Analysis of EEG Eye data with supervised machine learning methods

Darya Savitskaya

June 23, 2023

Abstract

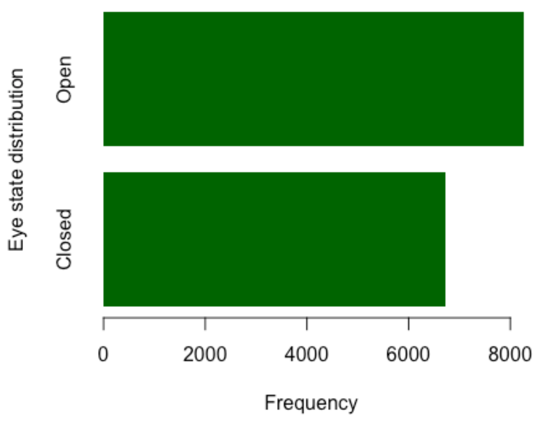
In this project, a number of supervised algorithms were applied to a data set containing EEG brain signal recordings with a binary dependent variable - eye state. The performance of three models (Logistic Regression, Random Forest and k-NN) was analysed with reference to the internal structure of the data. The lowest error rate was shown by a k-NN algorithm, which can be explained by the non-linearity of data and intrinsic complex relationships within. The worst performing variable was located by several models and was proposed to be dropped in future experiments.

1 Statement of the problem

1.1 Description of the dataset

The EEG Eye State Data Set is a collection of electroencephalogram (EEG) recordings designed to predict eye state (open or closed) based on brain wave signals. It was collected from a single subject performing a continuous visual task of keeping the eyes open or closed. The sample is 14,980 rows, each containing 14 attributes. The features correspond to 14 EEG measurements from the headset, originally labeled AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4, in that order.

Figure 1: The bar plot of the eye state distribution

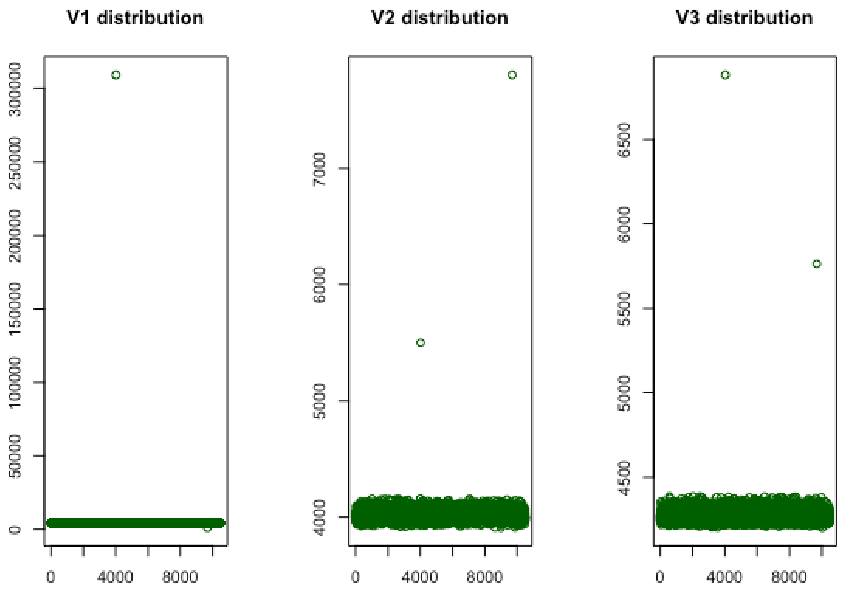


The distribution of eye states in the data set is relatively balanced, and a small majority are states of open eyes, as it can be seen in the bar plot.

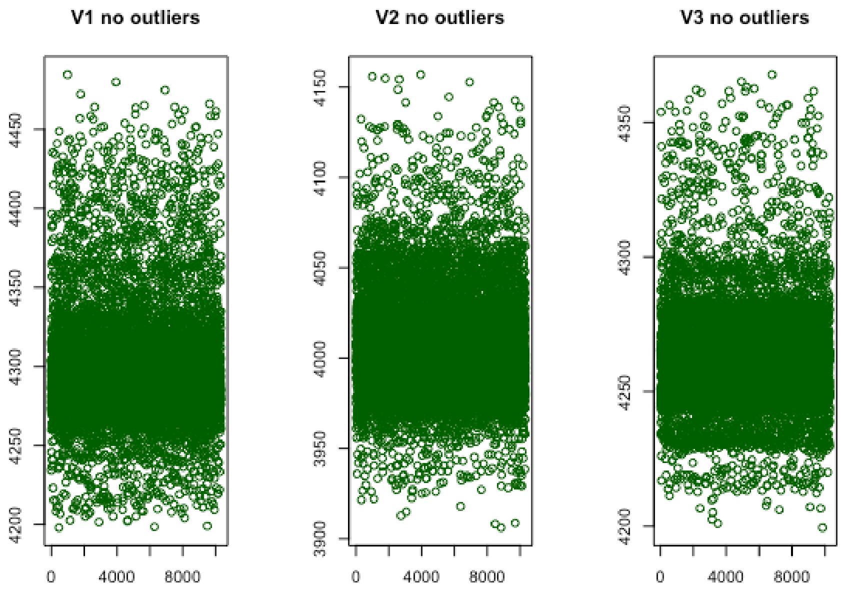
From the scatter plot visualisation of the variables, it can bee seen that there are some possible outliers. For example, see provided plots of the first three variables of the dataset in the Figure 2. To deal with the problem, I have calculated the z-score of every entry of every variable. Rows that contained at least one entry with z-value bigger than 3 were then eliminated to make data less volatile. The value of z-score equal to 3 guarantees that a data point that differs by three standard deviations from the average is considered an outlier, which is a sensible and commonly used practice. In the Figure 2 it can be seen that data is more evenly distributed after the procedure. As a result, the size of the dataset was reduced by 174 rows.

I have also conducted correlation analysis to detect some possible multicollinearity, please see Figure 3. Some variables had a high correlation between each other and it was decided to eliminate those variables that have a correlation higher than 0.8. In the end, V9 and V13 were eliminated as they were highly correlated with variables V1 and V11.

Other than that, the dataset had no missing values, the classes are relatively balanced and sample size is sufficiently big for the models used further.



(a) Distribution of variables V1, V2, V3 with outliers



(b) Distribution of variables V1, V2, V3 without outliers

Figure 2: Example distributions of particular variables before and after outlier elimination

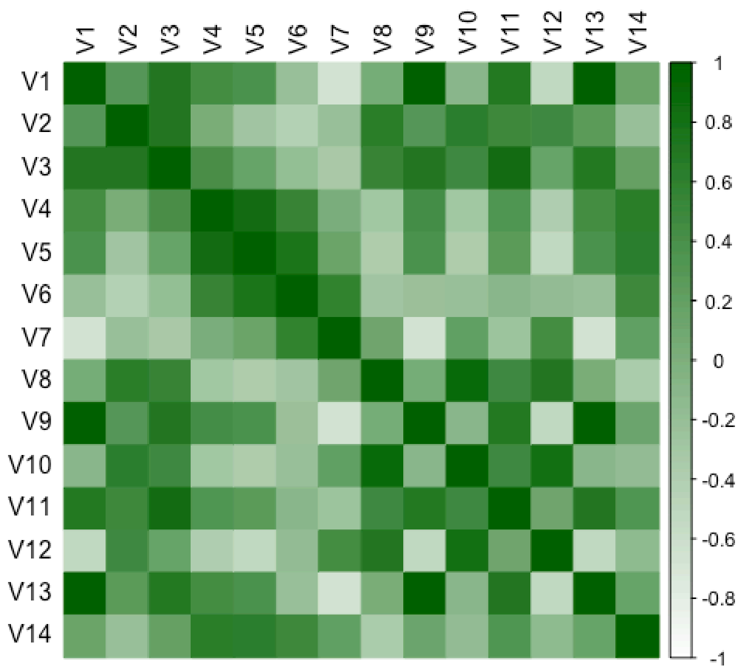


Figure 3: Correlation matrix before adjustments

1.2 Goals of the analysis

The goal of the analysis is to establish whether:

• EEG signals can accurately predict the eye state (open or closed) of an individual

• some sensors can be eliminated to decrease the potential costs of the experiment

• the performance of different classification algorithms varies when predicting eye states from EEG signals.

1.3 Key findings

• EEG signals could be used to explain a very high percentage of variance when predicting an eye state (up to 96%)

• Sensor O2 was shown to have the worst performance by both Logistic Regression and Random Forest model and could potentially be eliminated

• Top performer of the models was k-NN algorithm which could be explained by intrinsic structure of the data. However, for the variable analysis other models could be used, for example, Random Forest, that also showed comparably low error rate.

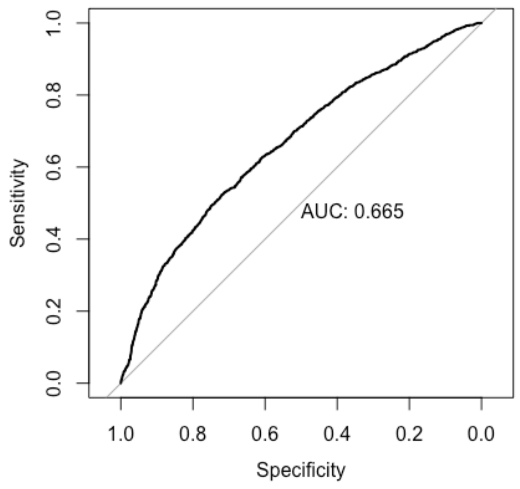
2 The Analysis

2.1 Logistic Regression

2.1.1 Model's performance

The first model I have decided to use was logistic regression. It is a great start in this case, I believe, because it is highly appropriate for the binary classification problem we have at hand, it is simple and also gives a room for improvement. As a comment, the creators of the dataset Oliver Rosler and David Suendermann in their paper "A First Step towards Eye State Prediction Using EEG" assessed a number of models' performance, including logistic regression. Logistic regression showed a moderate result with around 35% error rate.

Figure 4: ROC curve



The performance was assessed with the usual measures of accuracy, precision, sensitivity and specificity. Accuracy, proportion of the correctly predicted values over all values, is equal to the 63%. Precision, proportion of the correctly predicted closed eyes over all predictions, is equal to 22% (we should also take into account that number of closed eye instances is a bit smaller than open eye instances). Specificity and sensitivity were analysed using ROC curve that plots those parameters, please see figure 4. By plotting sensitivity (true positive rate) over false positive rate (1 - specificity) we get a ROC curve. The AUC - area under ROC curve - gives us a measure of the performance of the model, where AUC = 1 is a perfect model that was right in every case and AUC = 0 is a horrible model that was never right. Overall, according to AUC, model's performance is adequate and relates to previous experiments.

2.1.2 Model's results

The features correspond to 14 EEG measurements from the headset, originally labeled AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4, in that order, however, P8 and F8 were eliminated due to multicollinearity. In the table we can see that the most significant and influential sensors are P7, T7, F3, FC6 and then F7 and FC5 which are significant at a lesser level. We can also see that O2 is insignificant and can be eliminated from the model.

Sensor	Coef	Significance
AF3	0.006395	0.01
F7	-0.017910	0.01
F3	0.012949	0
FC5	-0.012321	0
T7	0.037516	0
P7	-0.038983	0
O1	0.001445	0
O2	0.004577	1
T8	0.003837	0.05
FC6	-0.009036	0
F4	0.010941	0.01
AF4	0.006100	0.05

2.2 Random Forest

2.2.1 Model's performance

In the previous works on this dataset, Random Forest model was shown to be one of the best performers.; it's error rate was smaller than 10 %. Random Trees is a developments of a Decision Tree model that divides predictor space into a number of regions; Random Trees adds an advancement to this which includes more than one tree.

Indeed, even with the small number of trees equal to 50, model's accuracy is equal to 91%, precision to 51%.

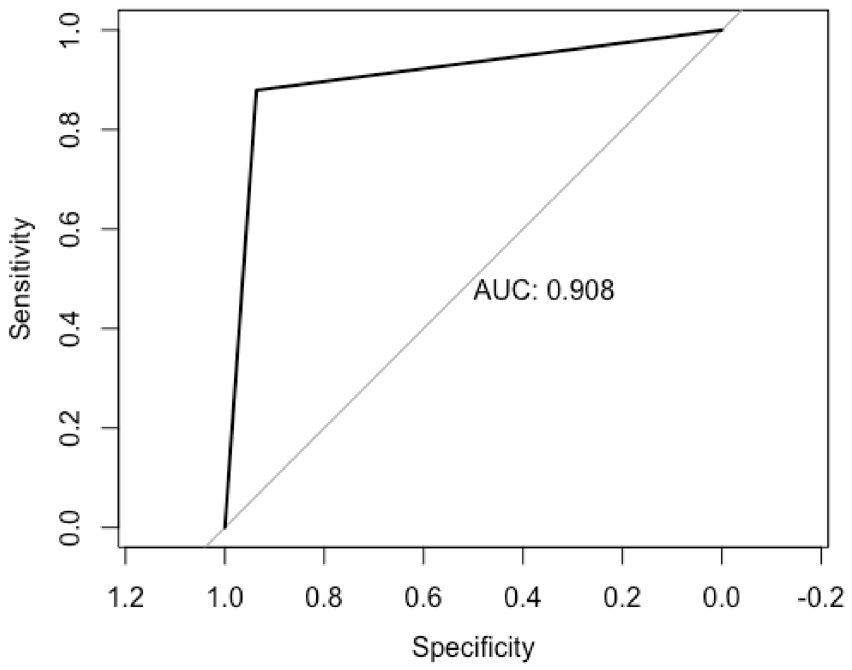


Figure 5: ROC curve for Random Forest

Seeing this result, it is possible that the relationship in the dataset is not linear.

2.2.2 Model's results

To assess the model's assessment of variables, we look at the variable importance plots. There are two types of assessment: by the average impact on model's accuracy and an Gini index. Gini index tells us about the purity of a model, i.e. a variable with high

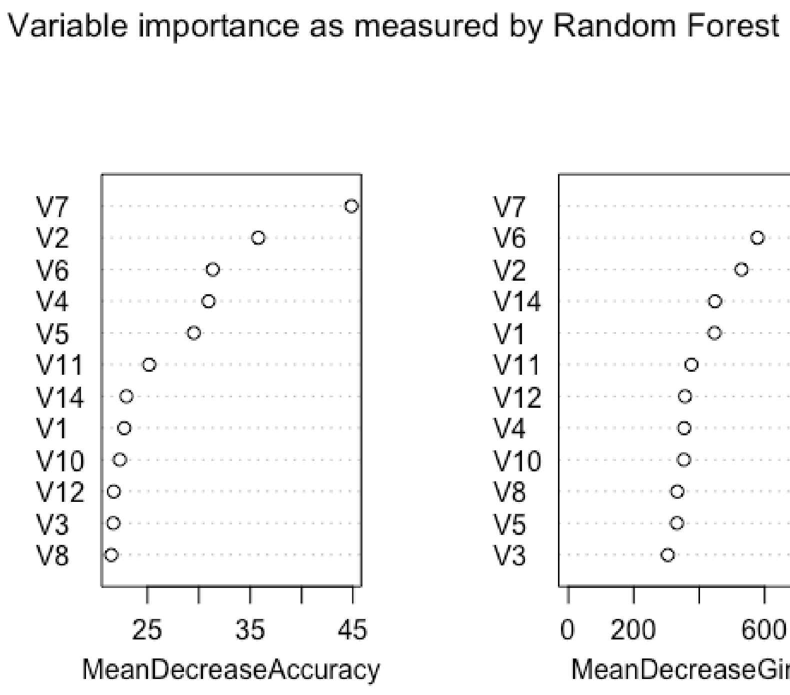


Figure 6: Variance importance

value of MeanDeacreaseGini gives a more homogenous subgroup if used for a split; a variable with high MeanDecreaseAccuracy has a high positive impact on model's accuracy. Both plots show us that variable V7 (O1) is of the highest importance for the model; comparing to the logistic regression, the variable was significant, however, its coefficient wasn't in the top largest. More importantly, we see that variables V3 (F3), V8 (O2) performed the worst, which supports the findings of the logistic model conserning O2, which was insignificant.

2.3 k-NN

2.3.1 Model's performance

Previous results show that the best performer model on the dataset is KStar model, which is a variation of the k-NearestNeighbours algorithm with added feature selection. Indeed, also the simpler k-NN model performs really well on the data: accuracy is equal to 96% and precision to 43%.

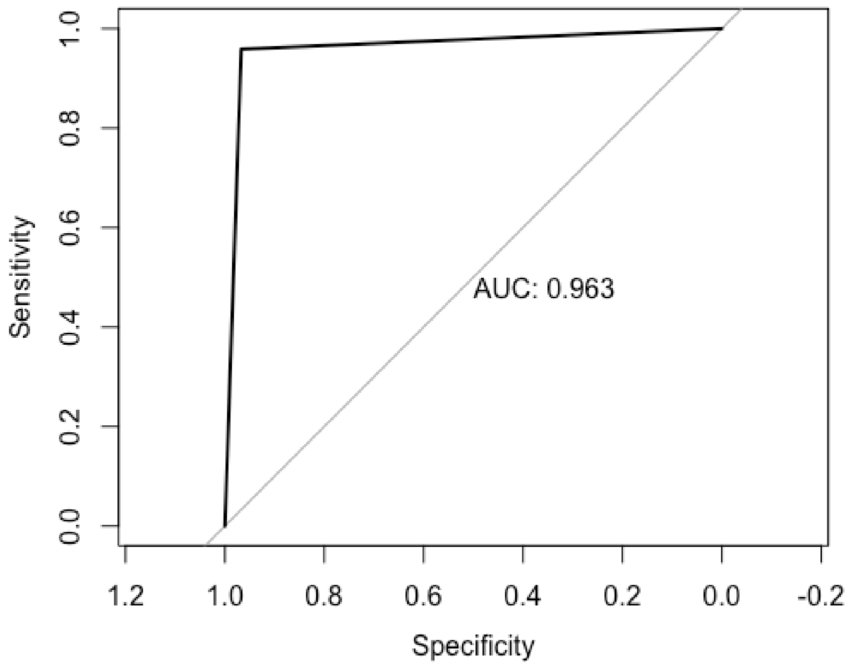


Figure 7: ROC curve for k-NN

K-NN algorithm is a model that separates points into a specified number of neighbourhoods based on the distance (usually Euclidian) between them. The model does not learn the data. As a result, it is impossible to assess the importance of the variable and their individual contribution directly, only indirectly by continuously eliminating the variables and the evaluating model's performance.

3 Conclusions

The goals set for this project were to establish whether EEG signals can be used to accurately predict the eye state, whether there are key EEG signals that could be eliminated due to low performance and assess the output of different supervised machine learning algorithms.

Definitely, the EEG signals have shown to be very useful in determining the eye state: the best performed model had an accuracy rate of 96%. The best performing model ended up being k-NN, followed by Random Forest and then by Logistic Regression. This tells us that more likely than not the relationships within data are non-linear and relationships within are complex. Non-linearity makes Logistic Regression perform worse and complex relationships (for example, some signals do not have a clear cut value that signifies an eye state) make Random Trees perform worse. Additionally, even though significant number of outliers were removed as well as highly correlated variables, it is possible that the data is still pretty noisy and some multicollinearity is present (but not one at an alarming level). This can also explain the models' performance, as k-NN is the one that is more robust to outliers, while Logistic Regression is pretty sensitive to multicollinearity. Lastly, even though the classes are represented in the dataset almost evenly, there are still more open eye states than not, it is possible that this could have affected the performance of the logistic regression.

A question whether some signals are more influential than others is important in this particular research endeavour, as decreasing number of signals helps reducing costs and make the technology more affordable. The results show that out of all sensors, O2 showed the worst performance and could be potentially eliminated to decrease costs.

The Appendix

References

[1] Rosler, O. and Suendermann, D. *A First Step towards Eye State Prediction Using EEG*, nternational Conference on Applied Informatics for Health and Life Sciences, 2013.

[2] Asquith, P., Ihshaish, H.. *Classification of eye-state using EEG recordings: speed-up gains using signal epochs and mutual information measure*, In Proceedings of the 23rd International Database Applications and Engineering Symposium IDEAS 2019, 2019.

R code

```
data <- read.csv("EEG Eye State.arff", header=FALSE, comment.char = "@")

### SPLITTING THE DATA ###

n_obs <- nrow(data)
train_proportion <- 0.7
test_proportion <- 0.3


train_size <- train_proportion*n_obs
test_size <- test_proportion*n_obs
train_size + test_size - n_obs #check
set.seed(17)
n <- seq(n_obs)

train_n <- sample(n, train_size, replace = FALSE)
n <- setdiff(n, train_n) #set difference between all indeces and those that have alre
test_n <- sample(n, test_size, replace = FALSE)


train_set <- data[train_n,]
test_set <- data[test_n,]

nrow(train_set)/n_obs #indeed 70%

### EDA ###

summary(data) #mean of the eye_state variable is 0.4488, which means there are more
# closed eye_sates than open
frequency <- c(sum(data$V15 == 1), sum(data$V15 == 0))
barplot(frequency, names.arg = c("Closed","Open"), horiz = TRUE, ylab = "Eye state di
install.packages('carData')
library(car)

help("plot")
layout(matrix(c(1, 2, 3), nrow = 1, ncol = 3))
plot(train_set$V1, main = "V1 distribution",xlab = NA, ylab = NA, col = "Dark Green")
plot(train_set$V2, main = ”V2 distribution”,xlab = NA, ylab = NA, col = ”Dark Green”)
plot(train_set$V3, main = ”V3 distribution”,xlab = NA, ylab = NA, col = ”Dark Green”)

layout(matrix(c(1, 2, 3), nrow = 1, ncol = 3))
plot(train_set$V4,main = ”V4 distribution”,xlab = NA, ylab = NA, col = ”Dark Green”)
plot(train_set$V5,main = ”V5 distribution”,xlab = NA, ylab = NA, col = ”Dark Green”)
plot(train_set$V6,main = ”V6 distribution”,xlab = NA, ylab = NA, col = ”Dark Green”)


plot(train_set$V7)
plot(train_set$V8)
plot(train_set$V9)
plot(train_set$V10)
plot(train_set$V11)
plot(train_set$V12)
plot(train_set$V13)
plot(train_set$V14)
# we can see there are some outliers in the variables
# I will use z-score method to detect and delete outliers for every var
z <- function(x) {
  (x - mean(x)) / sd(x)
}
train_set_noV15 <- train_set[,-ncol(train_set)]
z_scores <- apply(train_set_noV15, 2, z)
outliers <- train_set_noV15[apply(abs(z_scores) > 3, 1, any), ]
dim(outliers)
library(dplyr)
train_set1 <- anti_join(train_set, outliers, by = c(’V1’,’V2’,’V3’,’V4’,’V5’,’V6’,’V7
dim(train_set)- dim(train_set1)

summary(train_set1)
plot(train_set1$V1)

layout(matrix(c(1, 2, 3), nrow = 1, ncol = 3))
plot(train_set1$V1, main = ”V1 no outliers”,xlab = NA, ylab = NA, col = ”Dark Green”)
plot(train_set1$V2, main = ”V2 no outliers”,xlab = NA, ylab = NA, col = ”Dark Green”)
plot(train_set1$V3, main = ”V3 no outliers”,xlab = NA, ylab = NA, col = ”Dark Green”)
```

```
layout(matrix(c(1, 2, 3), nrow = 1, ncol = 3))
plot(train_set1$V4,main = ”V4 no outliers”,xlab = NA, ylab = NA, col = ”Dark Green”)
plot(train_set1$V5,main = ”V5 no outliers”,xlab = NA, ylab = NA, col = ”Dark Green”)
plot(train_set1$V6,main = ”V6 no outliers”,xlab = NA, ylab = NA, col = ”Dark Green”)



## correlation
train_set_noV15 <- train_set[,-ncol(train_set)]
train_set1 <- anti_join(train_set, outliers, by = c(’V1’,’V2’,’V3’,’V4’,’V5’,’V6’,’V7

layout(matrix(1))
install.packages(”corrplot”)
library(corrplot)
corrm <- cor(train_set_noV15)
my_palette <- colorRampPalette(c(”white”, ”dark green”))(n = 100)
corrplot(corrm, method = ”color”,col = my_palette, tl.col = ’black’)

diag(corrm) <- 0
max_cor <- max(corrm)
cor(train_set1)
indices <- which(corrm ==max_cor, arr.ind = TRUE)

#deleting V9
train_set1 <- train_set1[,-9]
dim(train_set1)

train_set_noV15 <- train_set_noV15[,-9]
corrm1 <- cor(train_set_noV15)
corrplot(corrm1, method = ”color”,col = my_palette, tl.col = ’black’)
diag(corrm1) <- 0
max_cor <- max(corrm1)
indices <- which(corrm1 ==max_cor, arr.ind = TRUE)

#deleting V13
train_set1 <- train_set1[,-12]
train_set_noV15 <- train_set_noV15[,-12]
corrm2 <- cor(train_set_noV15)
```

```
corrplot(corrm1, method = ”color”,col = my_palette, tl.col = ’black’)
diag(corrm2) <- 0
max_cor <- max(corrm2)
indices <- which(corrm2 ==max_cor, arr.ind = TRUE)




### TRAINING MODEL #1 ###
#Logistic Regression
lr_fit <- glm(V15 ~., data = train_set1,
              family=binomial(link=’logit’))
summary(lr_fit)

### EVALUATING MODEL #1 ###

#confusion matrix
lr_prob <- predict(lr_fit, test_set, type=”response”)
lr_pred <- ifelse(lr_prob > 0.5,”Yes”,”No”)
table(Predicted = lr_pred, Actual = test_set$V15)

#accuracy
lr_tab <- table(Predicted = lr_pred, Actual = test_set$V15)
lr_tab
accuracy <- (1850+987)/(1850+1050+607+987)
precision <- (987)/(1850+1050+607+987)
#ROC curve

library(pROC)
test_roc = roc(test_set$V15 ~ lr_prob, plot = TRUE, print.auc = TRUE)
as.numeric(test_roc$auc)
lr_prob

library(car)
vif(lr_fit) # variance inflation factors
sqrt(vif(lr_fit)) > 5
```

```
### TRAINING MODEL #2 ###
#Decision tree
install.packages(c("rpart","rpart.plot", "party","partykit"))
library("rpart")
library("rpart.plot")

tree<-rpart(V15 ~ ., data=train_set1, cp=0.2)
printcp(tree)
rpart.plot(tree)

#Random forest
install.packages('randomForest')
install.packages('caret')
library(randomForest)
library(caret)
help("randomForest")
set.seed(71)
train_set1$V15 <- as.factor(train_set1$V15)
rf_data = randomForest(V15 ~ ., data=train_set1, ntree=50, mtry=sqrt(12),importance =
varImpPlot(rf_data, sort = TRUE, main = "Variable importance as measured by Random Fo
help(varImpPlot)
rf_pred <- predict(rf_data, newdata = test_set)

### EVALUATING MODEL #2 ###

#accuracy
lr_tab_rf <- table(Predicted = rf_pred, Actual = test_set$V15)
lr_tab_rf
accuracy <- (2306+1786)/(2306+1786+246+156)
precision <- (2306)/(2306+1786+246+156)
#ROC curve
rf_pred <- as.numeric(rf_pred)
library(pROC)
test_roc = roc(test_set$V15 ~ rf_pred, plot = TRUE, print.auc = TRUE)
as.numeric(test_roc$auc)

### TRAINING MODEL #3 ###
```

```
library(class)
test_set1 <-test_set[,-9]
test_set1 <-test_set1[,-12]
dim(train_set1)
dim(test_set1)
knn.pred=knn(train_set1,test_set1,cl = train_set1[,13],k=3)


### EVALUATING MODEL #3 ###
table(knn.pred,test_set$V15)
mean(knn.pred==test_set$V15)
accuracy <- (2381+1948)/(2381+1948+81+84)
precision <- (1948)/(2381+1948+81+84)
knn.pred <- as.numeric(knn.pred)
test_roc = roc(test_set$V15 ~ knn.pred, plot = TRUE, print.auc = TRUE)
```