Project for Data Science Development

GEORGIOS ORFANAKIS

06/05/2020

```
module: "Data Science Development CMM535"
matriculation No: "1903446"
```

CONTENTS

--PROJECT--

1. PART-1 Introduction

The aim of this project is to use a dataset to address a classfication problem, provide a report showing the steps of this process and finally provide the results and some insight gained on the data. The dataset used here comes from an electroencephalogram (EEG) and goes under the name "eye_State" with the variable of interest being "eyeDetection". It is in the form of "0"-eyes open and "1"-eyes closed. It is assessed through 14 variables measured within a unit of time. The models created for this purpose are trained, tested and evaluated using these variables to predict if the eyes are open or closed. Time is not of importance in this study, especially, since as seen in the exploratory data analysis (EDA) all important activity of the eyes takes place within a very short time span of the whole EEG. This is a classification problem aimed to create a classifier being able to distinguish and predict the state of the eyes, open or close.

The classifiers used are Support Vector Machine (SVM) on the dataset, Support Vector Machine (SVM) on the PCA transformed dataset, and logistic regression through backward elimination on the dataset.

1.1 About the dataset

Dataset Description:

The data set consists of 14 EEG values and a value indicating the eye state. All data is from one continuous EEG measurement with the Emotiv EEG Neuroheadset. The duration of the measurement was 117 seconds. The eye state was detected via a camera during the EEG measurement and added later manually to the file after analysing the video frames. '1' indicates the eyes-closed state and '0' the eyes-open state. All values are in chronological order with the first measured value at the top of the data.

Number of Instances: 14980 Number of Attributes: 15

Due to the dataset being quite big, a random part of it was used.

```
#link to dataset: https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State#
```

1.2 Algorithms used

Support Vector Machine (SVM): A supervised machine learning algorithm which can be used for both classification or regression challenges. Here it is used for classification. In the SVM algorithm, we plot each data item as a point in n-dimensional space, with n being the number of features, with the value of each feature being the value of a particular coordinate. Then, classification is performed by finding the hyper-plane that differentiates the classes. Support Vectors are simply the co-ordinates of individual observations.[1]

Logistic regression: An algorithm that is used in solving classification problems. It is a predictive analysis that describes data and explains the relationship between variables. Logistic regression is applied to an input variable (X) where the output variable (y) is a discrete value which ranges between 1 and 0. It uses logistic (sigmoid) function to find the relationship between variables. The sigmoid function is an S-shaped curve that can take any real-value number and map it to a value between 0 and 1, but never exactly at those limits. The output of the logistic regression model is the probability of the successful outcome. [2] Generalized linear models equate the linear component to some function of the probability of a given outcome on the dependent variable. In logistic regression, that function is the logit transform: the natural logarithm of the odds that some event will occur.For logistic regression maximum likelihood estimation is used to solve for the parameters that best fit the data. [6]

Principal Component Analysis (PCA): Though PCA is not used here as a machine learning algorithm it is still used as an important process for de-noising the data. Principal component analysis (PCA) is a technique for reducing the dimensionality of such datasets, increasing interpretability but at the same time minimizing information loss. It does so by creating new uncorrelated variables that successively maximize variance. Finding such new variables, the principal components (PCs), reduces to solving an eigenvalue/eigenvector problem, and the new variables are defined by the dataset at hand, not a priori, hence making PCA an adaptive data analysis technique. It is adaptive in another sense too, since variants of the technique have been developed that are tailored to various different data types and structures. [3] Technically, a principal component can be defined as a linear combination of optimally-weighted observed variables. The output of PCA are these principal components, the number of which is less than or equal to the number of original variables. The principal components created provide low-dimensional linear surfaces that are closest to the observations. This means that principal components are not the initial predictors.

1.3 Metrics used

Metrics chosen:

Accuracy: the ratio of correctly predicted observation to the total observations.

Recall: the ratio of correctly predicted positive observations to the all observations in actual class

Precision: the ratio of correctly predicted positive observations to the total predicted positive observations.

F1-score: the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account.

p-value: shows statistical significance. If p-value <= 0.05 for 95% confidence interval it means that there is enough evidence to reject the null hypothesis or simply put that results are not due to chance.

Cohen's kappa coefficient/kappa score: measures the degree of agreement between the true values and the predicted values, which we use as the classifier's performance. 1 means perfect agreement and 0 is chance agreement.

VALUE OF K	LEVEL OF AGREEMENT	% OF DATA THAT ARE RELIABLE
0 - 0.20	None	0 - 4%
0.21 - 0.39	Minimal	4 - 15%
0.40 - 0.59	Weak	15 - 35%
0.60 - 0.79	Moderate	35 - 63%
0.80 - 0.90	Strong	64 - 81%
Above 0.90	Almost Perfect	82 - 100%

Matthews correlation coefficient (MCC)/phi-coefficient: The Matthews correlation coefficient is used in machine learning as a measure of the quality of binary and multiclass classifications. It takes into account true and false positives and negatives and is generally regarded as a balanced measure which can be used even if the classes are of very different sizes. The MCC is in essence a correlation coefficient value between -1 and +1. A coefficient of +1 represents a perfect prediction, 0 an average random prediction and -1 an inverse prediction. The statistic is also known as the phi coefficient [4]

ROC-AUCROC: The ROC curve is a popular graphic for simultaneously displaying the two types of errors for all possible thresholds.The overall performance of a classifier, summarized over all possible thresholds, is given by the area under the (ROC) curve (AUC).[5] ROC is a probability curve and AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes. The higher the AUC, the better the model is at predicting 0s as 0s and 1s as 1s. when AUC is 0.5, it means model has no class separation capacity whatsoever. The ROC curve is plotted with TPR-(Sensitivity-Recall) against the FPR(1-Specificity) where TPR is on y-axis and FPR is on the x-axis.

2. PART-2 EDA

2.1 Loading libraries & initial dataset

```
library(tidyverse)
library(foreign)
library(corrplot)
library(gridExtra)
library(grid)
library(ggplot2)
library(lattice)
require(reshape2)
library(caret)
library(e1071)
require(ggplot2)
library(ggbiplot)
library(MLmetrics)
```

```
library(mccr)
library(pROC)
library(gdata)
setwd("C:\\Users\\User\\Documents\\ds masters\\sem2\\data science dev CMM535\\eu")
eye_state <- read.arff("EEG Eye State.arff")
```

2.2 Overview of initial dataset

```
summary(eye_state)
```

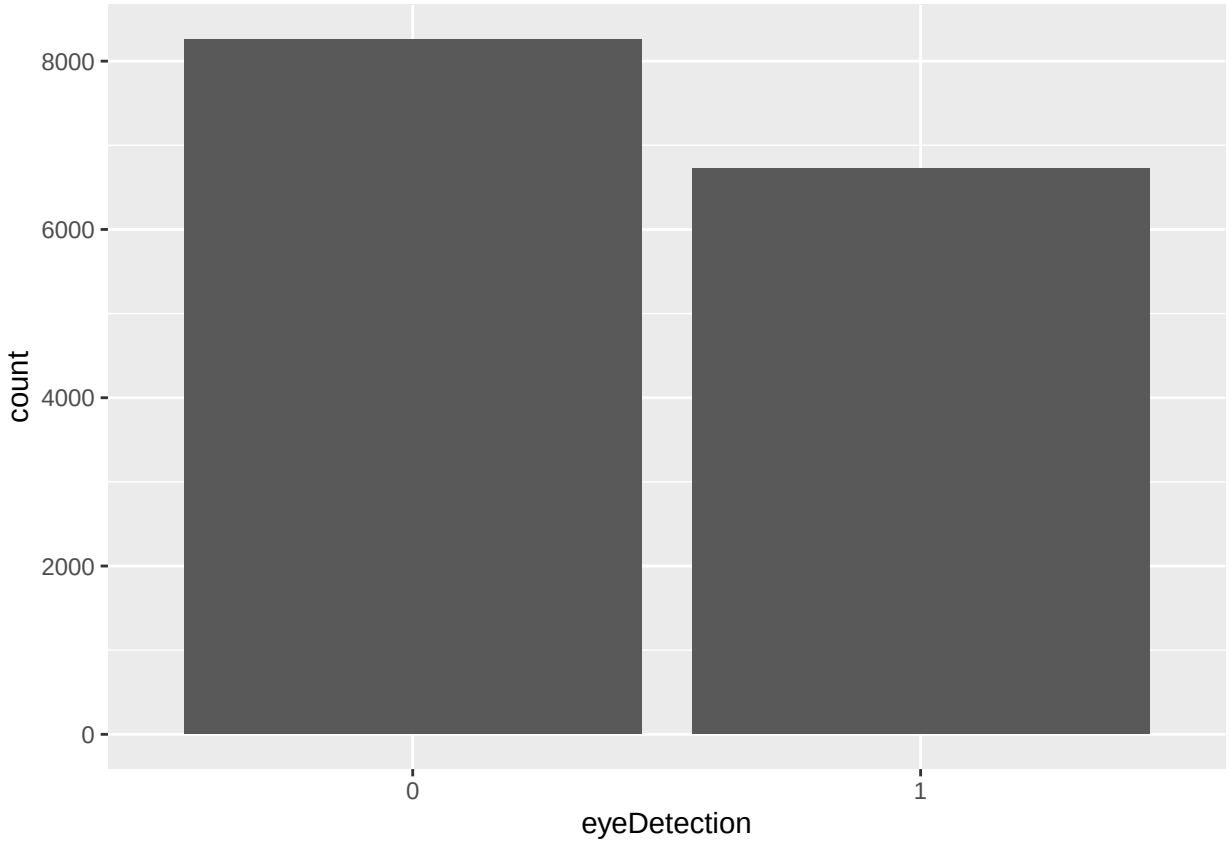
```
##       AF3               F7              F3             FC5               T7
##  Min.   :  1031   Min.   :2831   Min.   :1040   Min.   :  2453   Min.   :2090
##  1st Qu.:  4281   1st Qu.:3991   1st Qu.:4250   1st Qu.:  4108   1st Qu.:4332
##  Median :  4294   Median :4006   Median :4263   Median :  4120   Median :4339
##  Mean   :  4322   Mean   :4010   Mean   :4264   Mean   :  4165   Mean   :4342
##  3rd Qu.:  4312   3rd Qu.:4023   3rd Qu.:4271   3rd Qu.:  4132   3rd Qu.:4347
##  Max.   :309231   Max.   :7805   Max.   :6881   Max.   :642564   Max.   :6474
##        P7               O1              O2             P8
##  Min.   :  2768   Min.   :  2086   Min.   :4567   Min.   :  1358
##  1st Qu.:  4612   1st Qu.:  4058   1st Qu.:4605   1st Qu.:  4191
##  Median :  4618   Median :  4070   Median :4613   Median :  4199
##  Mean   :  4644   Mean   :  4110   Mean   :4616   Mean   :  4219
##  3rd Qu.:  4627   3rd Qu.:  4084   3rd Qu.:4624   3rd Qu.:  4209
##  Max.   :362564   Max.   :567179   Max.   :7264   Max.   :265641
##        T8              FC6             F4              F8
##  Min.   :1816   Min.   :3273   Min.   :2258   Min.   :    86.67
##  1st Qu.:4221   1st Qu.:4190   1st Qu.:4268   1st Qu.:  4590.77
##  Median :4229   Median :4201   Median :4277   Median :  4603.08
##  Mean   :4231   Mean   :4202   Mean   :4279   Mean   :  4615.21
##  3rd Qu.:4239   3rd Qu.:4211   3rd Qu.:4287   3rd Qu.:  4617.44
##  Max.   :6674   Max.   :6823   Max.   :7003   Max.   :152308.00
##       AF4          eyeDetection
##  Min.   :  1366   0:8257
##  1st Qu.:  4342   1:6723
##  Median :  4355
##  Mean   :  4416
##  3rd Qu.:  4373
##  Max.   :715897
```

```
str(eye_state)
```

```
## 'data.frame':    14980 obs. of  15 variables:
##  $ AF3         : num  4329 4325 4328 4329 4326 ...
##  $ F7          : num  4009 4005 4007 4012 4012 ...
##  $ F3          : num  4289 4294 4295 4296 4292 ...
##  $ FC5         : num  4148 4149 4156 4156 4151 ...
##  $ T7          : num  4350 4342 4337 4344 4348 ...
##  $ P7          : num  4586 4587 4584 4583 4587 ...
##  $ O1          : num  4097 4097 4097 4097 4096 ...
##  $ O2          : num  4641 4639 4630 4631 4628 ...
```

```
##  $ P8          : num  4222 4211 4208 4217 4211 ...
##  $ T8          : num  4238 4227 4222 4235 4244 ...
##  $ FC6         : num  4211 4208 4207 4211 4213 ...
##  $ F4          : num  4281 4279 4282 4288 4288 ...
##  $ F8          : num  4636 4633 4629 4632 4633 ...
##  $ AF4         : num  4394 4384 4389 4396 4398 ...
##  $ eyeDetection: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

Above we can see some basic statistics of the initial dataset.



We can see here the barplots of the counts of the two states of interest, eyes-open as "0" and ayes-closed as "1". The dataset is slightly imbalanced with eyes-open state having more observations.

2.3 Random sampling of initial dataset & overview of the new one

Since the dataset is quite big, a random half of it will be used.

```
##       AF3              F7              F3              FC5             T7
##  Min.   :  1031   Min.   :3798   Min.   :1040   Min.   :2453   Min.   :2090
##  1st Qu.:  4281   1st Qu.:3990   1st Qu.:4250   1st Qu.:4108   1st Qu.:4332
##  Median :  4294   Median :4006   Median :4263   Median :4121   Median :4339
##  Mean   :  4342   Mean   :4010   Mean   :4264   Mean   :4122   Mean   :4342
##  3rd Qu.:  4312   3rd Qu.:4024   3rd Qu.:4270   3rd Qu.:4132   3rd Qu.:4347
##  Max.   :309231   Max.   :7805   Max.   :6881   Max.   :5416   Max.   :6041
##        P7               O1              O2              P8              T8
##  Min.   :  2768   Min.   :2086   Min.   :4571   Min.   :  1358   Min.   :3915
```

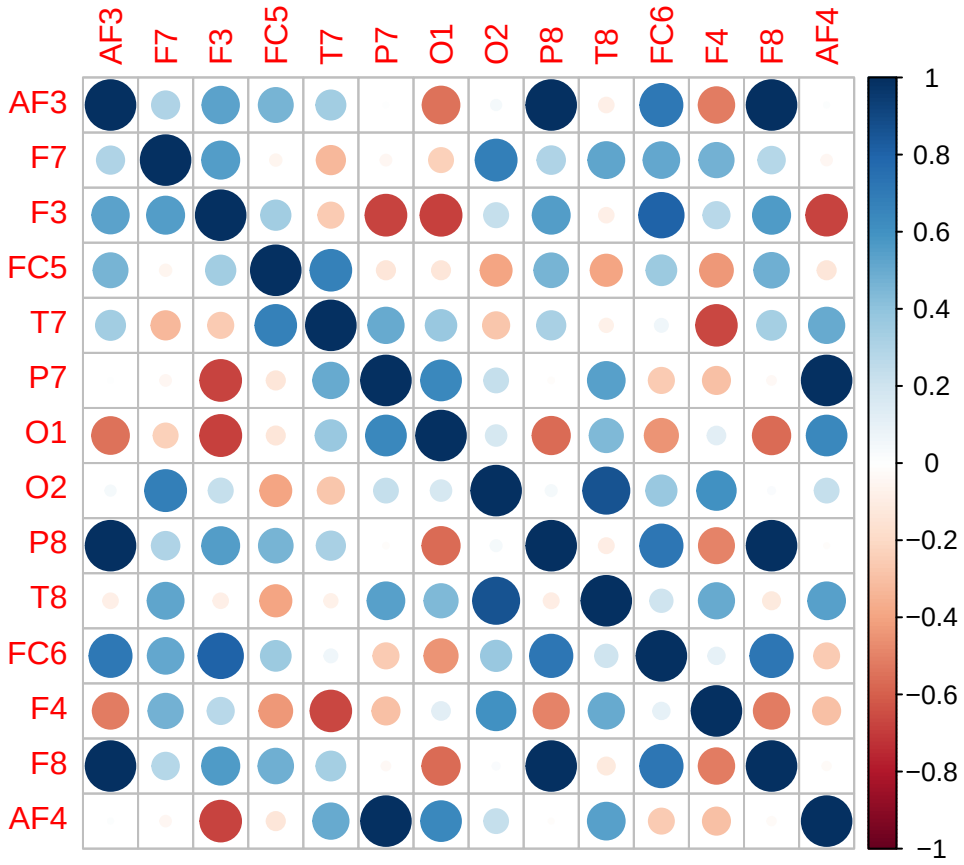
```
##  1st Qu.:  4612   1st Qu.:4058   1st Qu.:4605   1st Qu.:  4191   1st Qu.:4221
##  Median :  4618   Median :4070   Median :4614   Median :  4200   Median :4230
##  Mean   :  4668   Mean   :4073   Mean   :4616   Mean   :  4236   Mean   :4232
##  3rd Qu.:  4626   3rd Qu.:4084   3rd Qu.:4624   3rd Qu.:  4209   3rd Qu.:4240
##  Max.   :362564   Max.   :6350   Max.   :7264   Max.   :265641   Max.   :6674
##       FC6             F4             F8                AF4
##  Min.   :3273   Min.   :2258   Min.   :    86.67   Min.   :  1366
##  1st Qu.:4190   1st Qu.:4268   1st Qu.:  4590.77   1st Qu.:  4342
##  Median :4201   Median :4277   Median :  4603.08   Median :  4355
##  Mean   :4203   Mean   :4279   Mean   :  4624.82   Mean   :  4456
##  3rd Qu.:4211   3rd Qu.:4287   3rd Qu.:  4617.95   3rd Qu.:  4373
##  Max.   :6823   Max.   :7003   Max.   :152308.00   Max.   :715897
##  eyeDetection
##  0:4119
##  1:3371
##
##
##
##
```

Above we can see some basic statistics of the random part of the initial dataset we will work with.

2.4 Correlation of Variables

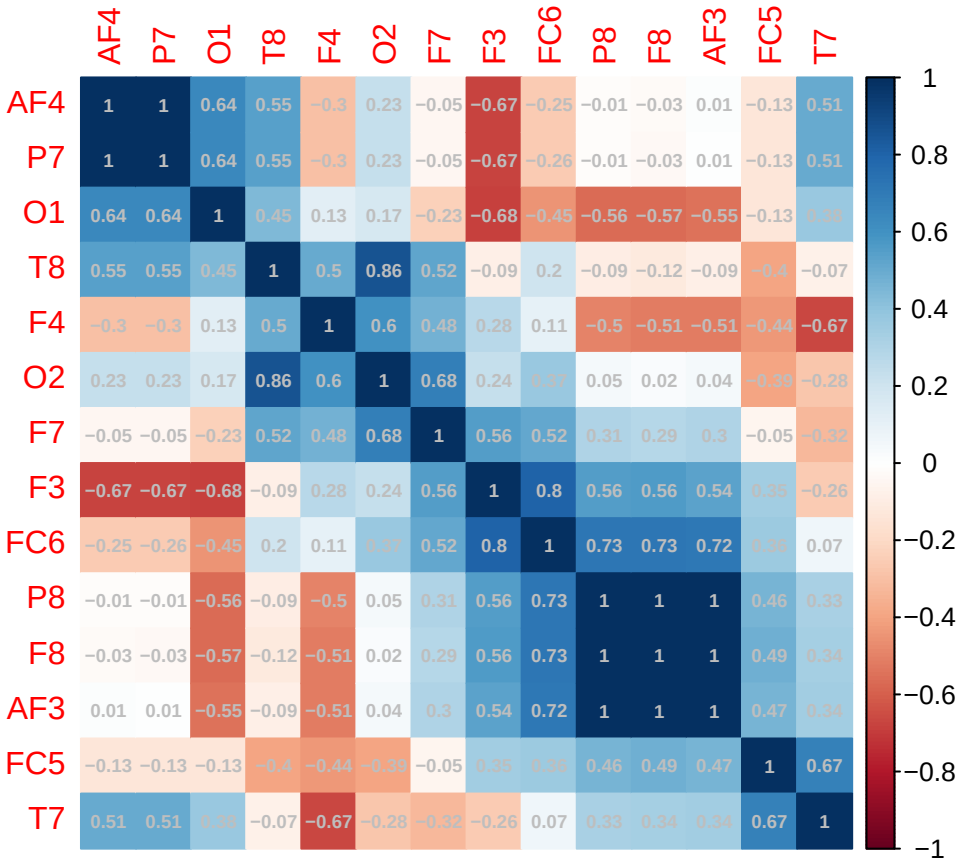
Now that the dataset to work with is set, we can go on and perform some deeper exploratory data analysis (EDA) on it. First, the correlation of the variables is plotted.

```
## Corelation plots
Corplot1 <-  corrplot(cor(X0[,-ncol(X0)]))
```



The bluer the dot the more positive the correlation. The more red the dot is the more negative the correlation. The bigger the size of the dot the stronger the correlation. AF3~P8 have very strong positive correlation. Same goes for AF3~F8, as well as for P7~AF4, and F8~P8. F3~FC6, T8~O2 have strong positive correlation. On the other hand, F3~P7, F3~O1, F3~AF4 and F4~P7 have quite strong negative correlation.

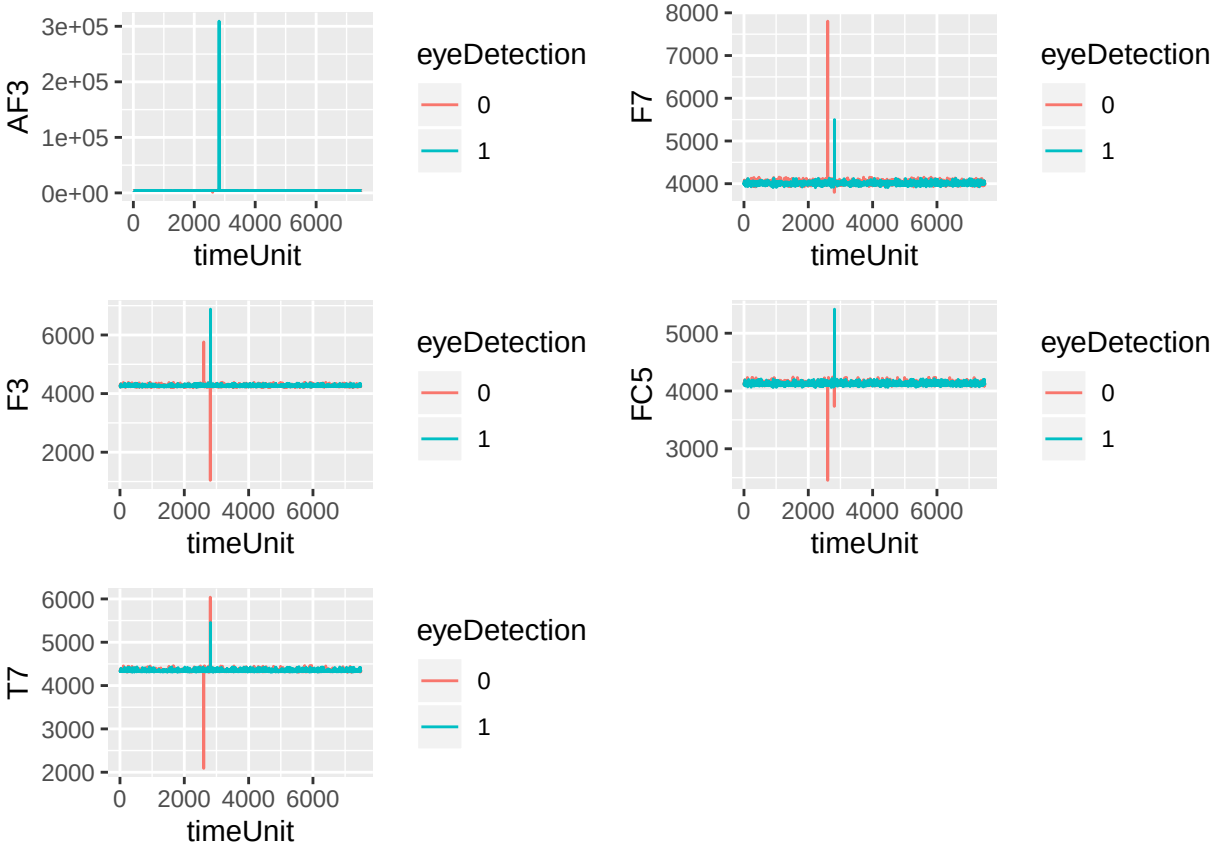
```
Corplot2 <- corrplot(cor(X0[,-ncol(X0)]), order = "AOE", method = "color",
                     addCoef.col = "gray", number.cex=0.55)
```

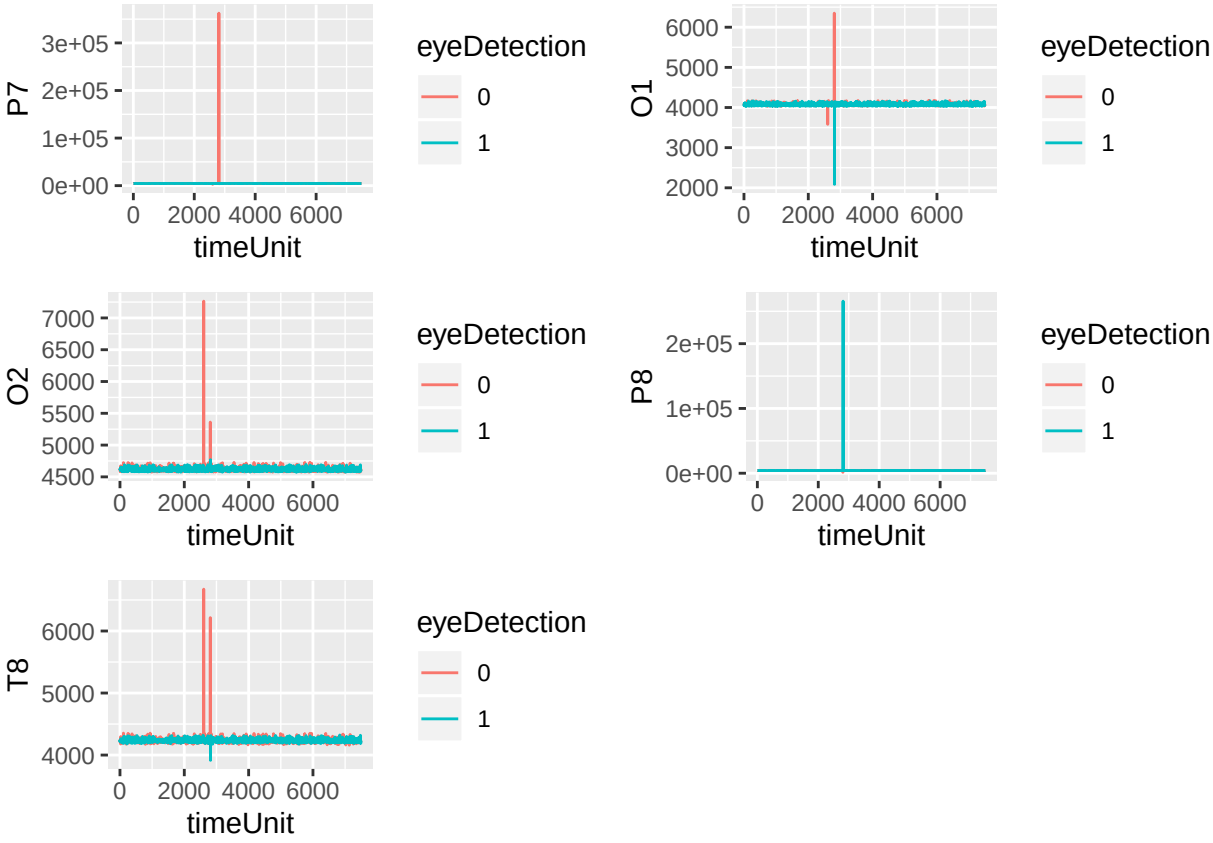


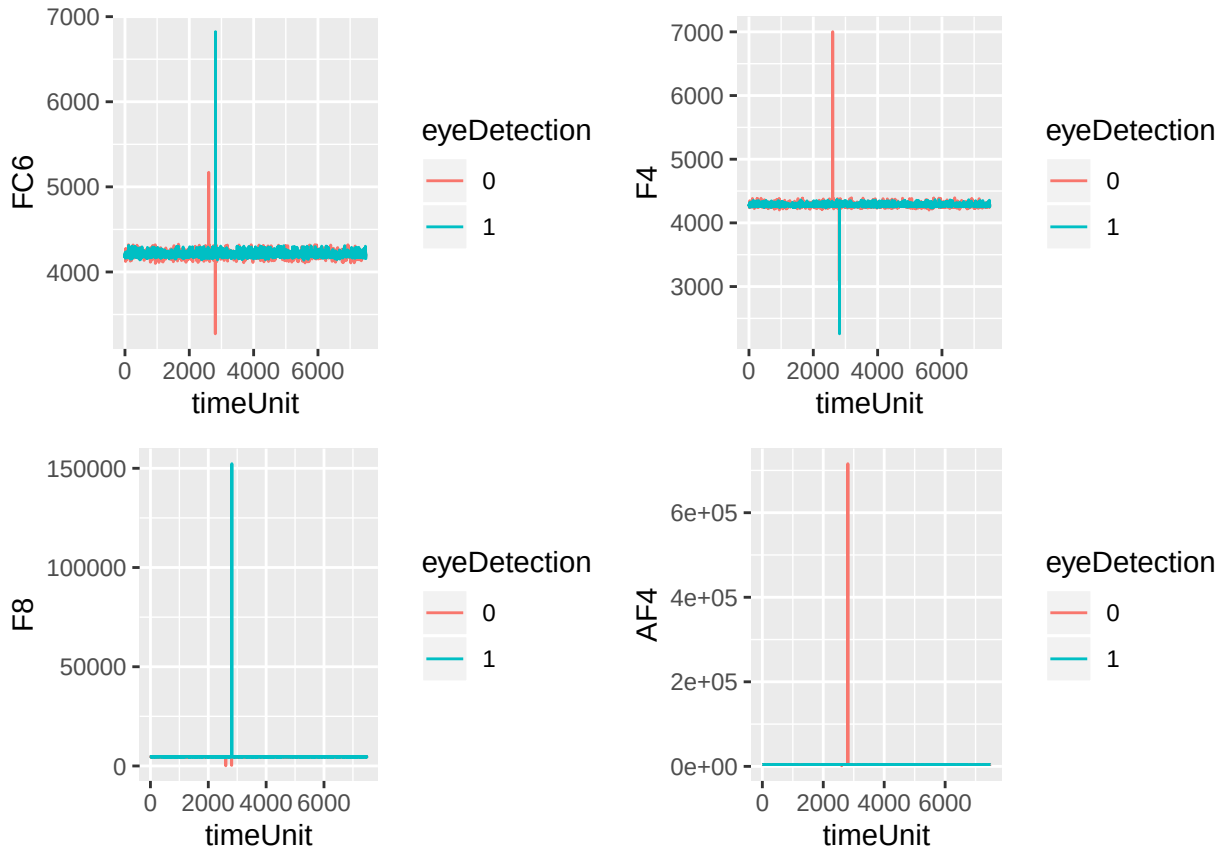
A more detailed represantation of the correlation can be seen on the second diagram, where Pearson correlation coefficient is also depicted. Same rules apply regarding the coloring, though white color basically means p-value close to zero. This translates into the two variables either being independent or non-lineary related.

2.5 Plotting the data over time

Now we plot the each variable with regards to whether eyes are open or closed over the unit of time to see how each variable detects the state of the eyes.


AF3
3e+05
2e+05
1e+05
0e+00
0
2000
4000
6000
timeUnit
eyeDetection
0
1
F7
8000
7000
6000
5000
4000
F3
FC5
T7

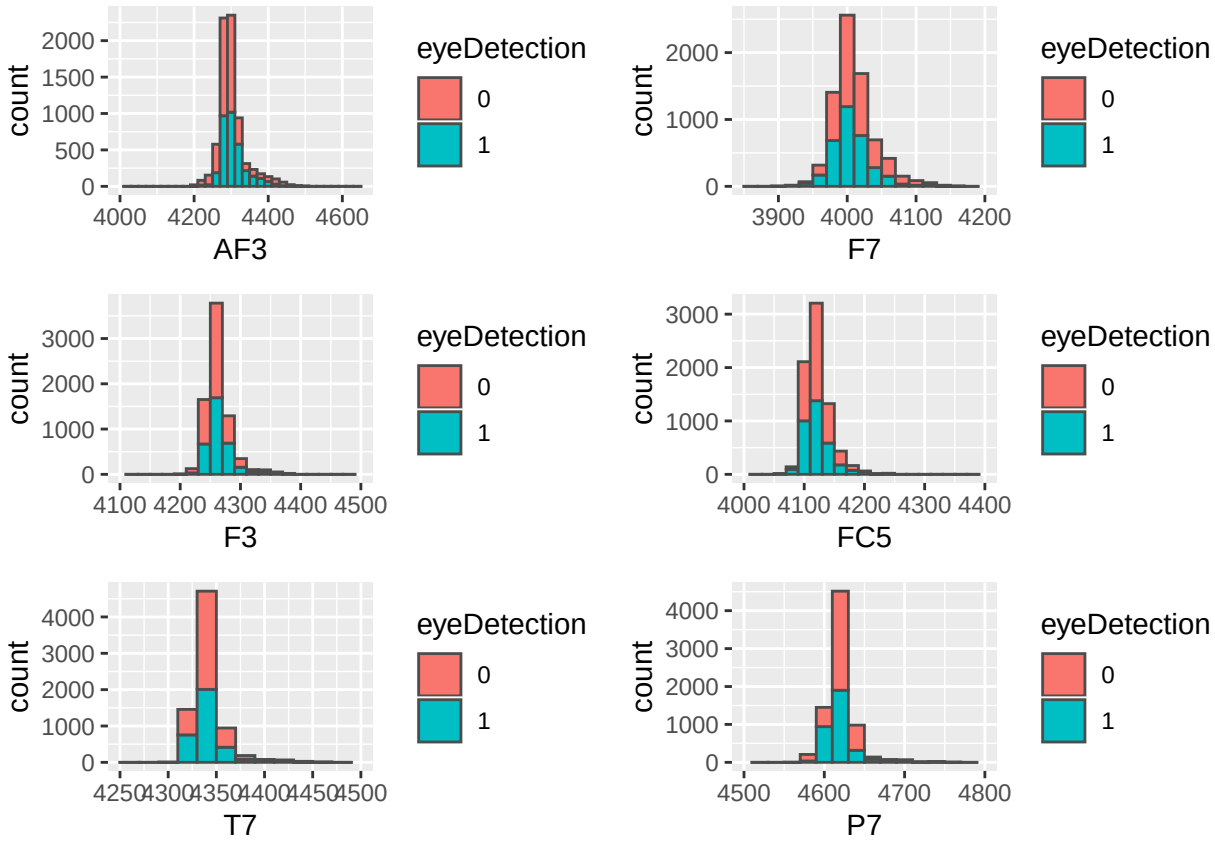

P7
3e+05
2e+05
1e+05
0e+00
0
2000
4000
6000
timeUnit
eyeDetection
0
1
O1
6000
5000
4000
3000
2000
O2
7000
6500
5500
4500
P8
T8

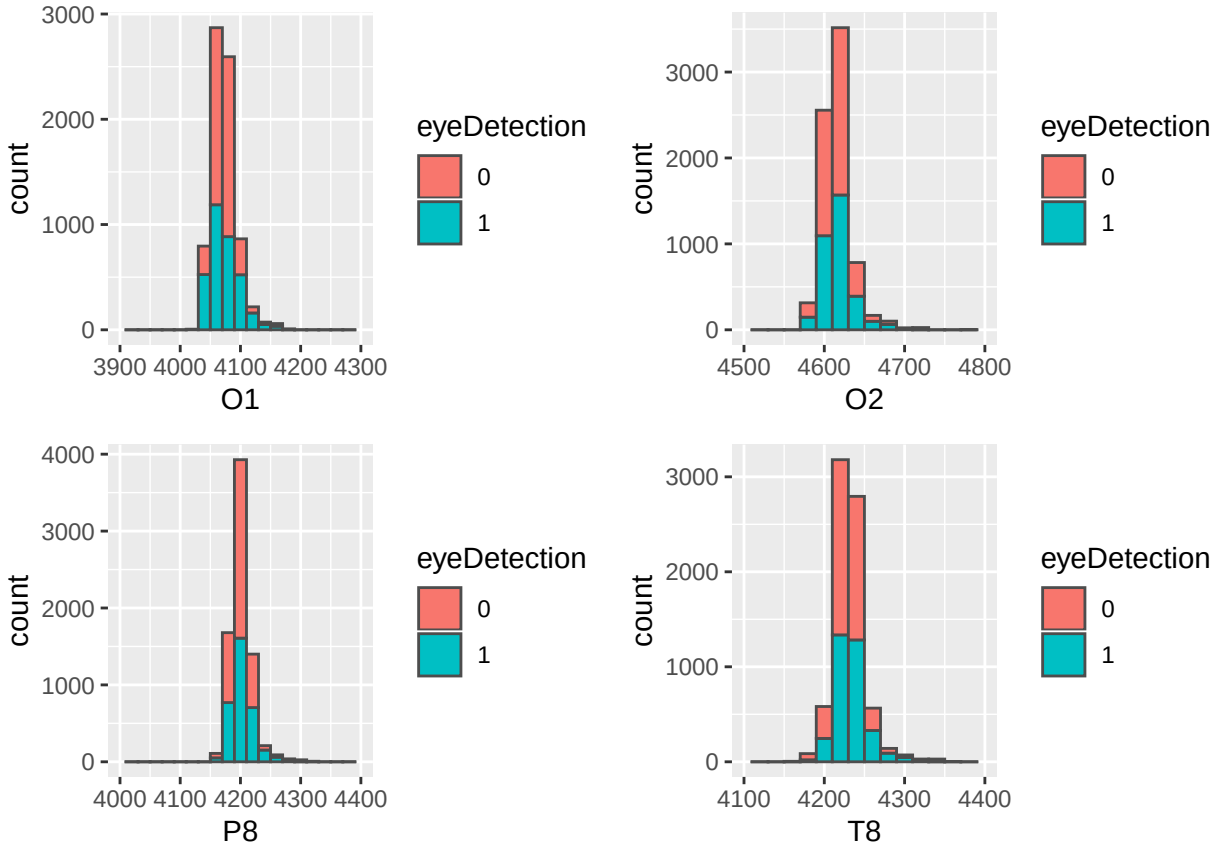


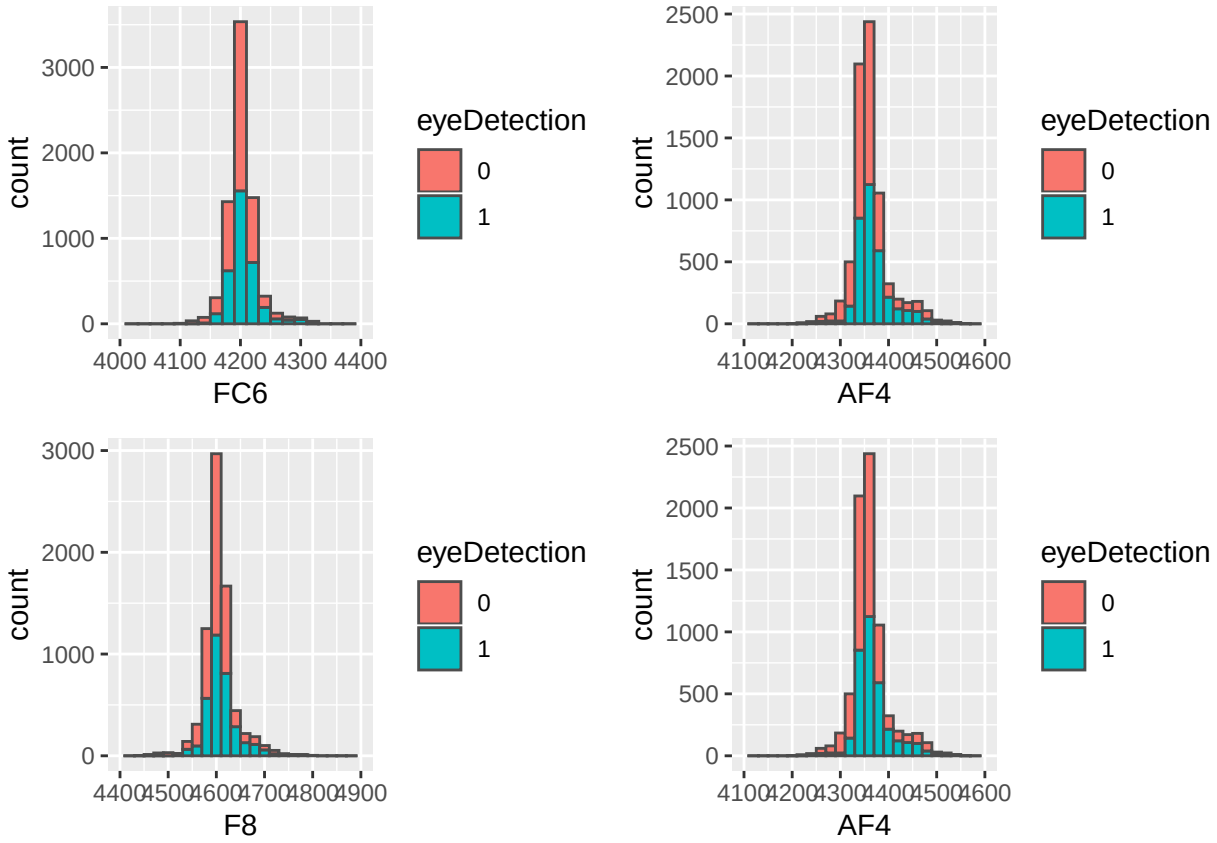
As we can see from all graphs the highest peak for 0 which means eyes open happens around the same time for the same amount of time for all features monitored. We can also see open eyes being better monitored by certain features i.e. AF3 variable seems to not detect the eyes open peak at all. Obviously, some features play a more important role for detecting the state of the eyes. To further dwelve on these points the following graphs are drawn.

2.6 Plotting the data to spot patterns over class of interest

In the plots below we see the absolute and relative relationship between eyes' state in the unit of time for each detection variable.


count
2000
1500
1000
500
0
4000
4200
4400
4600
AF3
eyeDetection
0
1
count
2000
1000
0
3900
4000
4100
4200
F7
eyeDetection
0
1
count
3000
2000
1000
0
4100
4200
4300
4400
4500
F3
eyeDetection
0
1
count
3000
2000
1000
0
4000
4100
4200
4300
4400
FC5
eyeDetection
0
1
count
4000
3000
2000
1000
0
4250
4300
4350
4400
4450
4500
T7
eyeDetection
0
1
count
4000
3000
2000
1000
0
4500
4600
4700
4800
P7
eyeDetection
0
1


3000
2000
1000
0
count
3900 4000 4100 4200 4300
O1
eyeDetection
0
1
3000
2000
1000
0
count
4500 4600 4700 4800
O2
eyeDetection
0
1
4000
3000
2000
1000
0
count
4000 4100 4200 4300 4400
P8
eyeDetection
0
1
3000
2000
1000
0
count
4100 4200 4300 4400
T8
eyeDetection
0
1



We can clearly see that all data is gathered within 4000 and 4600 unit of time. Also in all histograms we can clearly see that there is a central point were the count of observations peak and it is at that brief point that the state of eyes open seems to surpass the state of eyes closed. We can also see that not all variables get as many counts nor do they count the two states of the class of interest in the same manner.

3. PART-3 Model Building, Validating, Evaluating

Now that we have a better understanding of the dataset, variables and class of interest, we will begin with building a model that predicts the eyes' state. As predictors, all variables will be used for predicting the class of interest, eyes-open (state "0") or eyes-closed (state "1").

3.1 Training/test set split

First, we will randomly split the data into training/test set. 75% of data is for training set and 25% for test set.

```
## Splitting dataset into trainset/testset

#X0 <- eye_state

n <- nrow(X0)  # Number of observations
ntrain <- round(n*0.75)  # 75% for training set
set.seed(314)    # Set seed for reproducible results
tindex <- sample(n, ntrain)   # Create a random index
```

```r
train_X0 <- X0[tindex,]   # Create training set
test_X0 <- X0[-tindex,]
#colnames(X0)
#nrow(train_X0)
#nrow(test_X0)
```

No. rows-training set	No. rows-test set
5618	1872

3.2 Model building using SVM algorithm

We will go on and build our first model using the SVM algorithm. We will try to figure out which are the best parameters to use through 10-fold cross validation.

```r
## tuning basically performs via 10-fold cross validation

set.seed(100)
tune.out <- tune(svm, eyeDetection~. , data = train_X0, scale =TRUE,
                 ranges = list(cost = c(1, 10),
                               kernel = c("linear","radial", "polynomial"),
                               gamma=c(0.1, 0.01)))

#tune.out$sampling
#tune.out$performances
#tune.out$best.model
#tune.out$best.parameters
```

cost	kernel	gamma	error	dispersion
1	linear	0.10	0.3917813	0.02189864
10	linear	0.10	0.3844825	0.01892207
1	radial	0.10	0.2894336	0.04769134
10*	radial*	0.10*	0.2333600	0.04577888
1	polynomial	0.10	0.4179433	0.03953014
10	polynomial	0.10	0.3843058	0.05503602
1	linear	0.01	0.3917813	0.02189864
10	linear	0.01	0.3844825	0.01892207
1	radial	0.01	0.4150938	0.04234027
10	radial	0.01	0.3609860	0.05261037

The best model is radial with cost=10 and gamma = 0.1.

Now the best model is applied on the training set.

```r
### SVM on training set

set.seed(100)
svm1 <- svm(eyeDetection ~., data=train_X0,
          method="C-classification", kernal="radial", scale =TRUE,
          gamma=0.1, cost=10, probability=TRUE)
```

```r
## our model-svm1-with its confusion matrix to measure performance over training set
summary(svm1)
```

```
##
## Call:
## svm(formula = eyeDetection ~ ., data = train_X0, method = "C-classification",
##     kernal = "radial", gamma = 0.1, cost = 10, probability = TRUE,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  3759
##
##  ( 1887 1872 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```r
prediction <- predict(svm1, train_X0)
svmtraincm <- confusionMatrix(prediction, train_X0$eyeDetection, dnn = c("Prediction", "Reference"))
library(mccr)
mcc1_trSVM <- mccr(train_X0$eyeDetection, prediction)

#svmtraincm$byClass
```

Accuracy	Sensitivity/Recall	Specificity	Precision	F1-score	Kappa	MCC
0.7668	0.8946	0.6093	0.7384	0.8090	0.5166	0.5324

The model seems quite promising with 76.7% accuracy, Sensitivity/Recall at 89.5%, and Specificity at 60.9%. This figures are supported by a p-value<0.05 at 95% confidence interval, which means that the model output is of statistical importance. We need to bare in mind that kappa score is pretty low translating into a weak level of agreement between the predicted and the true values. Same goes for MCC wich further proves the low level of agreement between the predicted and the true values.

With these in mind we can go on and apply the model on the test set.

```r
## Model tested on test set

predicted_test <- predict(svm1, test_X0, probability = TRUE)

svmtsetcm <-confusionMatrix(predicted_test, test_X0$eyeDetection,
                            dnn = c("Prediction", "Reference"))
mcc1_teSVM <- mccr(test_X0$eyeDetection, predicted_test)
```

```
#svmtsetcm$byClass
```

Accuracy	Sensitivity/Recall	Specificity	Precision	F1-score	Kappa	MCC
0.7532	0.8909	0.5895	0.7208	0.7968	0.4913	0.5094

Accuracy is at 75%, sensitivity/recall is at 89% and specificity is at 60%. These figures are supported by a p-value<0.05 at 95% confidence intervals. This means that the model performs as expected by its application on the training set and we need to consider other metrics too, in order to decide if we are to considere this model reliable. One would say that the main ability of this model is to fetch actual cases of eyes being open or shut, but it does so by bringing in many false positive cases. Again the kappa score though shows weak level of agreement between predicted and true values. Same goes for MCC wich further proves the low level of agreement between the predicted and the true values. It is also important to notice that since the dataset is even slightly imbalanced, F1-score is of greater importance than Recall or Precision taking into account both of them.

3.3 Transforming data using PCA

Sometimes performing a PCA on the data and then applying a model helps in obtaining better results mainly due to PCA de-noising the data. It finds a low-dimensional representation of a data set that contains as much as possible of the variation. So here PCA is perormed on the data before the SVM model is applied.

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5    PC6     PC7
## Standard deviation     2.331 1.9649 1.8113 0.91104 0.55419 0.3261 0.28043
## Proportion of Variance 0.388 0.2758 0.2344 0.05929 0.02194 0.0076 0.00562
## Cumulative Proportion  0.388 0.6637 0.8981 0.95735 0.97929 0.9869 0.99250
##                           PC8     PC9    PC10    PC11     PC12     PC13
## Standard deviation     0.20917 0.17337 0.14718 0.09716 0.007361 0.003537
## Proportion of Variance 0.00313 0.00215 0.00155 0.00067 0.000000 0.000000
## Cumulative Proportion  0.99563 0.99777 0.99932 1.00000 1.000000 1.000000
##                           PC14
## Standard deviation     0.001395
## Proportion of Variance 0.000000
## Cumulative Proportion  1.000000
```

The graph shows our class of interest through the first two main principal components and how the variables are accumulated within these two components.

Now the test set must be transformed into PCA format.

```
### transform test set into PCA ###
set.seed(1234)
pca.test_X0 <- predict(pca.train_X0, newdata = test_X0)

summary(pca.test_X0)
```

```
##       PC1                 PC2                 PC3                 PC4
##  Min.   :-1.45967   Min.   :-1.53025   Min.   :-1.59796   Min.   :-1.85684
##  1st Qu.:-0.14180   1st Qu.:-0.14710   1st Qu.:-0.37085   1st Qu.:-0.52639
##  Median : 0.03592   Median : 0.02824   Median :-0.12920   Median :-0.16090
##  Mean   : 0.00317   Mean   : 0.01746   Mean   :-0.05565   Mean   : 0.00443
##  3rd Qu.: 0.20236   3rd Qu.: 0.20928   3rd Qu.: 0.14144   3rd Qu.: 0.31179
##  Max.   : 0.90101   Max.   : 1.30040   Max.   : 3.68746   Max.   : 5.84560
##       PC5                 PC6                  PC7                 PC8
##  Min.   :-2.28569   Min.   :-1.319775   Min.   :-0.95794   Min.   :-0.848954
##  1st Qu.:-0.28221   1st Qu.:-0.180561   1st Qu.:-0.14951   1st Qu.:-0.144729
##  Median : 0.01596   Median :-0.003668   Median : 0.03010   Median : 0.002672
##  Mean   :-0.02744   Mean   :-0.003183   Mean   : 0.01932   Mean   :-0.004870
##  3rd Qu.: 0.28271   3rd Qu.: 0.175047   3rd Qu.: 0.21055   3rd Qu.: 0.138602
##  Max.   : 2.97421   Max.   : 1.680066   Max.   : 0.76637   Max.   : 0.622079
##       PC9                 PC10                PC11
```

```
##  Min.   :-0.5893972   Min.   :-0.602361   Min.   :-0.470627
##  1st Qu.:-0.1163140   1st Qu.:-0.098639   1st Qu.:-0.064291
##  Median :-0.0008015   Median : 0.000020   Median :-0.005425
##  Mean   :-0.0008665   Mean   :-0.004753   Mean   :-0.005319
##  3rd Qu.: 0.1063958   3rd Qu.: 0.089607   3rd Qu.: 0.052696
##  Max.   : 0.7651365   Max.   : 0.490010   Max.   : 0.343357
##       PC12                 PC13                 PC14
##  Min.   :-2.475e-02   Min.   :-1.244e-02   Min.   :-4.949e-03
##  1st Qu.:-4.639e-03   1st Qu.:-2.276e-03   1st Qu.:-8.944e-04
##  Median :-3.144e-04   Median : 2.317e-05   Median :-7.246e-05
##  Mean   : 2.289e-05   Mean   :-5.839e-05   Mean   :-1.587e-05
##  3rd Qu.: 4.277e-03   3rd Qu.: 2.373e-03   3rd Qu.: 7.786e-04
##  Max.   : 3.587e-02   Max.   : 1.138e-02   Max.   : 7.125e-03
```

Above we can see some basic statistics of the principal components our test set has been transformed into.

3.4 Building model on PCA transformed data using SVM algorithm

Since the PCA transforms the dataset into a new one with different properties, the cross validation to find out the best model needs to rerun.

```
# tuning basically performs via the sampling method: 10-fold cross validation
tune.out2 <- tune(svm, eyeDetect~. , data = train.data, scale =TRUE,
                  ranges = list(cost = c(1,10),
                                kernel = c("linear","radial", "polynomial"),
                                gamma =c(0.1, 0.01)))



#tune.out2$sampling
#tune.out2$performances
#tune.out2$best.parameters
#tune.out2$best.model
```

coat	kernel	gamma	error	dispersion
1	linear	0.10	0.35599876	0.01868290
10	linear	0.10	0.35617638	0.01801221
1	radial	0.10	0.09949950	0.01560914
10*	radial*	0.10*	0.06425549	0.01057877
1	polynomial	0.10	0.17265908	0.01822809
10	polynomial	0.10	0.15201217	0.01673010
1	linear	0.01	0.35599876	0.01868290
10	linear	0.01	0.35617638	0.01801221
1	radial	0.01	0.23637664	0.02657034
10	radial	0.01	0.17870383	0.02202524

Interestingly enough, the best model is once again the one with kernel= “radial”, cost=10 and gamma=0.1.

We apply the best model on the PCA training test.

```r
set.seed(100)
svm2 <- svm(eyeDetect ~ ., data=train.data,
          method="C-classification", kernal="radial", scale =TRUE,
          gamma=0.1, cost=10)


summary(svm2)
```

```
##
## Call:
## svm(formula = eyeDetect ~ ., data = train.data, method = "C-classification",
##     kernal = "radial", gamma = 0.1, cost = 10, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  1596
##
##  ( 815 781 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```r
prediction2 <- predict(svm2, train.data)

cmatrixPCAtr<-confusionMatrix(prediction2, train.data$eyeDetect, dnn = c("Prediction", "Reference"))
PCAmcc2_trSVM <- mccr(train.data$eyeDetect, prediction2)

#cmatrixPCAtr$byClass
```

Accuracy	Sensitivity/Recall	Specificity	Precision	F1-score	Kappa	MCC
0.9692	0.9765	0.9603	0.9681	0.7014	0.9377	0.9377

Accuracy is very high (~97%) with p-vlaue<0.05 which supports the significance of predictions made by this model. Sensitivity/recall (97.7%), specificity(96%) and Precision (96.8%) are all very high which is quite promising. Though since the dataset is imbalanced we must take into consideration F1-score which shows moderate to good performance of precision and recall. Kappa score shows that the predicted and the true values are in excellent agreement. MCC is very high too wich further proves the excellent agreement between the predicted and the true values. It seems that PCA helps in the direction of producing a really good model.

We now go on and apply the model on the test set.

```r
###SVM on PCA test data###
# create PCA test set with class of interest
test.data <- data.frame(eyeDetect = test_X0$eyeDetect, pca.test_X0)
```

```r
predicted_test2 <- predict(svm2, test.data)

pcatestconfm <- confusionMatrix(predicted_test2, test.data$eyeDetect,
                                dnn = c("Prediction", "Reference"))

PCAmcc2_teSVM <- mccr(test.data$eyeDetect, predicted_test2)
#pcatestconfm$byClass
```

Accuracy	Sensitivity/Recall	Specificity	Precision	F1-score	Kappa	MCC
0.9407	0.9558	0.9228	0.9364	0.9460	0.8803	0.8805

When applied on the PCA transformed test set accuracy drops lightly (~94%). p-value <0.05 at confidence interval of 95% shows that the results obtained are of high statistical importance. Sensitivity/recall (95.6%) and specificity (92.3%) are both high. F1-score is excellent too though the agreement of predicted and true values drops lightly compared to training set, but is still considered quite good and it can be further verified by MCC.

It is quite clear that the model carries its prediction robustness from the training set to the test set, which is very important to be considered of actual value. PCA seems to play an important role in finally producing a model much more robust in predicting on unknown data.

3.5 Model building using Logistic Regression

We go on using another classification algorithm-logistic regression.

The approach followed was to find the best predictors to be used for the most robust model to be created. For this reason backward elimination was followed.

```r
# Backward elimination was followed in order to find the best predictors
# for the logistic regression model to be built

fullmodel <- glm(eyeDetection ~., data = train_X0, family="binomial")
step(fullmodel, direction = "backward", trace=FALSE )
```

```
##
## Call:  glm(formula = eyeDetection ~ F7 + F3 + FC5 + T7 + P7 + P8 + FC6 +
##     AF4, family = "binomial", data = train_X0)
##
## Coefficients:
## (Intercept)           F7           F3          FC5           T7           P7
##    7.823334    -0.018761     0.017358    -0.010337     0.039649    -0.041960
##          P8          FC6          AF4
##    0.011607    -0.009765     0.011409
##
## Degrees of Freedom: 5617 Total (i.e. Null);  5609 Residual
## Null Deviance:       7727
## Residual Deviance: 7177  AIC: 7195
```

```r
fullmodel$coefficients
```

```
##  (Intercept)          AF3           F7           F3          FC5           T7
##  8.356387406  0.002771584 -0.019426145  0.016080066 -0.010703426  0.039271555
##           P7           O1           O2           P8           T8          FC6
## -0.042751221  0.002192696 -0.002802370  0.011050796  0.003101994 -0.009405073
##           F4           F8          AF4
##  0.003975881 -0.002613741  0.008931024
```

```
lg_par <- drop1(fullmodel, test = "F")
```

Predictor	Df	Deviance	AIC	F value	Pr(>F)
AF3	1	7172	7200	5.5200e-01	0.457528
F7	1	7331	7359	1.2447e+02	<2.2e-16 ***
F3	1	7194	7222	1.7463e+01	2.974e-05 ***
FC5	1	7185	7213	1.0643e+01	0.001111 **
T7	1	7272	7300	7.8495e+01	<2.2e-16 ***
P7	1	7346	7374	1.3627e+02	2.2e-16 ***
O1	1	7172	7200	7.8460e-01	0.375765
O2	1	7172	7200	4.9900e-01	0.479961
P8	1	7179	7207	5.5182e+00	0.018854 *
T8	1	7172	7200	6.0340e-01	0.437300
FC6	1	7183	7211	9.0134e+00	0.002692 **
F4	1	7172	7200	5.8450e-01	0.444593
F8	1	7173	7173	1.0168e+00	0.313332
AF4	1	186634	186634	1.4021e+05	<2.2e-16 ***

The backward elimination process shows that according to the F-test the best predictors are F7, F3, FC5, T7, P7, P8, FC6, AF4, marked here with an asterisk gradient. We go on and build a logistic regression model using these predictors.

```
new_trX0 <- train_X0[,-c(1,7,8,10,12,13)]
set.seed(100)
best_logmodel <- glm(eyeDetection ~ ., data = new_trX0, family="binomial" )


summary(best_logmodel)
```

```
##
## Call:
## glm(formula = eyeDetection ~ ., family = "binomial", data = new_trX0)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1622  -1.0453  -0.7246   1.1466   2.1694
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  7.823334   7.952392   0.984 0.325228
## F7          -0.018761   0.001545 -12.144  < 2e-16 ***
## F3           0.017358   0.002950   5.884 4.00e-09 ***
## FC5         -0.010337   0.002828  -3.655 0.000257 ***
## T7           0.039649   0.003728  10.636  < 2e-16 ***
```

```
## P7          -0.041960   0.003368 -12.460  < 2e-16 ***
## P8           0.011607   0.002766   4.196 2.71e-05 ***
## FC6         -0.009765   0.002403  -4.063 4.84e-05 ***
## AF4          0.011409   0.001535   7.432 1.07e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7727.0  on 5617  degrees of freedom
## Residual deviance: 7177.1  on 5609  degrees of freedom
## AIC: 7195.1
##
## Number of Fisher Scoring iterations: 10
```

We see here a summary of the trained model upon the new training set which includes only the variables shown as important by the backward elimination process. All variabels get the highest of markings(*) on importance.

```
glm.probs <- predict(best_logmodel, new_trX0, type='response')
glm.probs[1:10]
```

```
##      6478      1408      1084      6531      5076      3491      3318      5830
## 0.2510320 0.4926671 0.4693396 0.6615141 0.4390192 0.2676405 0.3248899 0.3512650
##       332      1064
## 0.1983403 0.3503011
```

We se here an example of the probabilities projection of the first 10 elements in the training set.

```
glm.predictions <- rep("0", nrow(new_trX0))

glm.predictions[glm.probs>0.5] <-"1"

tableBest <- table(glm.predictions, new_trX0$eyeDetection)

resultsBest <- confusionMatrix(tableBest , positive = '0')

mcc3_trLR <- mccr(train_X0$eyeDetection, glm.predictions)

#precision(tableBest)
```

Accuracy	Sensitivity/Recall	Specificity	Precision	F1-score	Kappa	MCC
0.6424	0.7605	0.4968	0.6507	0.7013	0.2625	0.2674

As we can see accuracy is at 64.2%, Sensitivity/Recall is at 76% and Specificity is at 49,7%, with a p-value<0.05 at 95% confidence interval which shows that results obtained are of statistical importance. The model is not doing so well in terms of accuracy, but does a better job in fetching positive results for the class of interest. Given that the dataset is slightly imbalanced we need to rely on the F1-score more which shows moderate to somewhat good performance though kappa is extremmely low to the point were maximum 15% of the data could be considered reliable for predictions on it. The vey low MCC score further shows that

the predicted results are not in good agreement with the true values.

Logistic regression has no hyperparameters to tune over; the estimates for the coefficients will always be given by maximum likelihood. The repeated k-fold cross validation will do nothing to affect the estimates of the parameters/coefficients. Therefore cross-validation is redundant and backward elimination is adequete to yield a better model either by giving out better accuracy, sensitivity or specificity metrics or simply lowering p-value and rise kappa score, MCC and F1-score.

Next, the trained model is applied on the test set. The test set, just like the training set needs to be redeemed of the unimportant variables.

```
new_testX0 <- test_X0[,-c(1,7,8,10,12,13)]


TeprobBest = predict(best_logmodel, new_testX0 , type="response")

glm.predictions_testBest <- rep("0", nrow(new_testX0 ))
# replace "No" by "Yes" where the probabiliy obtained is > 0.5
glm.predictions_testBest[TeprobBest>0.5] <-"1"

table2Best <- table(glm.predictions_testBest, new_testX0$eyeDetection)

results3Best <- confusionMatrix(table2Best, positive = '0')
print(results3Best)
```

```
## Confusion Matrix and Statistics
##
##
## glm.predictions_testBest   0   1
##                        0 782 446
##                        1 235 409
##
##                Accuracy : 0.6362
##                  95% CI : (0.614, 0.6581)
##     No Information Rate : 0.5433
##     P-Value [Acc > NIR] : 2.379e-16
##
##                   Kappa : 0.2522
##
##  Mcnemar's Test P-Value : 8.470e-16
##
##             Sensitivity : 0.7689
##             Specificity : 0.4784
##          Pos Pred Value : 0.6368
##          Neg Pred Value : 0.6351
##              Prevalence : 0.5433
##          Detection Rate : 0.4177
##    Detection Prevalence : 0.6560
##       Balanced Accuracy : 0.6236
##
##        'Positive' Class : 0
##
```

```r
mcc3_teLR <- mccr(test_X0$eyeDetection, glm.predictions_testBest)

#precision(table2Best)
#F1_Score(glm.predictions, new_trX0$eyeDetection, positive = "0")
#results3Best$byClass
```

Accuracy	Sensitivity/Recall	Specificity	Precision	F1-score	Kappa	MCC
0.6362	0.7689	0.4784	0.6560	0.6967	0.2522	0.2548

No significant change in accuracy despite backward elimination process, though less data is used. The model manages an accuracy of 63.6%, Sensitivity/Recall of 76.9% and Specificity of 47.8% with a p-value<0.05 at 95% confidence interval when applied on the test set. It is easy to see that the results of the test set follow those of the training set, one could say that this is further proof that the model built and trained is trustworthy but with low kappa score meaning that there is low agreement between predicted and true values. MCC score adds further proof to this point.

4. PART-4 RESULTS-DISCUSSION

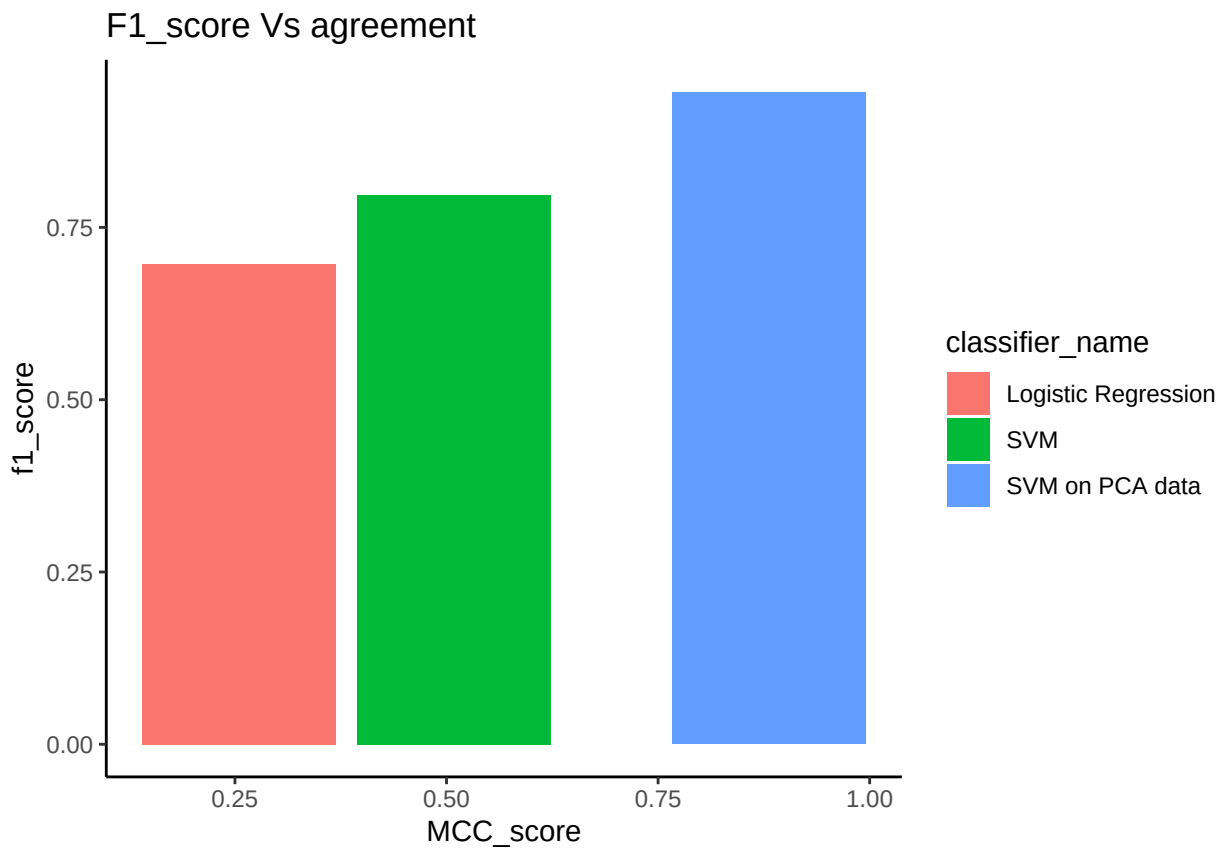
In order to sum up the work done we need to compare the separate classifiers built by how they perform on the test set, because this is what matters, the predictions over the unseen data. Here follows a table with the appropriate information.

Classifier	Accuracy	Sensitivity/Recall	Specificity	Precision	F1-score	Kappa	MCC
SVM	0.753	0.891	0.590	0.721	0.797	0.491	0.509
SVM on PCA data	0.940	0.956	0.923	0.936	0.946	0.880	0.881
Logistic Regression	0.636	0.769	0.478	0.656	0.697	0.252	0.255

```r
classifier_name <- c("SVM", "SVM on PCA data", "Logistic Regression" )
f1_score <- c(0.797, 0.946, 0.697)
MCC_score <- c(0.509, 0.881, 0.255)

df_final <- data.frame(classifier_name, f1_score, MCC_score)

theme_set(theme_classic())
ggplot(df_final, aes(MCC_score, f1_score)) +
  geom_bar(mapping = aes(fill = classifier_name),
           stat="identity", position = "dodge")+
  ggtitle("F1_score Vs agreement")
```



It is easy for everyone to see that the best classifier is SVM on PCA transformed data. All metrics are way more superior for this classifier and kappa score is pretty high too, meaning that the predictions and the true values of the test data are in good agreement, a conclusion further assisted by the high MCC score. As it can be seen on the graph, where F1_score and MCC are depicted since as mentioned before this is a slightly imbalanced dataset therefore these two metrics may be of better use, SVM on PCA data is superior. PCA has the ability to sort of de-noise data and in this way could help in increasing the ability of classifiers built, like in this case.

So the PCA performed may explain why this model is superior to just applying SVM on the non-transformed, original data, but it seems that maybe SVM here builds on the already higher performance SVM shows when applied to the original data compared to when Logistic Regression is applied on it. So we need to also examine the difference between SVM and Logistic regression.

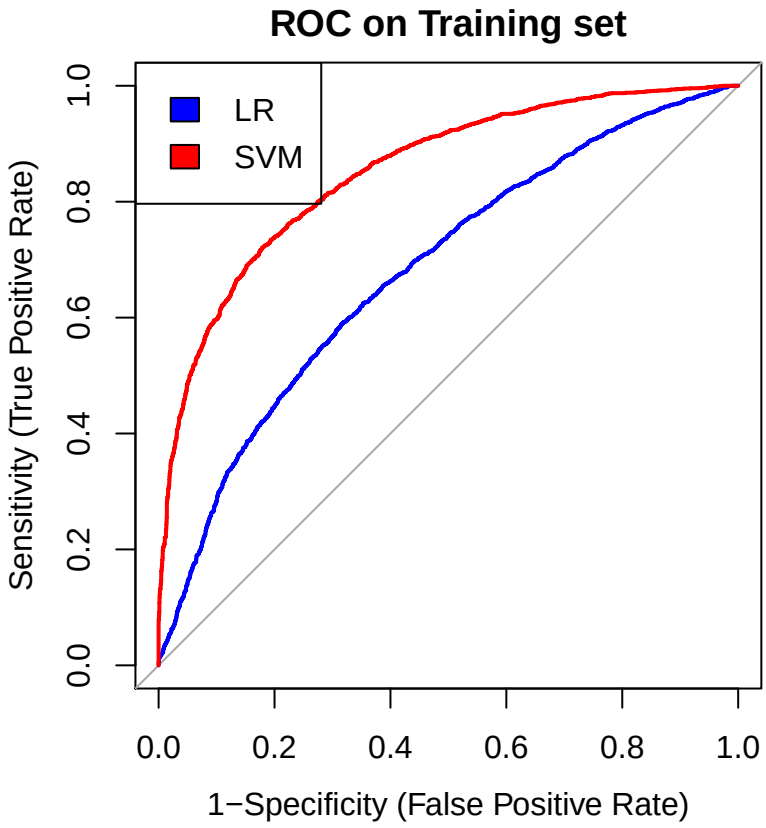
For starters, all metrics on the application of the models on the test set are higher, backed by double the kappa and MCC scores giving much better agreement between predicted and true values. Given that the data is slightly imbalanced we could go on and apply the metric of ROC too.

```
par(pty="s")
LRrocTtr <- roc(new_trX0$eyeDetection,best_logmodel$fitted.values,
                legacy.axes=TRUE, auc = TRUE, plot = TRUE,
                ylab="Sensitivity (True Positive Rate)",
                xlab="1-Specificity (False Positive Rate)",
                col="blue", main="ROC on Training set")
#plot(LRrocTtr)

SVMrocTtr <- roc(train_X0$eyeDetection,svm1$decision.values,
                 legacy.axes=TRUE, auc = TRUE, plot = TRUE,
```

```
                ylab="Sensitivity (True Positive Rate)",
                xlab="1-Specificity (False Positive Rate)",
                col="red", add=TRUE)

legend("topleft", c("LR","SVM"),
fill=c("blue", "red"))
```

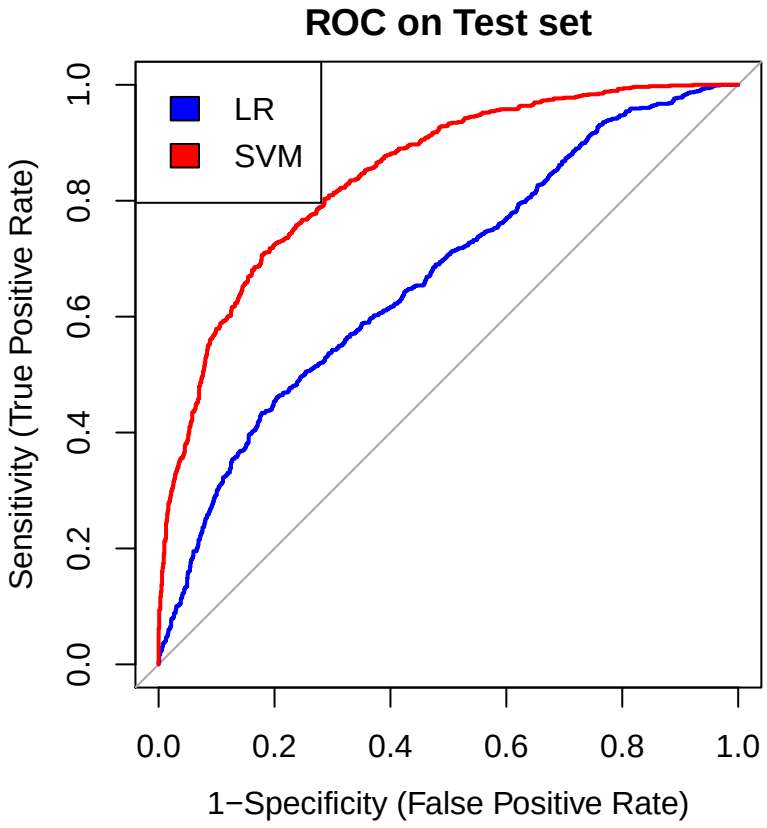



```
# get the predicted probabilities for the test set for SVM
probssvmte = attr(predicted_test,"probabilities")[,1]

par(pty="s")
LRrocTte <- roc(new_testXO$eyeDetection,TeprobBest, legacy.axes=TRUE,
                auc = TRUE, plot = TRUE,
                ylab="Sensitivity (True Positive Rate)",
                xlab="1-Specificity (False Positive Rate)",
                col="blue", main="ROC on Test set")

SVMrocTte <- roc(test_XO$eyeDetection,probssvmte, legacy.axes=TRUE,
                 auc = TRUE, plot = TRUE,
                ylab="Sensitivity (True Positive Rate)",
                xlab="1-Specificity (False Positive Rate)",
                col="red", add=TRUE)

legend("topleft", c("LR","SVM"),
fill=c("blue", "red"))
```




```
LRtrAUC <- LRrocTtr$auc
SVMtrAUC <- SVMrocTtr$auc
LRteAUC <- LRrocTte$auc
SVMteAUC <- SVMrocTte$auc
```

LR AUC-training set	SVM AUC-training set	LR AUC-test set	SVM AUC-test set
0.6818	0.8519	0.6694	0.845

Regarding ROC, classifiers that give curves closer to the top-left corner indicate a better performance. We can see this for SVM when ROC is applied on the training set. This pattern is carried on the test set, with the graphs clearly showing the better performance the SVM classifier yields.

To compare different classifiers, it can be useful to summarize the performance of each classifier into a single measure. One common approach is to calculate the area under the ROC curve, which is abbreviated to AUC. It is equivalent to the probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative instance. Here it can be clearly seen that both for the training and the test set AUC is much higher for the SVM model.

So one could say that the higher performance of SVM on PCA transformed data carries on from the already higher performance of SVM on the original data compared to Logistic Regression.

SVM tries to finds the "best" margin (distance between the line and the support vectors) that separates the classes and this reduces the risk of error on the data [4], while logistic regression does not, instead it can have different decision boundaries with different weights that are near the optimal point. The SVM algorithm creates a hyperplane or line (decision boundary) which separates data into classes. It uses the kernel trick

to find the best line separator (decision boundary that has same distance from the boundary point of both classes). It is a clear and more powerful way of learning complex non linear functions. So perhaps its higher ability lies exactly in the type of data it is here applied too.

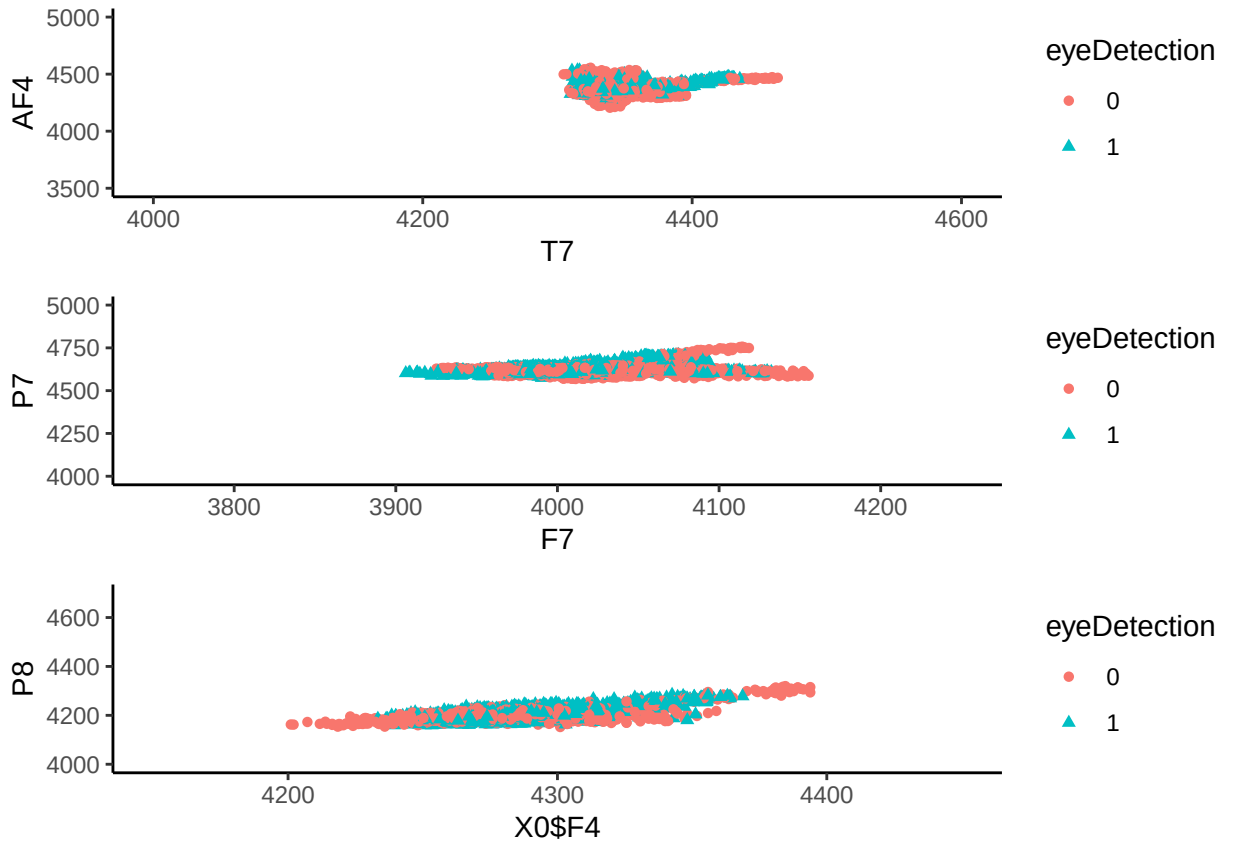
We go on and plot some variables of the data existing in both models.

```
R1 <- ggplot(X0, aes(x=T7, y=AF4, shape=eyeDetection, color=eyeDetection)) + geom_point()+
  xlim(c(4000, 4600))+
  ylim(c(3500,5000))

R2 <- ggplot(X0, aes(x=F7, y=P7, shape=eyeDetection, color=eyeDetection)) + geom_point()+
  xlim(c(3750, 4250))+
  ylim(c(4000,5000))

R3 <- ggplot(X0, aes(x=X0$F4, y=P8, shape=eyeDetection, color=eyeDetection)) + geom_point()+
  xlim(c(4150, 4450))+
  ylim(c(4000,4700))

grid.arrange(R1,R2,R3)
```



Looking at these indicative plots of the original data we see exactly this unstructureness, non linearity of the data that make SVM a much more succesfull algorithm to build a predictive model upon compared to logistic regression. It is this ability of SVM to find boundaries on geometrical properties rather than be based on statistical approaches, like logistic regression does, that makes it a better choice for this project.

5. PART-5 FUTURE CHALLENGES

Through logistic regression backward elimination process we saw that some predictors were removed. Perhaps we could apply the same strategy on the SVM model to see if it would yield better results. Another way to go at this project would be by building a classifier based on Decision Trees or Random Forest and compare the results of the same metrics with the ones presented here. Balancing the dataset could be one approach, though it could also end up being catastrophic, completely hiding the moment of interest where peaks of different magnitude for eyes-open and eyes-closed states occur, so algorithms like SMOTE may be tried out. One interesting approach would be to change the ratio of training to test set split to see if we can obtain the same or perhaps better results with less training data. This could save up time, processing power and perhaps prevent any chance of overfitting that may occur.

PART-6 LITERATURE

[1] Applications of Image Processing and Soft Computing Systems in Agriculture Razmjooy, NavidEstrela, Vania Vieira, 2019, IGI Global

[2] GIS and Geocomputation for Water Resource Science and Engineering, Barnali Dixon, Venkatesh Uddameri, 2016, Wiley

[3] Principal component analysis: a review and recent developments, Ian T. Jolliffe1 and Jorge Cadima2,3, Philos Trans A Math Phys Eng Sci. 2016 Apr 13; 374(2065): 20150202.

[4] The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation, December 2020, BMC Genomics 21(1)

[5] An Introduction to Statistical Learning with Applications in R, Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, 2017, Springer

[6] Maximum Likelihood Estimation of Logistic Regression Models: Theory and Implementation Scott A. Czepiel, 2002

--THE END--