ASSIGNMENT-2

A Report Submitted as a part of Machine learning Course

By

Shrihari Viswanath-2016A8PS0396P

Akshit Achara-2016B4PS0953P

Nitesh Kumar-2016A4PS0276P

To

Dr. Navneet Goyal

Department of Computer Science and Engineering

BITS Pilani, Pilani

Rajasthan, India

First Semester 2018-19

Problem Statement

A student grading data was given to us by Dr. Navneet Goyal, from a course taken by him a few years ago. It contained the following data- IDNO, YEAR, ATTENDANCE %, CGPA, MidSemester, MidSemester Grades, MidSemester Collection, Quiz1, Quiz2, Part A, Part B and Grade. The aim was to build various classification methods on R, to train them and to test and compare performances. We also had to frame a few questions by analysing the given data.

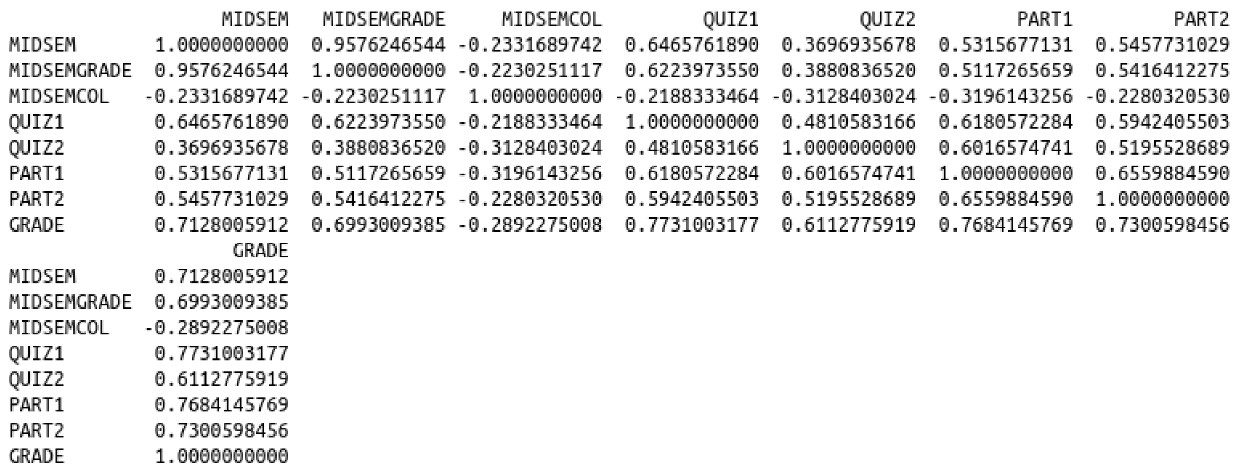
PREPROCESSING OF DATA

Firstly, all the types of data and their contribution to the observed output was noted. All the categorical variables had to be converted into numbers for any algorithms to be able to process them. We then replaced the grades with their corresponding numerical values (B-> 8, C->6, ………). NC, I and W were replaced with the number zero.

Also, it is evident that the attribute 'IDNO' doesn't affect the output in any way. In order to study the relation of various attributes with GRADES we calculate the correlation matrices of all attributes corresponding with 'GRADES'.

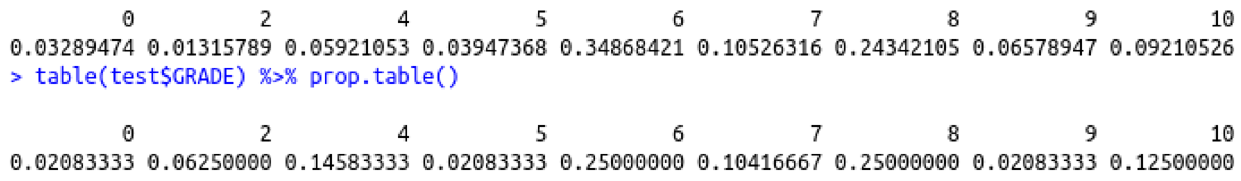
First of all we remove 'CGPA' from the data as it has only 40 values despite having a high correlation value. Considering a correlation threshold value of 0.3 we eliminate the unwanted variables. M/F', 'Year' and 'Attendance' have been eliminated due to their correlation values. In the end we have seven attributes in total which will be used.

Given below are the correlation values obtained:



```
                  MIDSEM   MIDSEMGRADE     MIDSEMCOL         QUIZ1         QUIZ2         PART1         PART2
MIDSEM       1.0000000000  0.9576246544 -0.2331689742  0.6465761890  0.3696935678  0.5315677131  0.5457731029
MIDSEMGRADE  0.9576246544  1.0000000000 -0.2230251117  0.6223973550  0.3880836520  0.5117265659  0.5416412275
MIDSEMCOL   -0.2331689742 -0.2230251117  1.0000000000 -0.2188333464 -0.3128403024 -0.3196143256 -0.2280320530
QUIZ1        0.6465761890  0.6223973550 -0.2188333464  1.0000000000  0.4810583166  0.6180572284  0.5942405503
QUIZ2        0.3696935678  0.3880836520 -0.3128403024  0.4810583166  1.0000000000  0.6016574741  0.5195528689
PART1        0.5315677131  0.5117265659 -0.3196143256  0.6180572284  0.6016574741  1.0000000000  0.6559884590
PART2        0.5457731029  0.5416412275 -0.2280320530  0.5942405503  0.5195528689  0.6559884590  1.0000000000
GRADE        0.7128005912  0.6993009385 -0.2892275008  0.7731003177  0.6112775919  0.7684145769  0.7300598456
                   GRADE
MIDSEM       0.7128005912
MIDSEMGRADE  0.6993009385
MIDSEMCOL   -0.2892275008
QUIZ1        0.7731003177
QUIZ2        0.6112775919
PART1        0.7684145769
PART2        0.7300598456
GRADE        1.0000000000
```

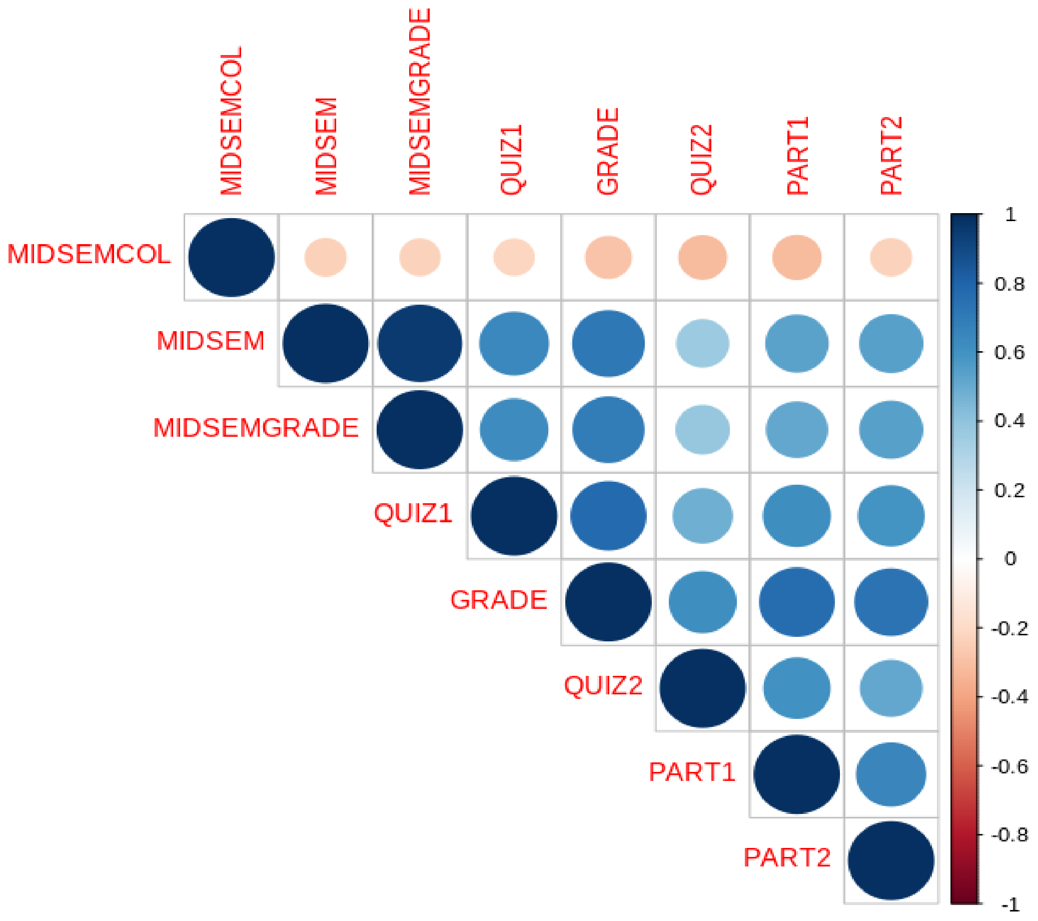
Probabilities of data:



```
         0          2          4          5          6          7          8          9         10 
0.03289474 0.01315789 0.05921053 0.03947368 0.34868421 0.10526316 0.24342105 0.06578947 0.09210526 
> table(test$GRADE) %>% prop.table()

         0          2          4          5          6          7          8          9         10 
0.02083333 0.06250000 0.14583333 0.02083333 0.25000000 0.10416667 0.25000000 0.02083333 0.12500000 
```

Heat Map for the data is as follows:



MIDSEMCOL
MIDSEM
MIDSEMGRADE
QUIZ1
GRADE
QUIZ2
PART1
PART2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1

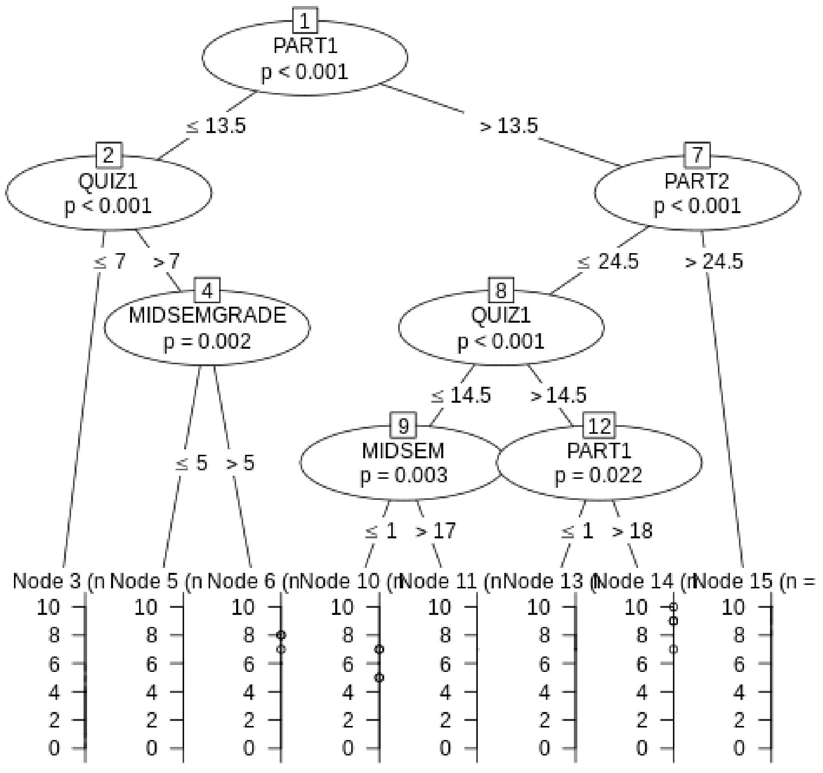
CLASSIFICATION OF DATA

1. Decision Tree

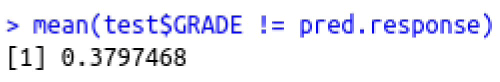
   Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

   A tree can be learned by splitting the source set into subsets based on an attribute value test. We can chose the best attributes to design the tree using information gain. In order to define information gain precisely, we begin by defining a measure commonly used in information theory, called entropy that characterizes the (im)purity of an arbitrary collection of examples.

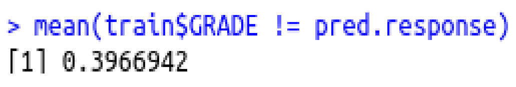
   A decision tree was programmed on R and using the library 'rpart' we split the data into 75% training and 25% test data. We trained the tree using the data and we collected the test and training errors using the data sets we have split. We can evaluate performance using these.



The test and training accuracies of the decision tree implemented are as follows:

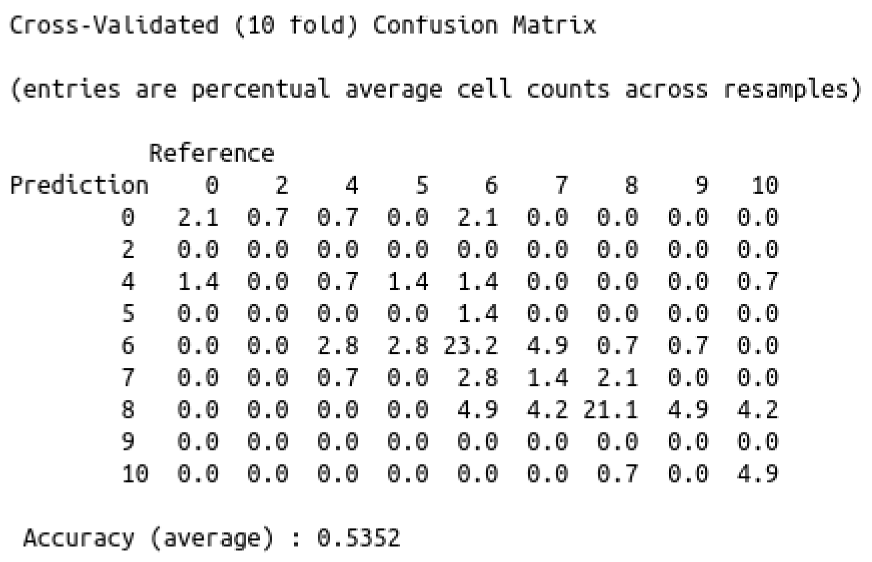


```
> mean(test$GRADE != pred.response)
[1] 0.3797468
```



```
> mean(train$GRADE != pred.response)
[1] 0.3966942
```

Decision tree accuracies and predictions, before and after PCA respectively :

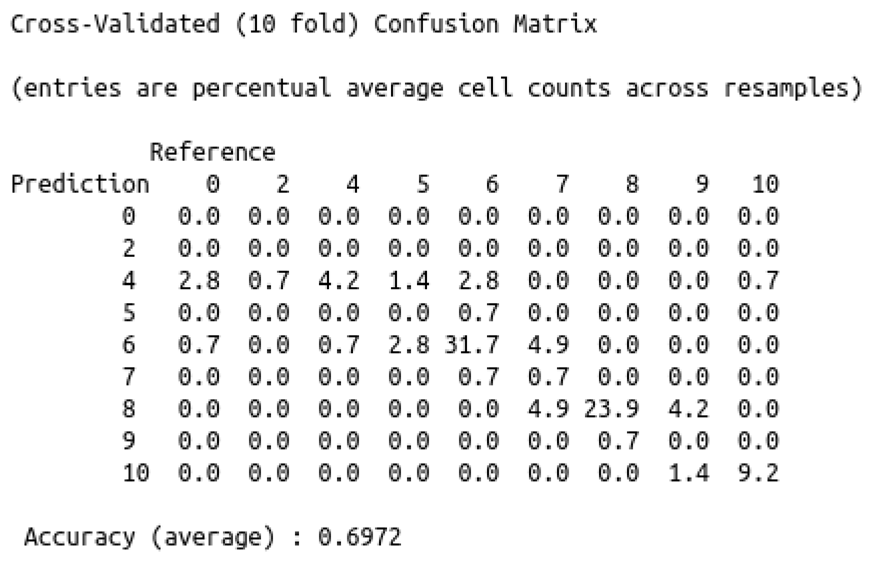


```
Cross-Validated (10 fold) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction    0    2    4    5    6    7    8    9   10
        0   2.1  0.7  0.7  0.0  2.1  0.0  0.0  0.0  0.0
        2   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
        4   1.4  0.0  0.7  1.4  1.4  0.0  0.0  0.0  0.7
        5   0.0  0.0  0.0  0.0  1.4  0.0  0.0  0.0  0.0
        6   0.0  0.0  2.8  2.8 23.2  4.9  0.7  0.7  0.0
        7   0.0  0.0  0.7  0.0  2.8  1.4  2.1  0.0  0.0
        8   0.0  0.0  0.0  0.0  4.9  4.2 21.1  4.9  4.2
        9   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
        10  0.0  0.0  0.0  0.0  0.0  0.0  0.7  0.0  4.9

 Accuracy (average) : 0.5352
```



```
Cross-Validated (10 fold) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction    0    2    4    5    6    7    8    9   10
        0   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
        2   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
        4   2.8  0.7  4.2  1.4  2.8  0.0  0.0  0.0  0.7
        5   0.0  0.0  0.0  0.0  0.7  0.0  0.0  0.0  0.0
        6   0.7  0.0  0.7  2.8 31.7  4.9  0.0  0.0  0.0
        7   0.0  0.0  0.0  0.0  0.7  0.7  0.0  0.0  0.0
        8   0.0  0.0  0.0  0.0  0.0  4.9 23.9  4.2  0.0
        9   0.0  0.0  0.0  0.0  0.0  0.0  0.7  0.0  0.0
        10  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.4  9.2

 Accuracy (average) : 0.6972
```

2. Naïve Bayes Classifier

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

The dataset is divided into two parts, namely, feature matrix and the response vector. Feature matrix contains all the vectors (rows) of dataset in which each vector consists of the value of dependent features. Response vector contains the value of class variable (prediction or output) for each row of feature matrix.

We assume that no pair of features are dependent. For example, the temperature being 'Hot' has nothing to do with the humidity or the outlook being 'Rainy' has no effect on the winds. Hence, the features are assumed to be independent. Secondly, each feature is given the same weight (or importance). For example, knowing only temperature and humidity alone can't predict the outcome accurately. None of the attributes is irrelevant and assumed to be contributing equally to the outcome.

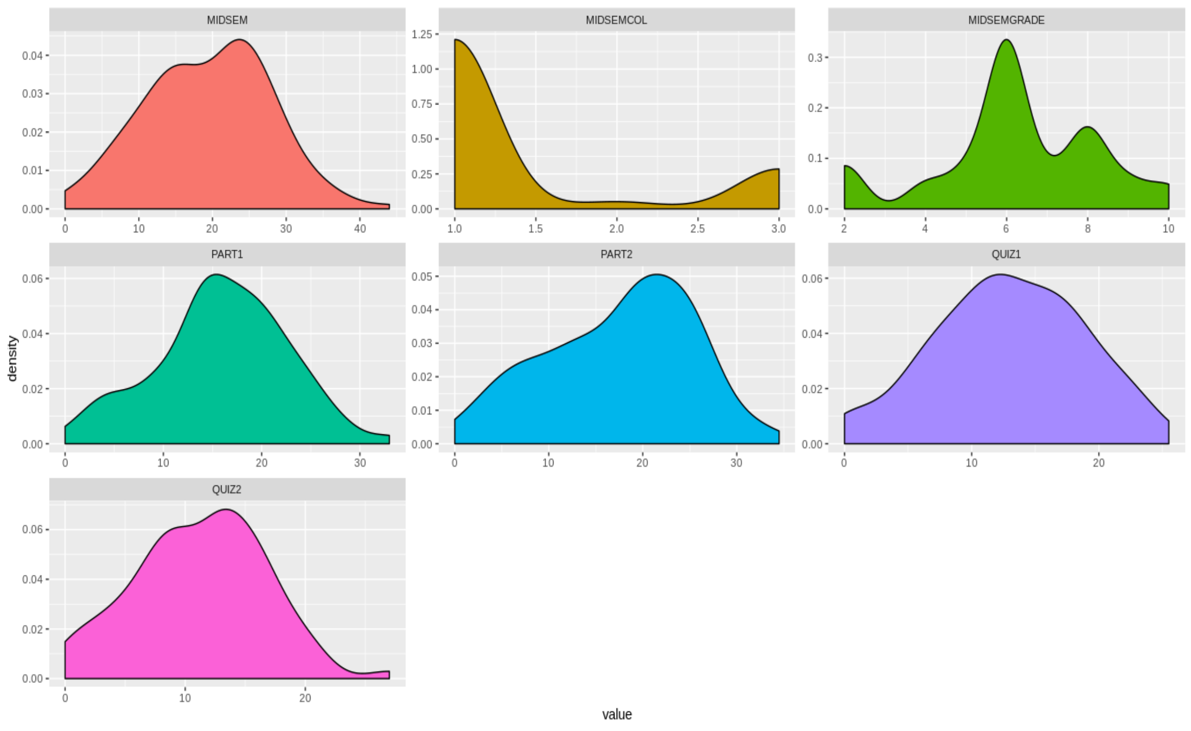
$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

P(A|B) : the conditional probability that event A occurs , given that B has occurred. This is also known as the posterior probability.

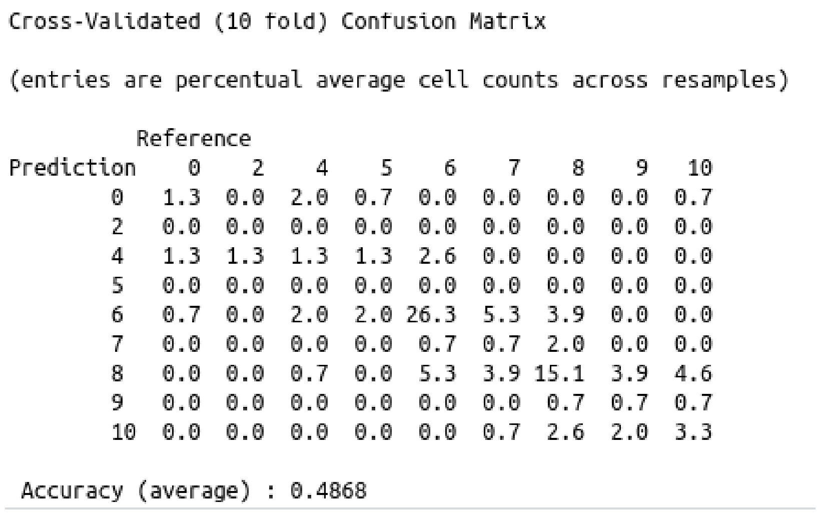
P(A) and P(B) : probability of A and B without regard of each other.

P(B|A) : the conditional probability that event B occurs , given that A has occurred.

Probability densities of data:



Naïve Bayes classifier accuracies and predictions, before and after PCA respectively :

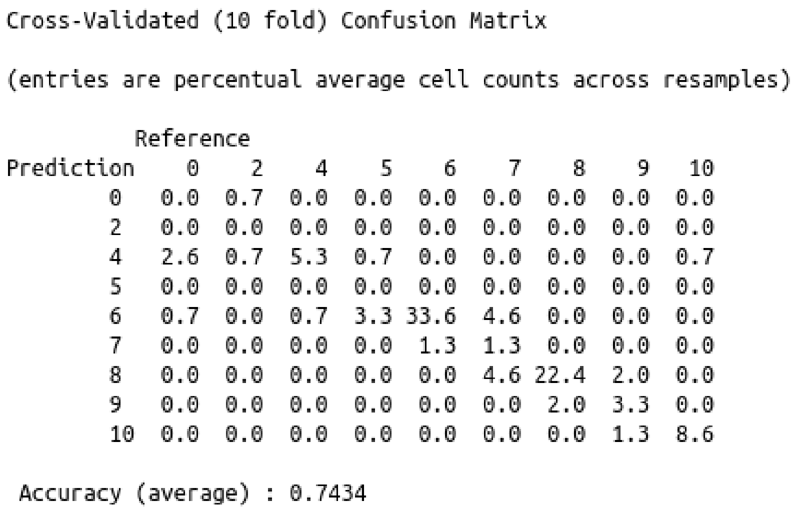


```
Cross-Validated (10 fold) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction    0    2    4    5    6    7    8    9   10
        0   1.3  0.0  2.0  0.7  0.0  0.0  0.0  0.0  0.7
        2   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
        4   1.3  1.3  1.3  1.3  2.6  0.0  0.0  0.0  0.0
        5   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
        6   0.7  0.0  2.0  2.0 26.3  5.3  3.9  0.0  0.0
        7   0.0  0.0  0.0  0.0  0.7  0.7  2.0  0.0  0.0
        8   0.0  0.0  0.7  0.0  5.3  3.9 15.1  3.9  4.6
        9   0.0  0.0  0.0  0.0  0.0  0.0  0.7  0.7  0.7
        10  0.0  0.0  0.0  0.0  0.0  0.7  2.6  2.0  3.3

 Accuracy (average) : 0.4868
```



```
Cross-Validated (10 fold) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction    0    2    4    5    6    7    8    9   10
        0   0.0  0.7  0.0  0.0  0.0  0.0  0.0  0.0  0.0
        2   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
        4   2.6  0.7  5.3  0.7  0.0  0.0  0.0  0.0  0.7
        5   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
        6   0.7  0.0  0.7  3.3 33.6  4.6  0.0  0.0  0.0
        7   0.0  0.0  0.0  0.0  1.3  1.3  0.0  0.0  0.0
        8   0.0  0.0  0.0  0.0  0.0  4.6 22.4  2.0  0.0
        9   0.0  0.0  0.0  0.0  0.0  0.0  2.0  3.3  0.0
        10  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.3  8.6

 Accuracy (average) : 0.7434
```

3. Support Vector Machines

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.

Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting).

Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

SVM Training Accuracy before PCA= 0.7385890254

SVM Training Accuracy after PCA= 0.6886428214

SVM Testing Accuracy before PCA= 0.8333333333

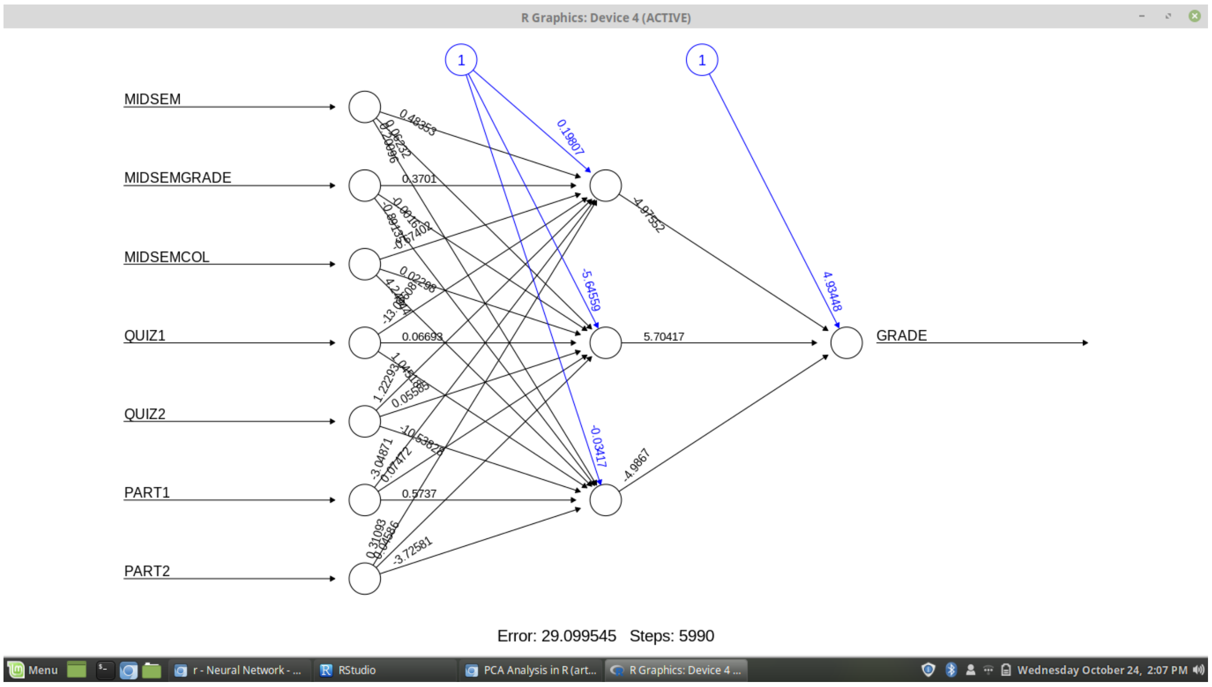
SVM Testing Accuracy after PCA= 0.9025615225

4. Neural Networks

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system.

It is composed of a large number of highly interconnected processing elements (neurones) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurones. This is true of ANNs as well.

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyse. This expert can then be used to provide projections given new situations of interest and answer "what if" questions.



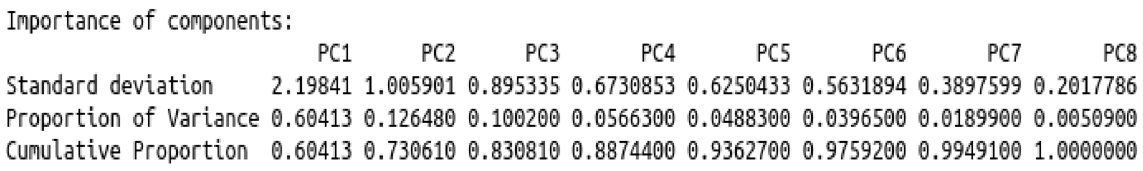
5. PCA (Principal Component Analysis)

   Principal Component Analysis (PCA) is a dimension-reduction tool that can be used to reduce a large set of variables to a small set that still contains most of the information in the large set.

   Principal component analysis (PCA) is a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible.

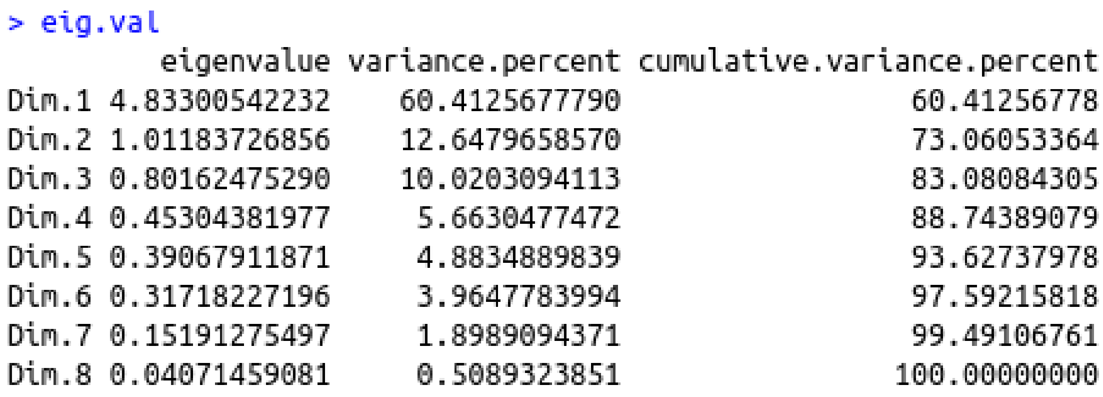
   For performing PCA, first feature scaling has to be performed to scale up/down the attributes to the same range. After this, PCA was performed using the above described method. All of this was accomplished using ‘’ method in ‘’ library.

PCA Matrix:



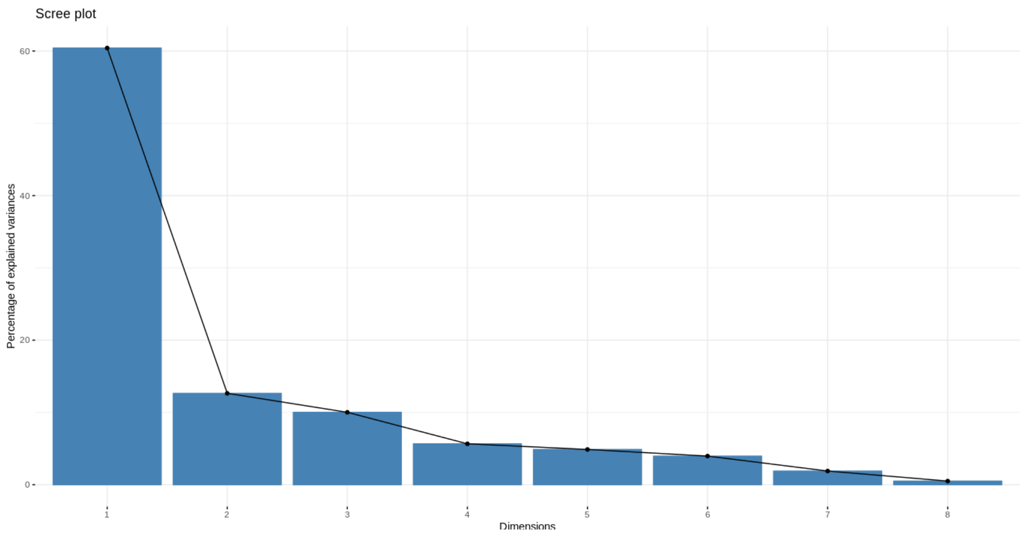
```
Importance of components:
                           PC1      PC2      PC3       PC4       PC5       PC6       PC7       PC8
Standard deviation     2.19841 1.005901 0.895335 0.6730853 0.6250433 0.5631894 0.3897599 0.2017786
Proportion of Variance 0.60413 0.126480 0.100200 0.0566300 0.0488300 0.0396500 0.0189900 0.0050900
Cumulative Proportion  0.60413 0.730610 0.830810 0.8874400 0.9362700 0.9759200 0.9949100 1.0000000
```

Eigen values:

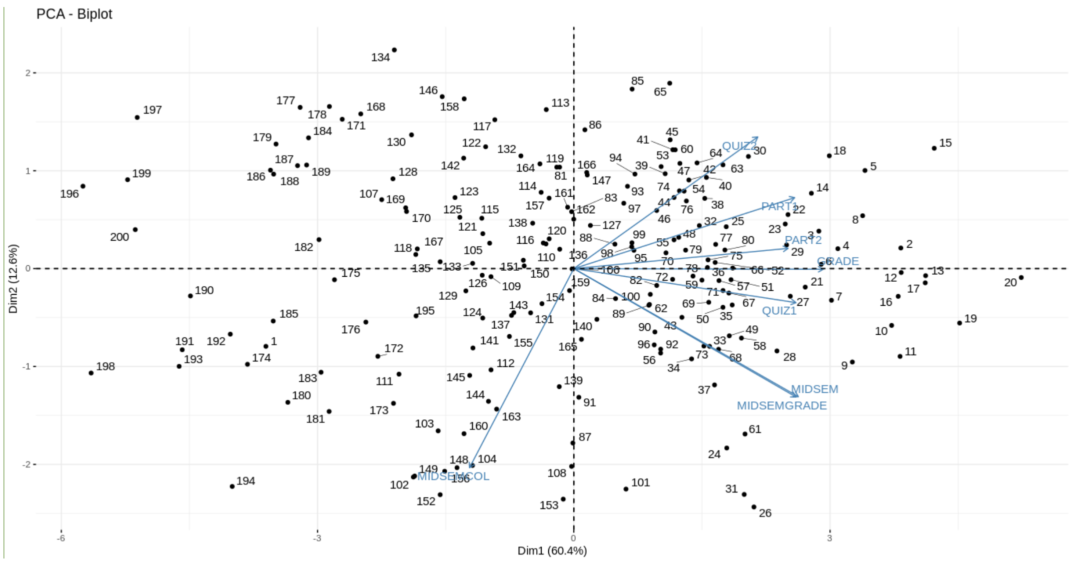


```
> eig.val
        eigenvalue variance.percent cumulative.variance.percent
Dim.1 4.83300542232    60.4125677790                 60.41256778
Dim.2 1.01183726856    12.6479658570                 73.06053364
Dim.3 0.80162475290    10.0203094113                 83.08084305
Dim.4 0.45304381977     5.6630477472                 88.74389079
Dim.5 0.39067911871     4.8834889839                 93.62737978
Dim.6 0.31718227196     3.9647783994                 97.59215818
Dim.7 0.15191275497     1.8989094371                 99.49106761
Dim.8 0.04071459081     0.5089323851                100.00000000
```

Scree Plot:



Scatter Plot:



REPRESENTATIVE QUESTIONS

- Can we predict the final grades of students using the data given?

  We can clearly observe that the above used algorithms like decision tree, Naïve Bayes, ANN and SVM have provided us with a good accuracy of about 65-80%.

- Which attributes contribute towards the final grade?

  It is evident that the attribute 'IDNO' doesn't affect the output in any way, hence it is removed from the equation. First of all we remove 'CGPA' from the data as it has only 40 values despite having a high correlation value. Considering a correlation threshold value of 0.3 we eliminate the unwanted variables. M/F', 'Year' and 'Attendance' have been eliminated due to their correlation values. Rest seven are used.

- Is PCA a good choice for dimensionality reduction?

  PCA has definitely increased the performance of all the algorithms (by almost 10-15%) and it has proved to be very effective for dimensionality reduction for the given dataset.

- Is there a causality relation between attributes?

  There is a cause-and-effect relation between the attributes. These relations can be observed from the correlation heatmap shown earlier. It can be seen that midsem grade has a direct cause and effect relationship with midsem marks. Moreover, CGPA influences the marks obtained in evaluation components like Part 1 and Part 2. Similarly, low attendance can be a cause for higher number of attempts required for midsem answer sheet collection.

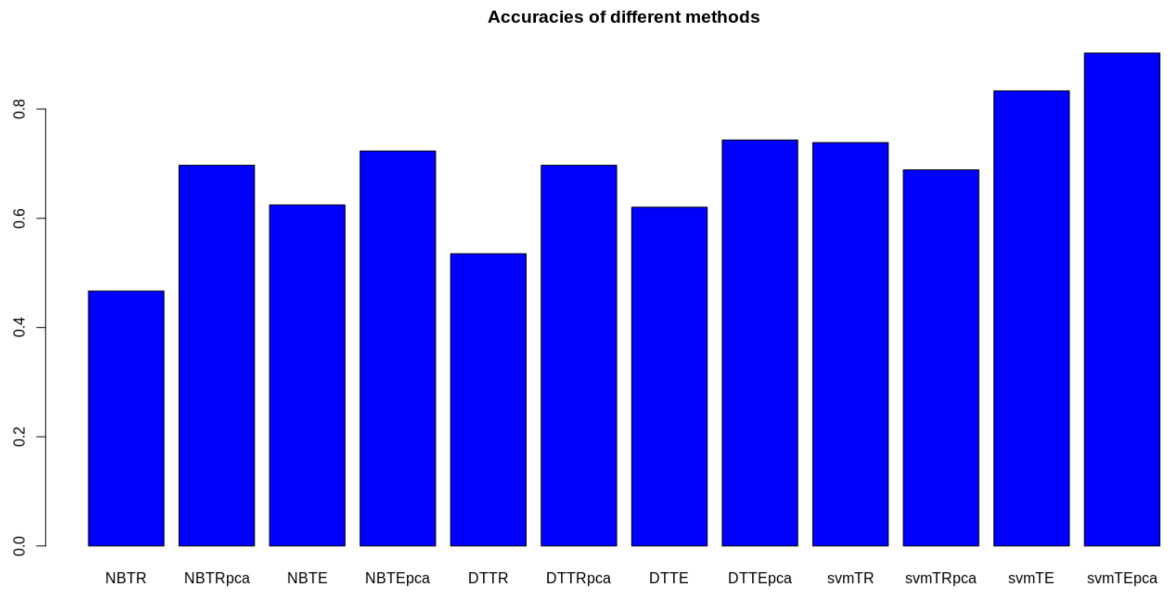
CONCLUSION

The datatset was initially processed and the and applied to various training algorithms. Unwanted parameters were eliminated by distinguishing useful ones from the useless ones.

Accuracies are satisfactory and we are moving in the right direction as there is a higher training accuracy and a lower generalisation accuracy. SVM was found to be the best algo for this problem.

Later we also performed dimensionality reduction with PCA and observed changes in accuracy.

This figure shows how accuracy is varying with different techniques:



CODES

1. Correlation

```
data <- read.csv("/home/akshit/Documents/Downloads/gradenew.csv")
d <- data.frame(MIDSEM = data$MIDSEM,
            MIDSEMGRADE = data$MIDSEMGRADE,
            MIDSEMCOL = data$MIDSEMCOL,
            QUIZ1 = data$QUIZ1,
            QUIZ2=data$QUIZ2,
            PART1=data$PART1,
            PART2 = data$PART2,
            GRADE = data$GRADE)
cor(d)
```

2. Decision Tree

```
library(caret)
df <- read.csv("/home/akshit/Documents/Downloads/gradenew.csv")
library(caTools)
set.seed(12345)
sample = sample.split(df$GRADE,SplitRatio = 0.60)
train <- subset(df,sample == TRUE) # creates a training dataset named train1 with rows which
are marked as TRUE
test <- subset(df, sample == FALSE)
print(train)
library(party)
input.dat <- train
png(file = "decision_tree.png")
output.tree <- ctree(
  GRADE ~ MIDSEM + MIDSEMGRADE + MIDSEMCOL + QUIZ1 + QUIZ2 + PART1 + PART2  ,
  data = input.dat)
plot(output.tree)
dev.off()
library (rpart)
rpartMod <- rpart(GRADE ~ MIDSEM
+MIDSEMGRADE+MIDSEMCOL+QUIZ1+QUIZ2+PART1+PART2,data = train, method = "class")
# build the model
printcp(rpartMod)  # print the cptable
out <- predict(rpartMod) # predict probabilities
pred.response <- colnames(out)[max.col(out, ties.method = c("random"))] # predict response
mean(train$GRADE != pred.response)
#out <- predict(rpartMod, test)
#print(out)
```

3. Naïve Bayes

```
library(e1071)
library(psych)
library(rsample)
library(dplyr)    # data transformation
library(ggplot2)  # data visualization
library(caret)    # implementing with caret
```

```
df <- read.csv("/home/akshit/Documents/Downloads/gradenew.csv")
#wb.raw                                                                                    <-
data.frame(df$MIDSEM,df$MIDSEMGRADE,df$MIDSEMCOL,df$QUIZ1,df$QUIZ2,df$PART1,df
$PART2)
#pairs.panels(wb.raw)
set.seed(123)
split <- initial_split(df, prop = .7, strata = "GRADE")
train <- training(split)
test  <- testing(split)
#print(train)
table(train$GRADE) %>% prop.table()
table(test$GRADE) %>% prop.table()
train %>%
  select(MIDSEMGRADE, MIDSEMCOL, QUIZ1, QUIZ2, PART1, PART2,MIDSEM) %>%
  gather(metric, value) %>%
  ggplot(aes(value, fill = metric)) +
  geom_density(show.legend = FALSE) +
  facet_wrap(~ metric, scales = "free")
features <- setdiff(names(train), "GRADE")
x <- train[, features]
y <- as.factor(train$GRADE)
train_control <- trainControl(
  method = "cv",
  number = 10
)
nb.m1 <- train(
  x = x,
  y = y,
  method = "nb",
  trControl = train_control
)
confusionMatrix(nb.m1)
search_grid <- expand.grid(
  usekernel = c(TRUE, FALSE),
  fL = 0:5,
  adjust = seq(0, 5, by = 1)
)
nb.m2 <- train(
  x = x,
  y = y,
  method = "nb",
  trControl = train_control,
  tuneGrid = search_grid,
  preProc = c("BoxCox", "center", "scale", "pca")
)
nb.m2$results %>%
  top_n(5, wt = Accuracy) %>%
  arrange(desc(Accuracy))
plot(nb.m2)
confusionMatrix(nb.m2)
```

4. Neural Networks

```
library(caret)
library(neuralnet)
samplesize = 0.60 * nrow(data)
set.seed(80)
index = sample( seq_len ( nrow ( df ) ), size = samplesize )
# Create training and test set
train = data[ index, ]
test = data[ -index, ]
#train[["GRADE"]] = factor(train[["GRADE"]])
#test[["GRADE"]] = factor(test[["GRADE"]])
max = apply(df , 2 , max)
min = apply(df, 2 , min)
scaled = as.data.frame(scale(df, center = min, scale = max - min))
trainNN = scaled[index , ]
testNN = scaled[-index , ]
set.seed(5)
NN = neuralnet(GRADE ~ MIDSEM + MIDSEMGRADE + MIDSEMCOL + QUIZ1 + QUIZ2 + PART1
+ PART2 ,data=train,hidden=3, linear.output = T )
plot(NN)
dev.off()
#predict_testNN = compute(NN, testNN[,c(1:5)])
#predict_testNN = (predict_testNN$net.result * (max(df$GRADE) - min(df$GRADE))) +
min(df$GRADE)
#plot(test$GRADE, predict_testNN, col='blue', pch=16, ylab = "predicted GRADE NN", xlab =
"real GRADE")
#abline(0,1)
# Calculate Root Mean Square Error (RMSE)
#RMSE.NN = (sum((test$GRADE - predict_testNN)^2) / nrow(test)) ^ 0.5
```

5. PCA

```
library(caret)
df <- read.csv("/home/akshit/Documents/Downloads/gradenew.csv")
print(df)
library(factoextra)
df.pca <- prcomp(df,center = TRUE,scale. =TRUE)
summary(df.pca)
fviz_eig(df.pca)
fviz_pca_ind(df.pca,
        col.ind = "cos2", # Color by the quality of representation
        gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
        repel = TRUE     # Avoid text overlapping
)
fviz_pca_biplot(df.pca, repel = TRUE,
          col.var = "#2E9FDF", # Variables color
          col.ind = "#696969"  # Individuals color
)
eig.val <- get_eigenvalue(df.pca)
eig.val
df.var <- get_pca_var(df.pca)
df.var$coord
df.var$contrib       # Contributions to the PCs
df.var$cos2
df.ind <- get_pca_ind(df.pca)
pp <- as.data.frame(df.ind$coord)     # Coordinates
df.ind$contrib        # Contributions to the PCs
df.ind$cos2          # Quality of representation
#str(df.pca)
#devtools::install_github("kassambara/ggpubr")
#devtools::install_github("kassambara/factoextra")
library(caTools)
set.seed(12345)
sample = sample.split(pp$Dim.1 ,SplitRatio = 0.60)
train <- as.data.frame(subset(df.ind$coord ,sample == TRUE)) # creates a training dataset
named train1 with rows which are marked as TRUE
test <- as.data.frame(subset(df.ind$coord , sample == FALSE))
print(train)
library(party)
input.dat <- train
png(file = "decision_tree.png")
output.tree <- ctree(
  GRADE ~ MIDSEM + MIDSEMGRADE+MIDSEMCOL+QUIZ1+QUIZ2+PART1+PART2,
  data = input.dat)
plot(output.tree)
dev.off()
library (rpart)
rpartMod <- rpart(GRADE ~ ., data = train, method = "class")  # build the model
printcp(rpartMod)  # print the cptable
out <- predict(rpartMod) # predict probabilities
pred.response <- colnames(out)[max.col(out, ties.method = c("random"))] # predict response
mean(test$GRADE != pred.response)
out <- predict(rpartMod, test)
```

6. SVM

```
library(caret)
df <- read.csv("/home/akshit/Documents/Downloads/gradenew.csv")
png(file = "svm.png")
plot(df, pch=16)
dev.off()
set.seed(3033)
split <- createDataPartition(y = df$GRADE, p= 0.7, list = FALSE)
train <- df[split,]
test <- df[-split,]
dim(train)
dim(test)
anyNA(df)
train[["GRADE"]] = factor(train[["GRADE"]])
test[["GRADE"]] = factor(test[["GRADE"]])
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
set.seed(3233)
svm_Linear <- train(GRADE ~., data = train, method = "svmLinear",
                trControl=trctrl,
                preProcess = c("center", "scale"),
                tuneLength = 10)
```

```
svm_Linear
test_pred <- predict(svm_Linear, newdata = test)
confusionMatrix(test_pred,test$GRADE)
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
set.seed(3233)
svm_Linear <- train(GRADE ~., data = train, method = "svmLinear",
                 trControl=trctrl,
                 preProcess = c("center", "scale","pca"),
                 tuneLength = 10)
svm_Linear
test_pred <- predict(svm_Linear, newdata = test)
confusionMatrix(test_pred,test$GRADE)
```
