## - Fold2: mtry=3, splitrule=extratrees, min.node.size=1
## + Fold2: mtry=4, splitrule=extratrees, min.node.size=1
## - Fold2: mtry=4, splitrule=extratrees, min.node.size=1
## + Fold2: mtry=5, splitrule=extratrees, min.node.size=1
## - Fold2: mtry=5, splitrule=extratrees, min.node.size=1
## + Fold2: mtry=2, splitrule=gini, min.node.size=1
## - Fold2: mtry=2, splitrule=gini, min.node.size=1
## + Fold2: mtry=3, splitrule=gini, min.node.size=1
## - Fold2: mtry=3, splitrule=gini, min.node.size=1
## + Fold2: mtry=4, splitrule=gini, min.node.size=1
## - Fold2: mtry=4, splitrule=gini, min.node.size=1
## + Fold2: mtry=5, splitrule=gini, min.node.size=1
## - Fold2: mtry=5, splitrule=gini, min.node.size=1
## + Fold2: mtry=2, splitrule=extratrees, min.node.size=2
## - Fold2: mtry=2, splitrule=extratrees, min.node.size=2
## + Fold2: mtry=3, splitrule=extratrees, min.node.size=2
## - Fold2: mtry=3, splitrule=extratrees, min.node.size=2
## + Fold2: mtry=4, splitrule=extratrees, min.node.size=2
## - Fold2: mtry=4, splitrule=extratrees, min.node.size=2
## + Fold2: mtry=5, splitrule=extratrees, min.node.size=2
## - Fold2: mtry=5, splitrule=extratrees, min.node.size=2
## + Fold2: mtry=2, splitrule=gini, min.node.size=2
## - Fold2: mtry=2, splitrule=gini, min.node.size=2
## + Fold2: mtry=3, splitrule=gini, min.node.size=2
## - Fold2: mtry=3, splitrule=gini, min.node.size=2
```

```
## - Fold2: mtry=3, splitrule=gini, min.node.size=2
## + Fold2: mtry=4, splitrule=gini, min.node.size=2
## - Fold2: mtry=4, splitrule=gini, min.node.size=2
## + Fold2: mtry=5, splitrule=gini, min.node.size=2
## - Fold2: mtry=5, splitrule=gini, min.node.size=2
## + Fold3: mtry=2, splitrule=extratrees, min.node.size=1
## - Fold3: mtry=2, splitrule=extratrees, min.node.size=1
## + Fold3: mtry=3, splitrule=extratrees, min.node.size=1
## - Fold3: mtry=3, splitrule=extratrees, min.node.size=1
## + Fold3: mtry=4, splitrule=extratrees, min.node.size=1
## - Fold3: mtry=4, splitrule=extratrees, min.node.size=1
## + Fold3: mtry=5, splitrule=extratrees, min.node.size=1
## - Fold3: mtry=5, splitrule=extratrees, min.node.size=1
## + Fold3: mtry=2, splitrule=gini, min.node.size=1
## - Fold3: mtry=2, splitrule=gini, min.node.size=1
## + Fold3: mtry=3, splitrule=gini, min.node.size=1
## - Fold3: mtry=3, splitrule=gini, min.node.size=1
## + Fold3: mtry=4, splitrule=gini, min.node.size=1
## - Fold3: mtry=4, splitrule=gini, min.node.size=1
## + Fold3: mtry=5, splitrule=gini, min.node.size=1
## - Fold3: mtry=5, splitrule=gini, min.node.size=1
## + Fold3: mtry=2, splitrule=extratrees, min.node.size=2
## - Fold3: mtry=2, splitrule=extratrees, min.node.size=2
## + Fold3: mtry=3, splitrule=extratrees, min.node.size=2
## - Fold3: mtry=3, splitrule=extratrees, min.node.size=2
## + Fold3: mtry=4, splitrule=extratrees, min.node.size=2
## - Fold3: mtry=4, splitrule=extratrees, min.node.size=2
## + Fold3: mtry=5, splitrule=extratrees, min.node.size=2
## - Fold3: mtry=5, splitrule=extratrees, min.node.size=2
## + Fold3: mtry=2, splitrule=gini, min.node.size=2
## - Fold3: mtry=2, splitrule=gini, min.node.size=2
## + Fold3: mtry=3, splitrule=gini, min.node.size=2
## - Fold3: mtry=3, splitrule=gini, min.node.size=2
## + Fold3: mtry=4, splitrule=gini, min.node.size=2
## - Fold3: mtry=4, splitrule=gini, min.node.size=2
## + Fold3: mtry=5, splitrule=gini, min.node.size=2
## - Fold3: mtry=5, splitrule=gini, min.node.size=2
## + Fold4: mtry=2, splitrule=extratrees, min.node.size=1
## - Fold4: mtry=2, splitrule=extratrees, min.node.size=1
## + Fold4: mtry=3, splitrule=extratrees, min.node.size=1
## - Fold4: mtry=3, splitrule=extratrees, min.node.size=1
## + Fold4: mtry=4, splitrule=extratrees, min.node.size=1
## - Fold4: mtry=4, splitrule=extratrees, min.node.size=1
## + Fold4: mtry=5, splitrule=extratrees, min.node.size=1
## - Fold4: mtry=5, splitrule=extratrees, min.node.size=1
## + Fold4: mtry=2, splitrule=gini, min.node.size=1
## - Fold4: mtry=2, splitrule=gini, min.node.size=1
## + Fold4: mtry=3, splitrule=gini, min.node.size=1
## - Fold4: mtry=3, splitrule=gini, min.node.size=1
## + Fold4: mtry=4, splitrule=gini, min.node.size=1
## - Fold4: mtry=4, splitrule=gini, min.node.size=1
## + Fold4: mtry=5, splitrule=gini, min.node.size=1
## - Fold4: mtry=5, splitrule=gini, min.node.size=1
## + Fold4: mtry=2, splitrule=extratrees, min.node.size=2
## - Fold4: mtry=2, splitrule=extratrees, min.node.size=2
## + Fold4: mtry=3, splitrule=extratrees, min.node.size=2
## - Fold4: mtry=3, splitrule=extratrees, min.node.size=2
## + Fold4: mtry=4, splitrule=extratrees, min.node.size=2
## - Fold4: mtry=4, splitrule=extratrees, min.node.size=2
## + Fold4: mtry=5, splitrule=extratrees, min.node.size=2
## - Fold4: mtry=5, splitrule=extratrees, min.node.size=2
## + Fold4: mtry=2, splitrule=gini, min.node.size=2
## - Fold4: mtry=2, splitrule=gini, min.node.size=2
## + Fold4: mtry=3, splitrule=gini, min.node.size=2
## - Fold4: mtry=3, splitrule=gini, min.node.size=2
## + Fold4: mtry=4, splitrule=gini, min.node.size=2
## - Fold4: mtry=4, splitrule=gini, min.node.size=2
## + Fold4: mtry=5, splitrule=gini, min.node.size=2
## - Fold4: mtry=5, splitrule=gini, min.node.size=2
## + Fold5: mtry=2, splitrule=extratrees, min.node.size=1
## - Fold5: mtry=2, splitrule=extratrees, min.node.size=1
## + Fold5: mtry=3, splitrule=extratrees, min.node.size=1
## - Fold5: mtry=3, splitrule=extratrees, min.node.size=1
```

```
## + Fold5: mtry=4, splitrule=extratrees, min.node.size=1
## - Fold5: mtry=4, splitrule=extratrees, min.node.size=1
## + Fold5: mtry=5, splitrule=extratrees, min.node.size=1
## - Fold5: mtry=5, splitrule=extratrees, min.node.size=1
## + Fold5: mtry=2, splitrule=gini, min.node.size=1
## - Fold5: mtry=2, splitrule=gini, min.node.size=1
## + Fold5: mtry=3, splitrule=gini, min.node.size=1
## - Fold5: mtry=3, splitrule=gini, min.node.size=1
## + Fold5: mtry=4, splitrule=gini, min.node.size=1
## - Fold5: mtry=4, splitrule=gini, min.node.size=1
## + Fold5: mtry=5, splitrule=gini, min.node.size=1
## - Fold5: mtry=5, splitrule=gini, min.node.size=1
## + Fold5: mtry=2, splitrule=extratrees, min.node.size=2
## - Fold5: mtry=2, splitrule=extratrees, min.node.size=2
## + Fold5: mtry=3, splitrule=extratrees, min.node.size=2
## - Fold5: mtry=3, splitrule=extratrees, min.node.size=2
## + Fold5: mtry=4, splitrule=extratrees, min.node.size=2
## - Fold5: mtry=4, splitrule=extratrees, min.node.size=2
## + Fold5: mtry=5, splitrule=extratrees, min.node.size=2
## - Fold5: mtry=5, splitrule=extratrees, min.node.size=2
## + Fold5: mtry=2, splitrule=gini, min.node.size=2
## - Fold5: mtry=2, splitrule=gini, min.node.size=2
## + Fold5: mtry=3, splitrule=gini, min.node.size=2
## - Fold5: mtry=3, splitrule=gini, min.node.size=2
## + Fold5: mtry=4, splitrule=gini, min.node.size=2
## - Fold5: mtry=4, splitrule=gini, min.node.size=2
## + Fold5: mtry=5, splitrule=gini, min.node.size=2
## - Fold5: mtry=5, splitrule=gini, min.node.size=2
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 5, splitrule = extratrees, min.node.size = 1 on full training set
```

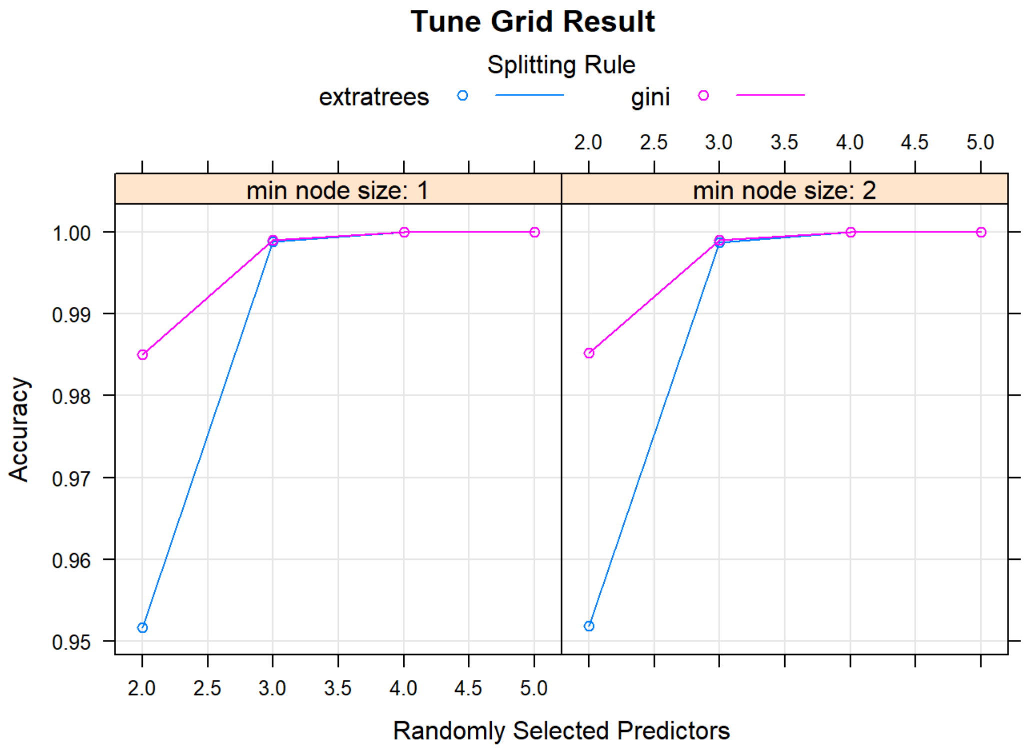
- Fit Ranger

```
fit_tunegrid_ranger
```

```
## Random Forest
##
## 37500 samples
##     6 predictor
##     3 classes: 'DRUGS_AND_GUNS', 'OTHERS', 'THEFT'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 30002, 29999, 30000, 30000, 29999
## Resampling results across tuning parameters:
##
##   mtry  splitrule   min.node.size  Accuracy   Kappa
##   2     extratrees  1              0.9517334  0.8351617
##   2     extratrees  2              0.9518395  0.8353651
##   2     gini        1              0.9849599  0.9524047
##   2     gini        2              0.9851732  0.9530752
##   3     extratrees  1              0.9988000  0.9963103
##   3     extratrees  2              0.9986667  0.9958980
##   3     gini        1              0.9990133  0.9969670
##   3     gini        2              0.9990133  0.9969671
##   4     extratrees  1              0.9999467  0.9998363
##   4     extratrees  2              0.9999467  0.9998363
##   4     gini        1              0.9999467  0.9998363
##   4     gini        2              0.9999467  0.9998364
##   5     extratrees  1              1.0000000  1.0000000
##   5     extratrees  2              1.0000000  1.0000000
##   5     gini        1              1.0000000  1.0000000
##   5     gini        2              1.0000000  1.0000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 5, splitrule =
##  extratrees and min.node.size = 1.
```

- Plotting Grid Search CV Results

```
plot(fit_tunegrid_ranger, xlab = "Randomly Selected Predictors", ylab = "Accuracy", main= "Tune Grid Result"
)
```



Conclusion and Model evaluation

I managed to perform the model and applied the necassary ML Algortihms. The Random Forest Classification was the fastest, most stable and most accurate, followed by the kNN-Neighbour Classifier.

KNN with an accuracy of almost 100% shows that there is a great dependency between the features due to overfitting

```
comparison_accuracy <- data.frame(c("KNN", "Random Forest"), c(knn_accuracy, rf_accuracy))
comparison_accuracy
```

```
##   c..KNN....Random.Forest.. c.knn_accuracy..rf_accuracy.
## 1                       KNN                      0.81952
## 2             Random Forest                      1.00000
```