EARLY DETECTION OF CERVICAL CANCER BASED ON BEHAVORIAL AND SOCIAL FACTORS USING CLASSIFICATION ALGORITHMS

Analysis done by: Sahana Mukherjee

UBIT Name: mukherj3

Introduction: Cervical Cancer is a cancer that occurs in the cells of the cervix in women, and happens to be the second most common cancer among women worldwide. Important behavioral and social variables act as determining factors, when it comes to assessing the risk of an individual having the disease. Cervical cancer can be cured if detected at an early stage, and here is where our assignment comes in. Here I have deployed classification algorithms, to categorize two classes, class 1 being those individuals who have the disease and class 2 being those not having the disease. Based on 19 social and behavioral variables, we want to classify whether an individual has the disease or not, and using the train-test convention, we aim to predict the same on test data or unforeseen data.

Environment used: RStudio

Libraries and packages used: ggplot2, heatmaply, pysch ,logistf ,ROCit, ModelMetrics, caret

---

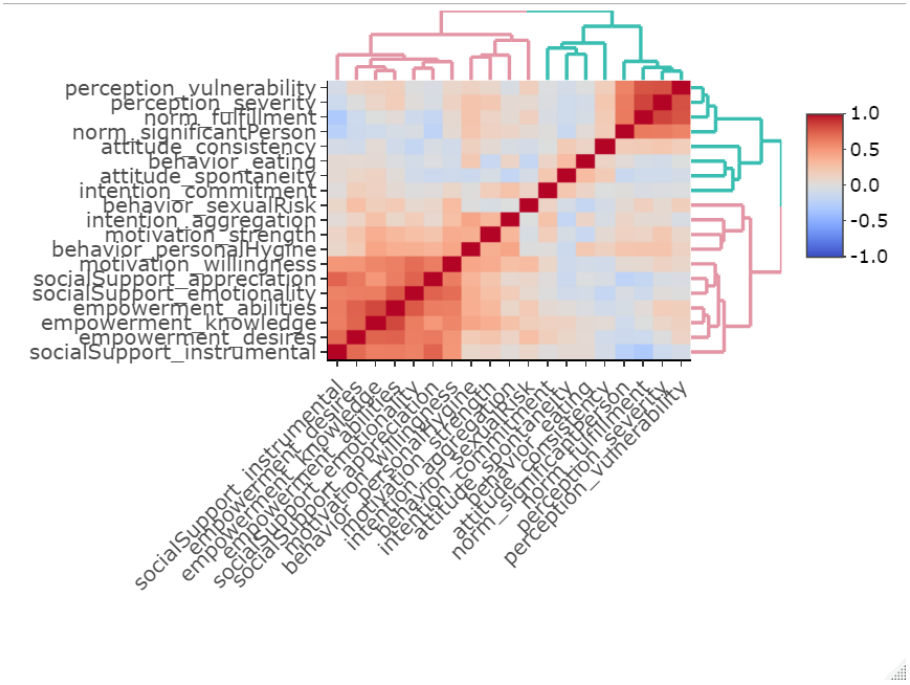
Exploratory Data Analysis:

We first do an exploration on the available data set.

```
Cancer_DataSet <- read.csv("cervical_cancer.csv", sep=",", header = TRUE)

sum(is.na(Cancer_DataSet[1:20])) #no missing values found in the data set

head(Cancer_DataSet) # checking the first few entries of the data

summary(Cancer_DataSet)

pairs(Cancer_DataSet[,-20], gap=0.9, pch=19, cex=0.4, col="darkblue")

library(ggplot2) #using predefined R library ggplot2

library(heatmaply) #using predefined R library heatmaply, for plotting heatmap which gives the
correlation among independent variables.

heatmaply_cor(

  cor(Cancer_DataSet[,-20]),

  k_col = 2,

  k_row = 2)
```

Observation:

Using the above usual commands, we observe the correlation among the independent variables, summary of the data, mean, median, IQR of the variables and also find that there are no null values in our data set. The following heatmap gives the correlation among the independent variables.



We observe that there exists correlation among the independent variables. Ideal scenario should be such that there exists no correlation among the independent variables and they should be only related with the target variable. However, real world data sets come with correlation among the variables, and we have multiple ways to deal with them. We will explore one of those methods later in this assignment.

Classification using Logistic Regression:

We deploy the most common classification algorithm of logistic regression on our model.

Usual train-test split of the data set is done and the model is trained on the data and then tested on test data. Pieces of R code are given below:

```
smp_size <- floor(0.70 * nrow(Cancer_DataSet)) # using 70-30 split for train and test.

set.seed(1)   ##setting a random seed value to get the same split everytime.

train_ind <- sample(seq_len(nrow(Cancer_DataSet)), size = smp_size)

scaled_X_train <- scale(training_set[,-20])

y_train <- training_set$ca_cervix

scaled_x_test <- scale(testing_set[, -20])
```

y_actual<-testing_set$ca_cervix

y_actual

mylogit <- logistf(y_train~ . ,data = as.data.frame(scaled_X_train))

predicted <- predict(mylogit, scaled_x_test, type="response")

y_predicted <- ifelse(predicted > 0.8, 1, 0) ##setting the threshold probability at 80%, which means if probability of getting cervical cancer is 80% or more, we assign the value of 1 to that individual.

Observation:

After having trained our model on the training set, and then making predictions on the test data, we now want to evaluate how good our model performed on the test data. Because in real world, our model will go out into the real data sets and will be responsible for making predictions on unforeseen data.

<u>CONTINGENCY TABLE:</u>

	Predicted	
Actual	0	1
0	8	5
1	1	8

- Accuracy: 72.7%
- True Positives: 8, True Negatives: 8, False Positives: 5, False Negatives: 1
- Sensitivity: 61%,
- Specificity : 88%

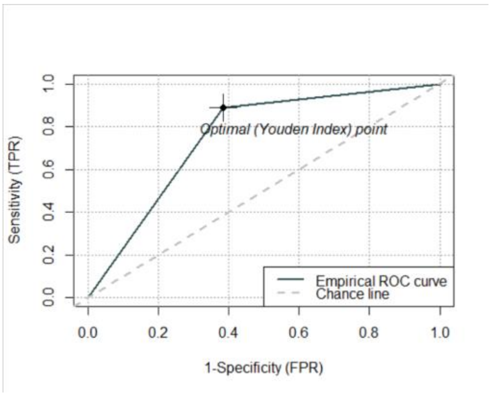
The ROC :

- Area under the curve: 75%

Piece of code: *plot(rocit(score= y_predicted,class = y_actual))*

area_under_ROC_curve <-rocit(score=y_predicted,class=y_actual)

area_under_ROC_curve



Conclusion from LR Algorithm : After performing classification using LR, we find that our model has about 72% accuracy, which is pretty good, and it has only one false negative, which again is a plus. This is because, from healthcare and patient's safety point of view, our model should predict less false negatives. False negatives mean that the individual actually has the disease but our model failed to predict that, instead predicted that he/she does not have the disease, and this raises false safety alarm, and prevents the individual from knowing the risk and getting proper treatment, which is undesirable.

The AUC covers about 75% which gives a pretty well trade off between true positives and true negatives.

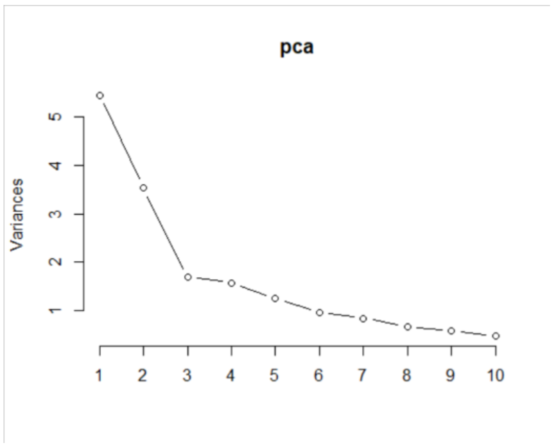
So, overall, we can conclude that our model performed quite well on the test data and yielded good accuracy scores.

Classification using Logistic Regression after performing Principal Component Analysis:

In exploratory data analysis, we have seen that there exists correlation among the independent variables, which is not the ideal case. Now we would like to reduce the dimension of the data set from 19 independent variables to that number of principal components which can explain the maximum variability in the data. Then after doing this reduction, we perform LR on this reduced data set and observe the accuracy scores. Pieces of code for PCA are given below:

```
pca <- prcomp(scale(Cancer_DataSet_ForPCA[,-20]))

scores = pca$x

pca_scores = scores[, 1:7] #####using first 7 principal components for further analysis

smp_size <- floor(0.70 * nrow(pca_scores))## using the usual 70-30 train test split method###

pca_training_set<- pca_scores[train_ind, ]

pca_testing_set <- pca_scores[-train_ind, ]

pca_model <- glm(y_train~ .,data=as.data.frame(pca_training_set), family=binomial(link="logit"))

ypca_predicted_prob <- predict(pca_model, newdata = as.data.frame(pca_testing_set), type =
"response")

ypca_predicted=ifelse(ypca_predicted_prob>0.8,1,0)
```

Observations from PCA : After doing PCA, excluding the target variable, we find that the first 7 principal components are sufficient to describe 80.4% of the variability of the data. So we reduce the dimension of our data set from 19 independent variables to 7 independent variables.



Now we apply logistic regression on this reduced data set, where we use the first 7 principal components as the independent variables and based on those, we make predictions on the target variable.

Piece of code:

pca_model <- glm(y_train~ .,data=as.data.frame(pca_training_set), family=binomial(link="logit"))

Observation :

<u>CONTINGENCY TABLE:</u>

	Reference	
Prediction	0	1
0	10	3
1	0	9

- Accuracy = 86%
- True Positives: 9, True Negatives: 10, False Positives: 0, False Negatives: 3
- Sensitivity: 75%
- Specificity = 1
- Precision : 1
- Recall : 75%
- F1 = 85%
- AUC : 88%

Conclusion from LR model, after performing PCA: We observe that, if we apply LR on our data set, after performing PCA, we get a considerable increase in the accuracy of the model. This is due to the underlying fact that PCA reduces the dimensions, and the new principal components which are formed by the weighted linear combination of the original variables, does not share any correlation among them, unlike the original input variables. This leads us to the conclusion that multicollinearity, that is, dependency among input variables can interfere with the model performance, and we should aim at reducing this dependency, yet capturing as much as information as possible.

Classification using Linear Discriminant Analysis (LDA):

Now, we employ the algorithm of LDA to classify our target variable. LDA like PCA, looks at weighted linear combinations of independent variables, but unlike PCA, it looks exclusively at the differences between the classes.

lda_model=lda(lda_y_train~.,data=as.data.frame(lda_training_set[,-20]))

lda_predicted = predict(lda_model, newdata = as.data.frame(lda_testing_set))

Observation From LDA:

CONTINGENCY TABLE:

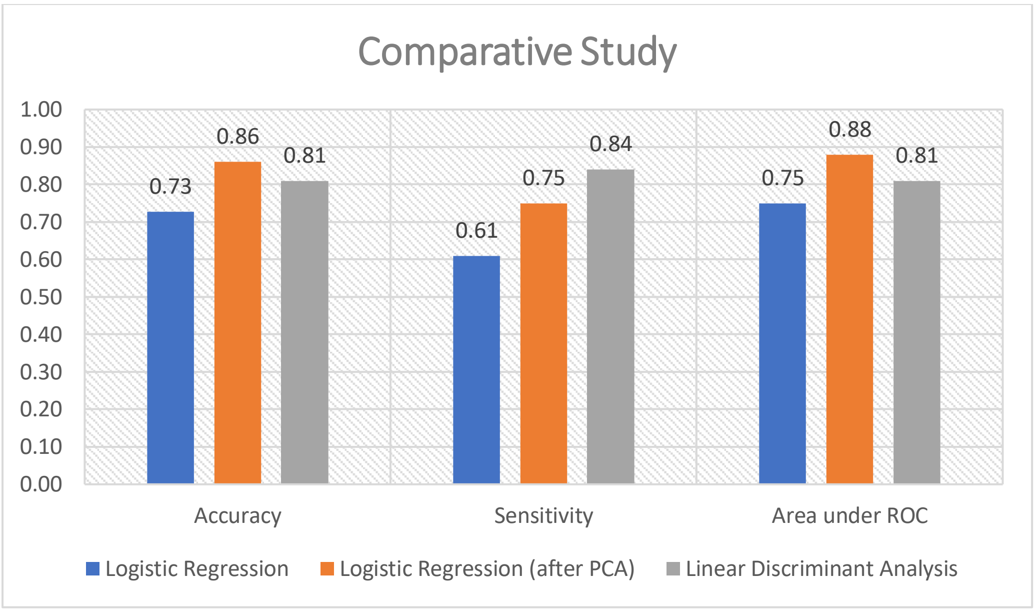
	Prediction	
Actual	0	1
0	11	2
1	2	7

- Accuracy = 81%
- Sensitivity = 84%
- Specificity: 77%
- Precision : 84%
- Recall = 84%
- F1 score = 84%
- Area under ROC or AUC : 81%

Conclusion from LDA: By doing linear discriminant analysis, we find that we can significantly increase the performance of the model, as compared to logistic regression. LDA explicitly looks at the differences between the two classes, and separates them, based on the weighted linear combination of independent variables.

COMPARATIVE STUDY ON THE THREE CLASSIFICATIONS:

Now, we do a comparative study of the three classification models used, and how their performance varies. We see that we achieve the maximum performance when we do a LR after PCA on the data set, followed by LDA, and LR. This leads us to the conclusion, that dimension reduction plays a significant role in impacting the model's performance, because multicollinearity interferes with the accuracy. In ideal scenario, the input variables should share no correlation among them, but this does not occur in real time, so our aim is always to get rid of such dependencies, yet not losing information of the original data set. Below is the graph showing the model performances, taking into account three important parameters, Accuracy, Sensitivity and AUC.



<u>IDENTIFYING CULTIVAR OF WHITE WINE, USING NAÏVE BAYES AND RANDOM FOREST CLASSIFIER</u>

Analysis done by : Sahana Mukherjee

UBIT Name: mukherj3

Environment used: RStudio

Packages used: randomForest, ModelMetrics, caret, e1071

Introduction and Exploratory Data Analysis:

White Wine has several chemical properties and based on that the cultivar of the wine can be identified. Our aim is to predict the cultivar of the white wine, given 13 chemical properties, on a completely new unforeseen data, after training our model on the available test data.

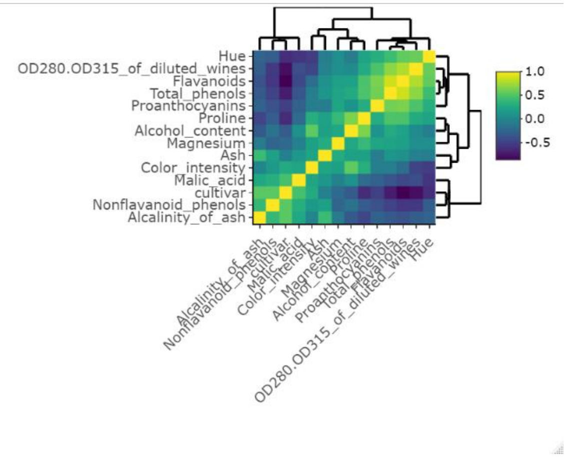
sum(is.na(WhiteWineCultivar)) #### checking for missing values in the dataset

WhiteWineCultivar <- na.omit(WhiteWineCultivar) ####omitting the missing values###

summary(WhiteWineCultivar)

library(heatmaply) ##### plotting the correlation among independent variables using heatmap

Observation: We observe that there are 70 missing values in our data set, and we proceed after omitting those. Then we do a heat map to obtain a visual idea about the correlation among the 13 independent variables.



We see that there are pretty strong correlation among the independent variables, which should not be the ideal scenario. We will deal with this later as we move on.

Prediction using Naive Bayes Classifier:

Now, we use the conventional train-test split of the available data, and train our model using the Naïve Bayes Classifier.

Naive_Bayes_Model=naiveBayes(y_train ~., data=train_df)

NB_Predictions=predict(Naive_Bayes_Model,test_df)

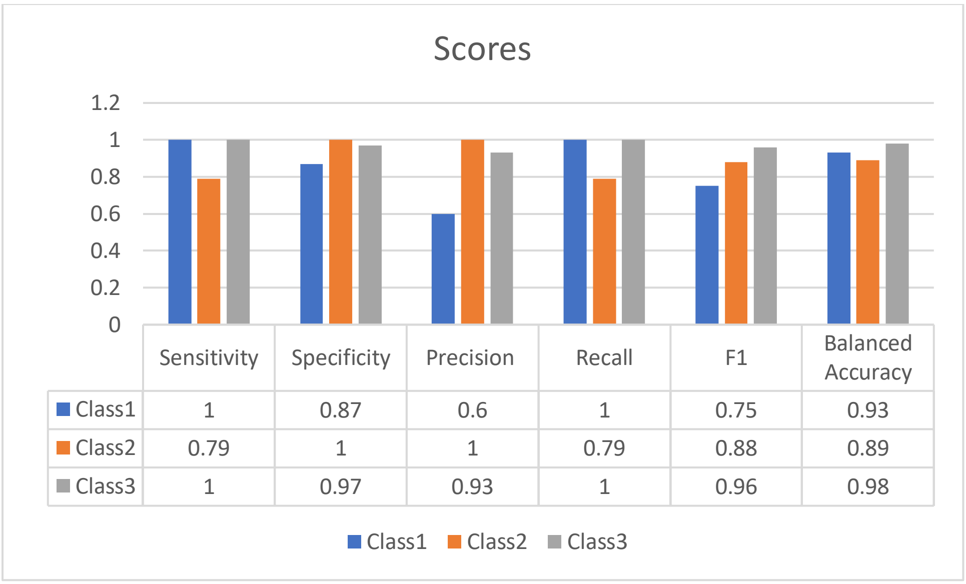
Observations:

CONTINGENCY TABLE:

	Actual		
Predictions	1	2	3
1	9	6	0
2	0	27	0
3	0	1	15

Comparative study on scores:

Now, we do a comparative study on the model's performance, based on important parameters like accuracy, precision, recall etc., for the three classes. We observe that our model has performed pretty well on the test data set of the available data. Once our model performs good on the test data, we can now go ahead to deploy this model to predict on unforeseen data.



	Sensitivity	Specificity	Precision	Recall	F1	Balanced Accuracy
🟦 Class1	1	0.87	0.6	1	0.75	0.93
🟧 Class2	0.79	1	1	0.79	0.88	0.89
⬜ Class3	1	0.97	0.93	1	0.96	0.98

```
##############importing the unforeseen data#################

WhiteWineCultivarTest <- read.csv("./winedata_testing.csv", sep=",", header = TRUE)

NewPred=predict(Naive_Bayes_Model,scale(WhiteWineCultivarTest))

NewPred  #####These are the new predictions on unforeseen data#########

1 1 1 2 2 2 2 3 3 3

Levels: 1 2 3
```

Prediction using Random Forest Classifier:

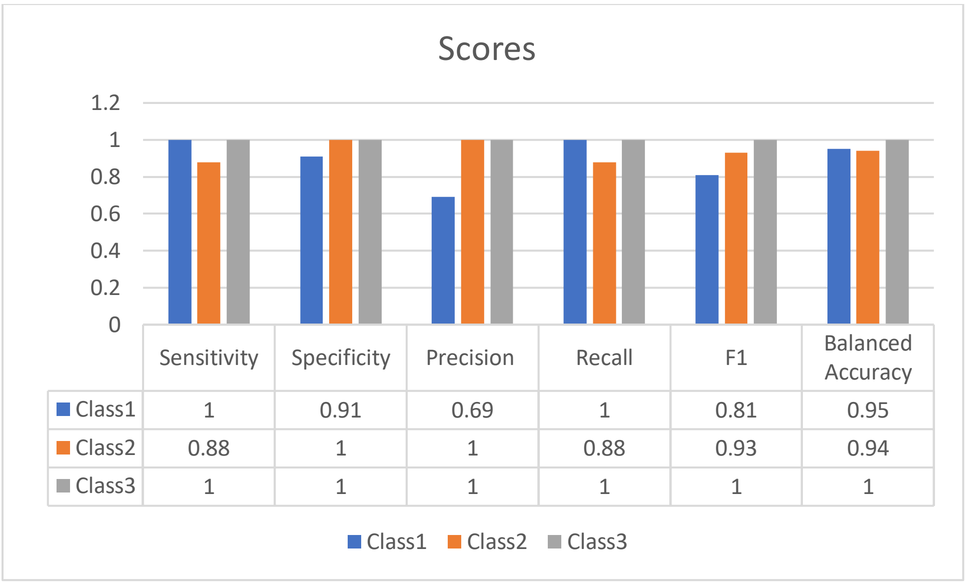
Now we use another common classifier algorithm, Random forest to make the predictions on the unforeseen data and evaluate how the random forest classifier performs on the train test data sets of the available data.

```
model=randomForest(y_train ~ ., data = training_set, ntree = 500, mtry = 3, importance = TRUE)

pred_testing_data=predict(model,testing_set,type="class")
```

<u>CONTINGENCY TABLE:</u>

	Actual		
Predictions	1	2	3
1	9	6	0
2	0	28	0
3	0	0	15

Comparative Study on Scores:



	Sensitivity	Specificity	Precision	Recall	F1	Balanced Accuracy
🟦 Class1	1	0.91	0.69	1	0.81	0.95
🟧 Class2	0.88	1	1	0.88	0.93	0.94
⬜ Class3	1	1	1	1	1	1

Prediction on Unforeseen new data:

```
WhiteWineCultivarTest <- read.csv("./winedata_testing.csv", sep=",", header = TRUE)

Pred_unoforeseen_data=predict(model,scale(WhiteWineCultivarTest),type="class")

Pred_unoforeseen_data
```

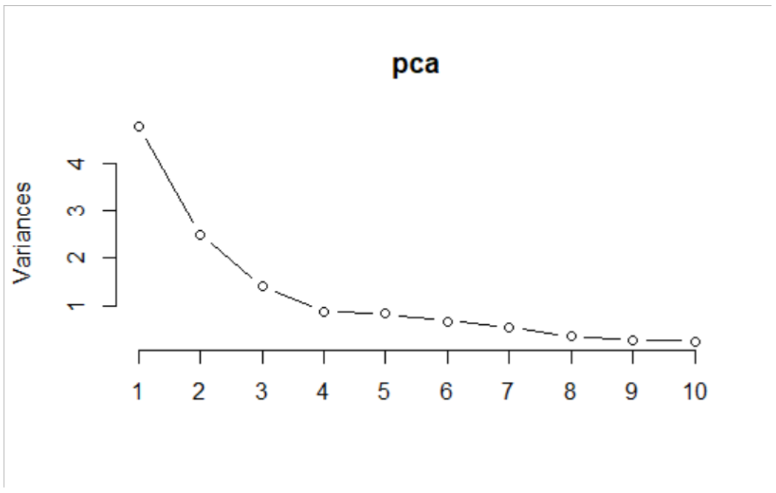
1 1 1 2 2 2 2 3 3 3

Levels: 1 2 3

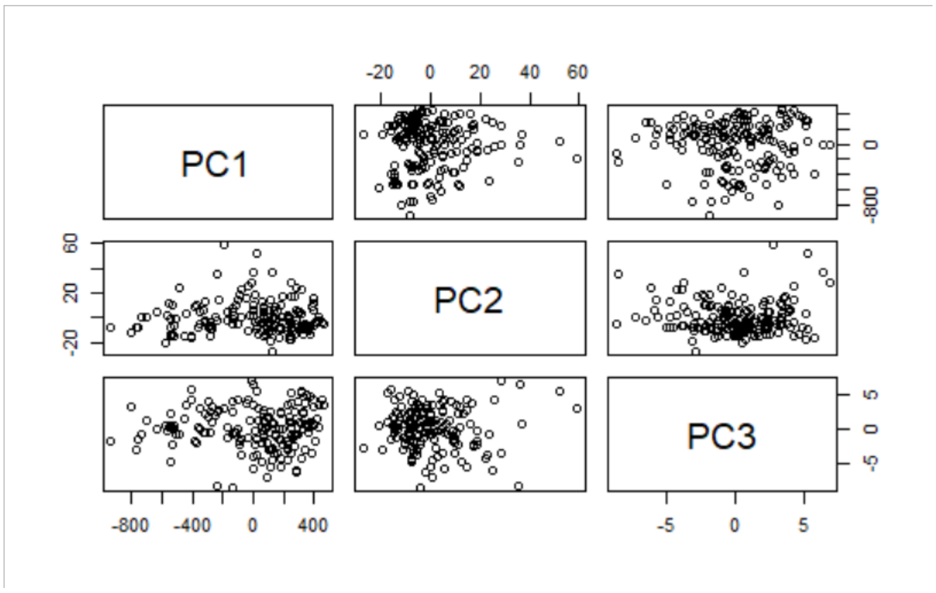
A LOOK INTO PCA:

In the previous classification, we have seen that dimension reduction using PCA, can significantly lead to an increase in the accuracy of the models.

With the White Wine data, when we do the PCA, we see that three PCs can describe about 67% of the variability in the data.This is definitely a good option, because on one hand it helps us to reduce the dimension of the huge data set, as well as allows us to get rid of multicollinearity, which we have seen previously, interferes with the model’s performance.

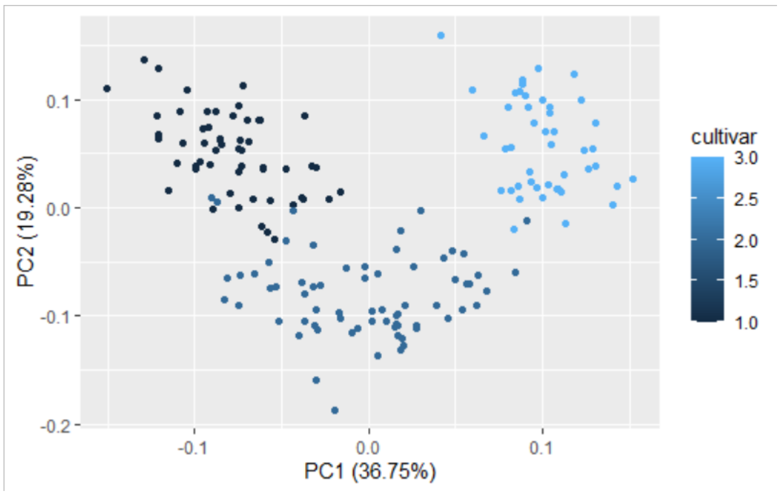


Pair Plot of the new training data set composed of 3 principal components:

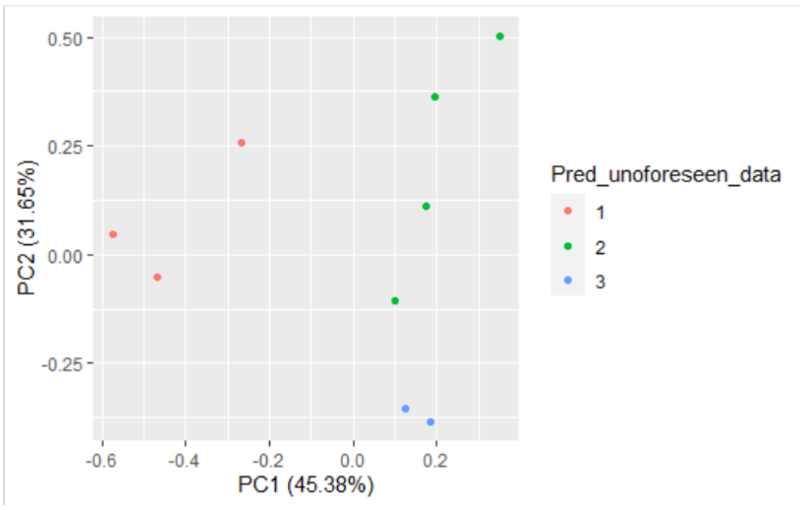


Observation: There is no correlation among these three PCs, unlike the 13 original input variables of the data set. So we have successfully reduced the dimension of the data set, yet capturing 99% of the variability of the data.

Now we do a plot to see how these 3 dimensions can describe the three different classes. We find that the three classes are well separated with very little overlap.



Now, we run a PCA on the unforeseen data, then, bind it with the newly predicted cultivar values, and observe how the three different classes are separated.



So our model performs pretty well in classifying the three cultivar classes and also PCA , helps us in reducing the dimensions and get rid of multicollinearity.

Some important notes:

There are 70 missing values in the data set of White Wine Cultivar.

We need to normalize the data before performing any train test split.

We need to normalize the data before doing PCA.

For Cervical Cancer data set, it is important to convert the target variable into factor.