**CODINGS**

**ANACONDA SPYDER PYTHON**

**# -\*- coding: utf-8 -\*-**

**"""**

**Created on Fri May 22 19:25:22 2020**

**@author: user**

**"""**

**# -\*- coding: utf-8 -\*-**

**"""**

**Created on Sun Mar 1 21:17:54 2020**

**@author: user**

**"""**

**import pandas as pd**

**mydataset = pd.read\_csv('data.csv')**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**import numpy as np**

**mydataset.shape**

**mydataset.head(7)**

**# FINDING A NULL VALUES**

**mydataset.isna().sum()**

**#DROP THE COLUMN WITH ALL MISSING VALUES**

**mydataset = mydataset.dropna(axis=1)**

**#GET the new count of the number of rows and columns**

**mydataset.shape**

**mydataset.describe()**

**mydataset.isnull().values.any()**

**#GET THE COUNT OF THE NUMBER OF MALIGANT (M) OR BENIGN (B) CELLS**

**mydataset['diagnosis'].value\_counts()**

**print("Cancer data set dimensions : {}".format(mydataset.shape))**

**#Understand the target varibale further**

**mydataset["diagnosis"].value\_counts()#Visualize dataset using Histograms**

**#A histogram is a plot that lets you discover, and show, the underlying frequency distribution**

**import matplotlib.pyplot as plt**

**num\_bins = 10**

**mydataset.hist(bins=num\_bins, figsize=(20,15))**

**plt.show()**

**#VISUALIZE THE COUNT**

**sns.countplot(mydataset['diagnosis'], label='count')**

**#Look at the (dtype) data types to see which columns need to be encoded**

**mydataset.dtypes**

**#ENCODE THE CATEGORICAL DATA VALUES**

**from sklearn.preprocessing import LabelEncoder**

**labelencoder\_Y = LabelEncoder()**

**mydataset.iloc[:,1] = labelencoder\_Y.fit\_transform(mydataset.iloc[:,1].values)**

**mydataset.iloc[:,1]**

**# create apair plot**

**sns.pairplot(mydataset.iloc[:,1:6])# 6 is nothing but index**

**sns.pairplot(mydataset.iloc[:,1:6], hue='diagnosis')**

**#print the first 5 rows of the new data**

**mydataset.head(5)**

**#get the correlation of the columns**

**mydataset.iloc[:,1:12].corr()**

**"""**

**#IF WE WANT IN COLOR HEATMAP WE CAN GO FOR IT#**

**from matplotlib.colors import ListedColormap**

**plt.figure(figsize=(8, 8))**

**sns.heatmap(mydataset.iloc[:,1:12].corr(), cbar=True, annot=False, fmt='.0%',**

**cmap=ListedColormap(['#C71585', '#DB7093', '#FF00FF', '#FF69B4', '#FFB6C1', '#FFC0CB']),**

**)**

**plt.show()**

**"""**

**#visualize the corelation**

**plt.figure(figsize=(10,10))**

**sns.heatmap(mydataset.iloc[:,1:12].corr(), annot=True, fmt='.0%')**

**#box plot to check outlier in each category**

**#define function can be call later**

**def boxPlot(dff):**

**d=dff.drop(columns=['diagnosis'])**

**for column in d:**

**plt.figure(figsize=(5,2))**

**sns.boxplot(x=column,data=d,palette="colorblind")**

**boxPlot(mydataset)**

**#Box plot for radius mean**

**#We can use box plots to identify outliers in a dataset.**

**#These can be rare occurrence or errors. Sometimes these can provide insight.**

**melted\_data = pd.melt(mydataset,id\_vars = "diagnosis",value\_vars = ['radius\_mean'])**

**plt.figure(figsize = (15,10))**

**sns.boxplot(x = "variable", y = "value", hue="diagnosis",data= melted\_data)**

**plt.show()**

**#From the box plot we can see the range of the radius mean values for bening and malignant tumors.**

**#We can also clearly see the outliers in the above visualizations (marked points lying outside the range).**

**#Box plot for area mean**

**melted\_data = pd.melt(mydataset,id\_vars = "diagnosis",value\_vars = ['area\_mean'])**

**plt.figure(figsize = (15,10))**

**sns.boxplot(x = "variable", y = "value", hue="diagnosis",data= melted\_data)**

**plt.show()**

**#INFERENCE: From the graph, we can say that after a certain threshold for the values of the given features,**

**# we can classify the tumor as malignant (but we also see outliers, so maybe not with a 100% certainty**

**#every time, there could be some false positives).**

**#Box plot for perimeter mean**

**melted\_data = pd.melt(mydataset,id\_vars = "diagnosis",value\_vars = ['perimeter\_mean'])**

**plt.figure(figsize = (15,10))**

**sns.boxplot(x = "variable", y = "value", hue="diagnosis",data= melted\_data)**

**plt.show()**

**#split the data set into independent (X) and dependent (y) data sets**

**X=mydataset.iloc[:,2:31].values**

**Y=mydataset.iloc[:,1].values**

**#split the dataset into 75% training and 25% testing**

**from sklearn.model\_selection import train\_test\_split**

**X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size =0.25, random\_state =0)**

**#scale the data(feature scaling)**

**from sklearn.preprocessing import StandardScaler**

**sc = StandardScaler()**

**X\_train = sc.fit\_transform(X\_train)**

**X\_test = sc.fit\_transform(X\_test)**

**X\_train**

**Y\_train**

**X\_test**

**Y\_test**

**from sklearn.metrics import accuracy\_score**

**from sklearn.metrics import f1\_score**

**#Model Evaluation**

**#1 LOGISTIC REGRESSION**

**from sklearn.linear\_model import LogisticRegression**

**classifier = LogisticRegression(random\_state = 0)**

**#Train the model**

**classifier.fit(X\_train, Y\_train)**

**#Test it using the test dataset**

**Y\_pred1 = classifier.predict(X\_test)**

**import seaborn as sns**

**def seaborn\_plot\_cm(cm, model\_name):**

**sns.heatmap(cm, annot = True, fmt = '.2f',**

**xticklabels = ['Benign', 'Malignant'], yticklabels = ['Benign', 'Malignant'])**

**plt.ylabel('True Class')**

**plt.xlabel('Predicted Class')**

**plt.title(model\_name)**

**plt.savefig(model\_name)**

**from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report**

**def print\_classification\_report(classifier\_name, Y\_pred1, Y\_test):**

**print(classifier\_name + ' Accuracy: {:.3f}'.format(accuracy\_score(Y\_test,**

**Y\_pred1)))**

**cm = confusion\_matrix(Y\_test, Y\_pred1)**

**print(classification\_report(Y\_test, Y\_pred1))**

**seaborn\_plot\_cm(cm, classifier\_name)**

**print\_classification\_report('Logistic Regression', Y\_pred1, Y\_test)**

**cm = confusion\_matrix(Y\_test, Y\_pred1)**

**tp = cm[0,0]**

**fp = cm[0,1]**

**fn = cm[1,0]**

**tn = cm[1,1]**

**print ('True Positive >', tp)**

**print ('False Positive >', fp)**

**print ('False Negetive >', fn)**

**print ('True Negetive >', tn)**

**print ('Final prediction >', round((tp+tn)/(len(Y\_pred1))\*100,2))**

**# 1.2 roc curve**

**from sklearn.metrics import roc\_curve, roc\_auc\_score**

**from matplotlib import pyplot as plt**

**fpr, tpr, \_ = roc\_curve(Y\_test, Y\_pred1)**

**print('ROC AUC: %.3f' % roc\_auc\_score( y\_true=Y\_test, y\_score=Y\_pred1))**

**plt.clf()**

**plt.plot(fpr, tpr)**

**plt.xlabel('FPR')**

**plt.ylabel('TPR')**

**plt.title('ROC curve')**

**plt.show()**

**#2 RANDOM FOREST**

**from sklearn.ensemble import RandomForestClassifier**

**rd\_forest\_classifier = RandomForestClassifier(n\_estimators = 10,**

**criterion = 'entropy',**

**random\_state = 0)**

**rd\_forest\_classifier.fit(X\_train, Y\_train)**

**rd\_forest\_Y\_pred = rd\_forest\_classifier.predict(X\_test)**

**print\_classification\_report('Random Forest', rd\_forest\_Y\_pred, Y\_test)**

**#2.1 confusion matrix for random forest**

**cm = confusion\_matrix(Y\_test, rd\_forest\_Y\_pred)**

**tp = cm[0,0]**

**fp = cm[0,1]**

**fn = cm[1,0]**

**tn = cm[1,1]**

**print ('True Positive >', tp)**

**print ('False Positive >', fp)**

**print ('False Negetive >', fn)**

**print ('True Negetive >', tn)**

**print ('Final prediction >', round((tp+tn)/(len(rd\_forest\_Y\_pred))\*100,2))**

**#2.2 ROC**

**fpr, tpr, \_ = roc\_curve(Y\_test, rd\_forest\_Y\_pred)**

**print('ROC AUC: %.3f' % roc\_auc\_score( y\_true=Y\_test, y\_score=rd\_forest\_Y\_pred))**

**plt.clf()**

**plt.plot(fpr, tpr)**

**plt.xlabel('FPR')**

**plt.ylabel('TPR')**

**plt.title('ROC curve')**

**plt.show()**

**#3 GAUSSIAN NAIVE BAYES**

**from sklearn.naive\_bayes import GaussianNB**

**gbNB\_classifier = GaussianNB()**

**gbNB\_classifier.fit(X\_train, Y\_train)**

**gb\_NB\_Y\_pred = gbNB\_classifier.predict(X\_test)**

**print\_classification\_report('Gaussian Naive Bayes', gb\_NB\_Y\_pred, Y\_test)**

**#3.1confusion matrix**

**cm = confusion\_matrix(Y\_test, gb\_NB\_Y\_pred)**

**tp = cm[0,0]**

**fp = cm[0,1]**

**fn = cm[1,0]**

**tn = cm[1,1]**

**print ('True Positive >', tp)**

**print ('False Positive >', fp)**

**print ('False Negetive >', fn)**

**print ('True Negetive >', tn)**

**print ('Final prediction >', round((tp+tn)/(len(gb\_NB\_Y\_pred))\*100,2))**

**#3.2ROC FOR NAIVE BAYES**

**fpr, tpr, \_ = roc\_curve(Y\_test, gb\_NB\_Y\_pred)**

**print('ROC AUC: %.3f' % roc\_auc\_score( y\_true=Y\_test, y\_score=gb\_NB\_Y\_pred))**

**plt.clf()**

**plt.plot(fpr, tpr)**

**plt.xlabel('FPR')**

**plt.ylabel('TPR')**

**plt.title('ROC curve')**

**plt.show()**

**#4 DECISION TREE**

**from sklearn.tree import DecisionTreeClassifier**

**DT\_classifier = DecisionTreeClassifier(criterion = 'entropy', random\_state = 0)**

**DT\_classifier.fit(X\_train, Y\_train)**

**DT\_pred = DT\_classifier.predict(X\_test)**

**print\_classification\_report('Decision Tree', DT\_pred, Y\_test)**

**#4.1 confusion matrix**

**cm = confusion\_matrix(Y\_test, DT\_pred)**

**tp = cm[0,0]**

**fp = cm[0,1]**

**fn = cm[1,0]**

**tn = cm[1,1]**

**print ('True Positive >', tp)**

**print ('False Positive >', fp)**

**print ('False Negetive >', fn)**

**print ('True Negetive >', tn)**

**print ('Final prediction >', round((tp+tn)/(len(DT\_pred))\*100,2))**

**#4.2 ROC FOR NAIVE BAYES**

**fpr, tpr, \_ = roc\_curve(Y\_test, DT\_pred)**

**print('ROC AUC: %.3f' % roc\_auc\_score( y\_true=Y\_test, y\_score=DT\_pred))**

**plt.clf()**

**plt.plot(fpr, tpr)**

**plt.xlabel('FPR')**

**plt.ylabel('TPR')**

**plt.title('ROC curve')**

**plt.show()**

**"""**

**model = DecisionTreeClassifier()**

**model.fit(X\_train, Y\_train)**

**predictions = classifier\_6.predict\_proba(X\_test)**

**Y\_pred6 = classifier\_6.predict(X\_test)**

**print (roc\_auc\_score(X\_test, predict\_proba[:,1])**

**"""**

**#lets plot the bar graph**

**#accuracy score**

**import numpy as np**

**import matplotlib.pyplot as plt**

**ALGORITHM = [95.1, 93.71, 96.5, 94.41]**

**ACCURACY\_SCORE = ('LR', 'DT', 'RF', 'GNB')**

**y\_pos = np.arange(len(ACCURACY\_SCORE))**

**plt.bar(y\_pos, ALGORITHM, color=('r', 'b','orange', 'salmon'))**

**plt.xticks(y\_pos, ACCURACY\_SCORE)**

**plt.show()**

**#roc auc**

**ALGORITHM = [0.949, 0.942, 0.964, 0.936]**

**ROC\_AUC = ('LR', 'DT', 'RF', 'GNB')**

**y\_pos = np.arange(len(ROC\_AUC))**

**plt.bar(y\_pos, ALGORITHM, color=('g', 'r','yellow', 'b'))**

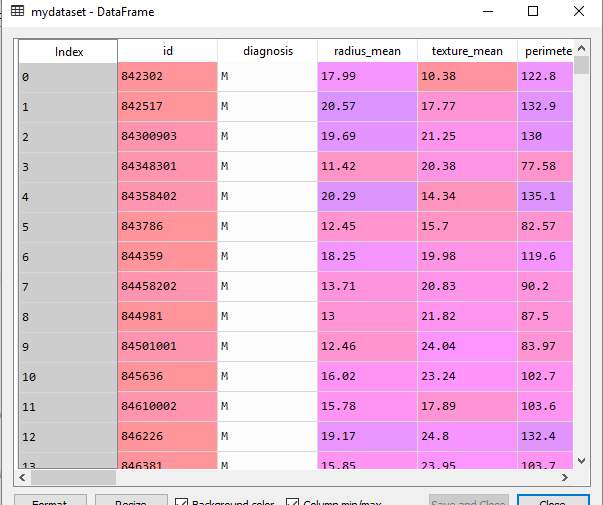
**plt.xticks(y\_pos, ROC\_AUC)**

**plt.show()**

**RESULT**

* 1. . SNAPSHOTS AND DESCRIPTION:

1. Dataset Data Frame



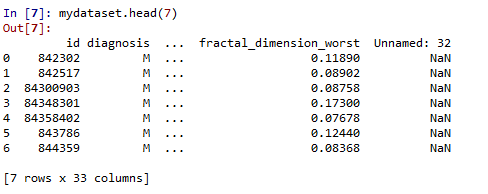
(FIG 11: DATASET LOAD)

1. SHAPE OF THE DATSET



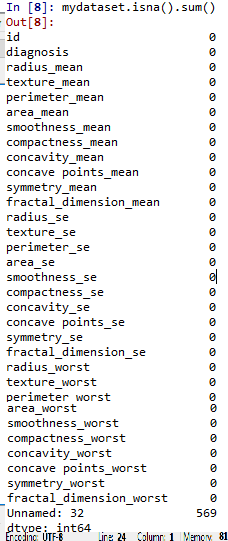
(FIG 12: SHAPE OF DATASET)

1. HEADER PART OF DATASET



(FIG 13: HEAD SECTION)

1. FINDING A NULL VALUES



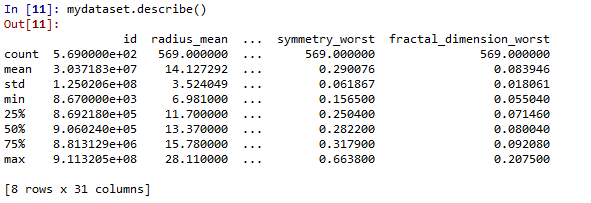
(FIG 14: PREPROCESSING)

1. DROP THE COLUMN WITH ALL MISSING VALUES



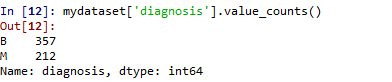
(FIG 15: DROP WITH NULL VALUES)

1. DESCRIBING DATASET



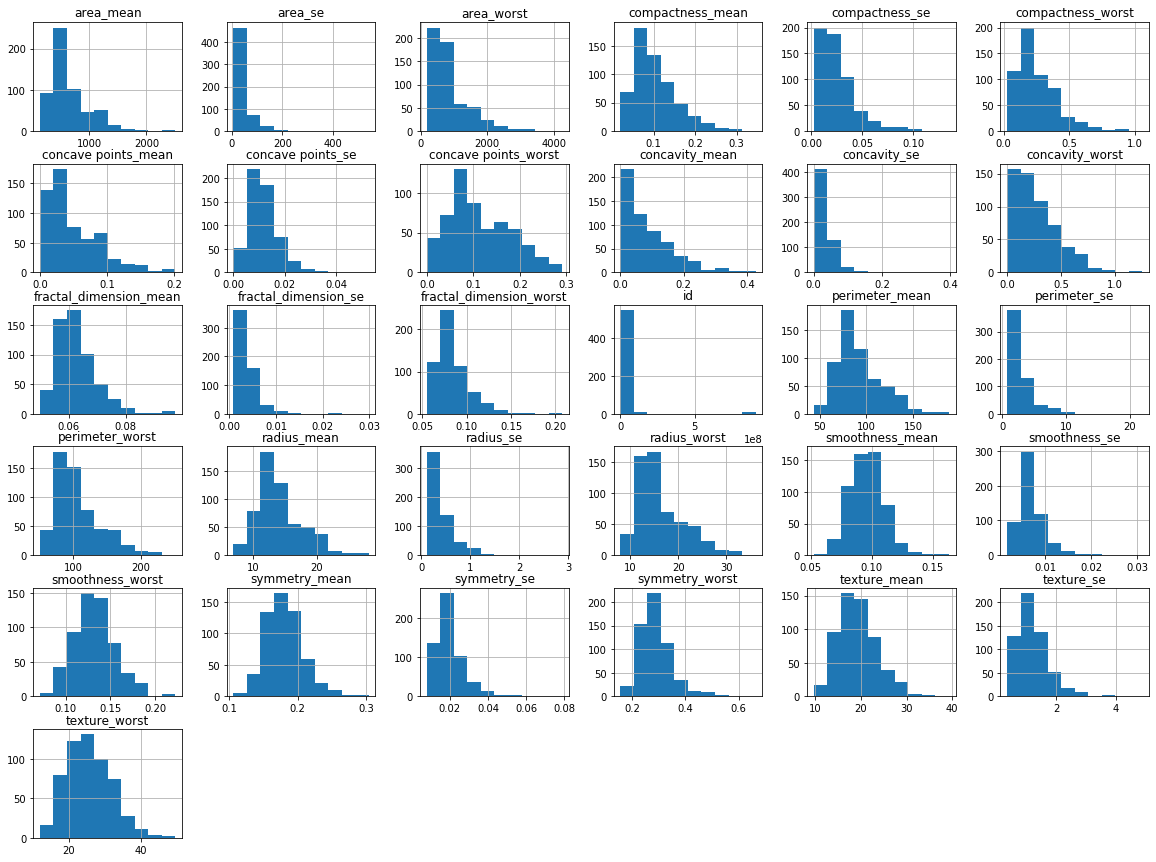
(FIG 16: DESCRIBE)

1. COUNT OF THE DIAGNOSIS



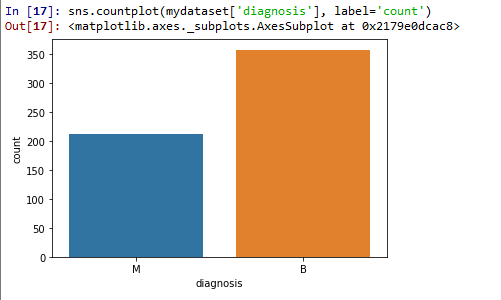
(FIG 17: COUNT OF MALIGANT AND BENGIN)

1. A HISTOGRAM IS A PLOT THAT LETS YOU DISCOVER, AND SHOW, THE UNDERLYING FREQUENCY DISTRIBUTION



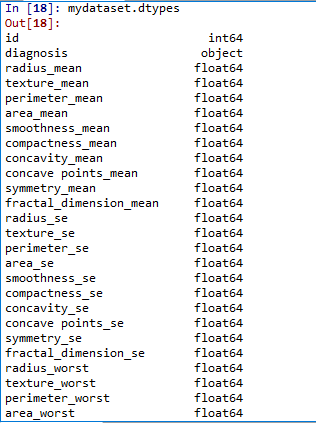
(FIG 18: #A histogram is a plot that lets you discover, and show, the underlying frequency distribution)

1. VISUALIZE THE COUNT



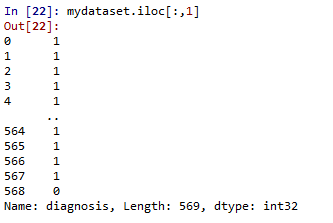
(FIG 19: VISUALIZE THE COUNT)

1. LOOK AT THE (DTYPE) DATA TYPES TO SEE WHICH COLUMNS NEED TO BE ENCODED



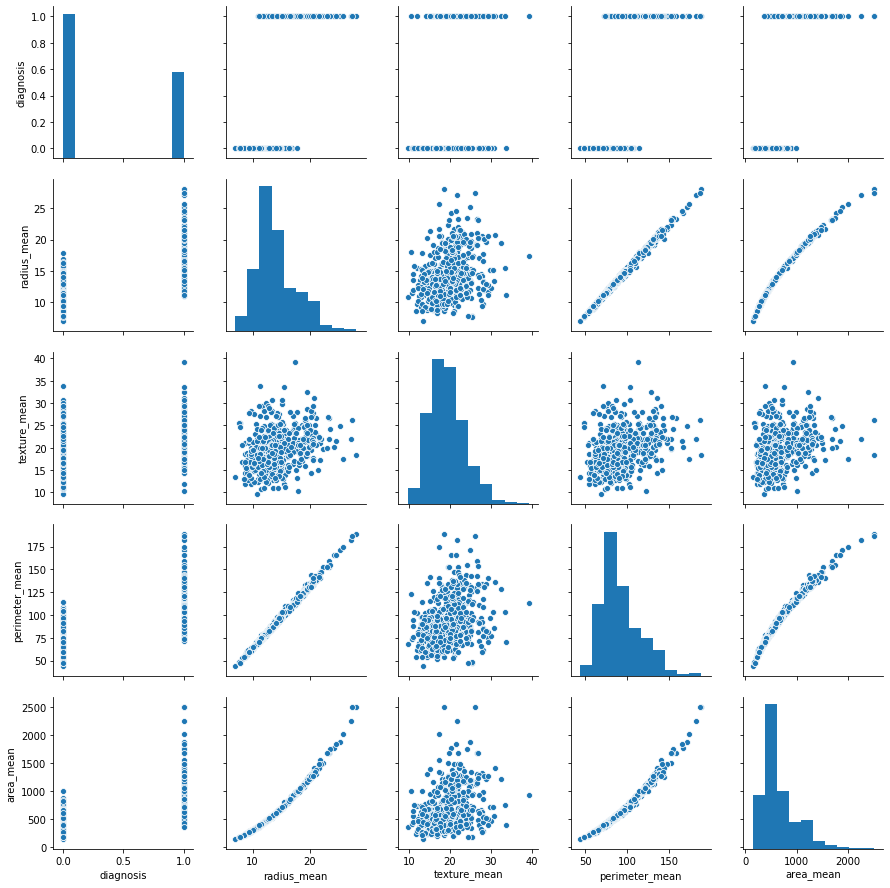
(FIG 20: DATATYPES FOR ENCODING)

1. ENCODE THE CATEGORICAL DATA VALUES



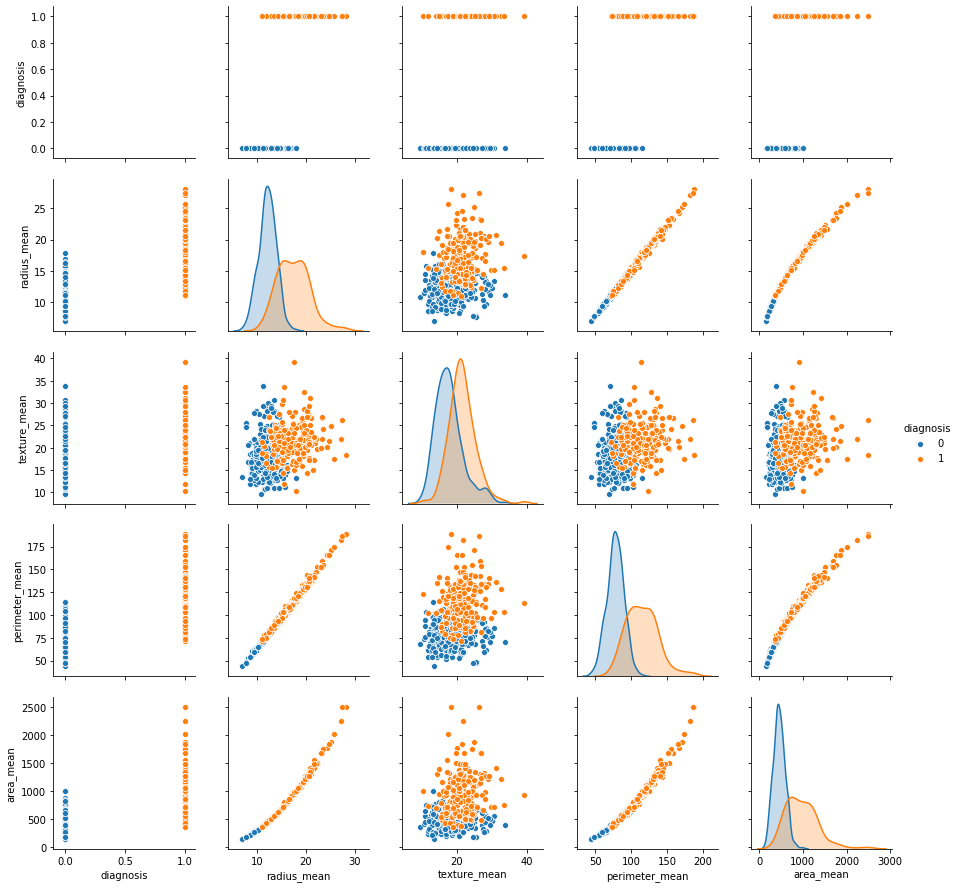
(FIG 21: ENCODE)

1. CREATE APAIR PLOT



(FIG 22: A PAIRPLOT)

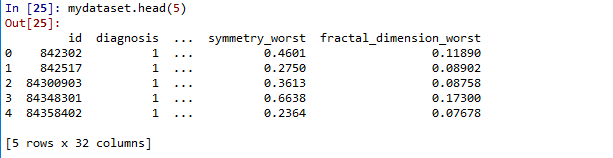
1. CREATE APAIR PLOT WITH HUE



(FIG 23: PAIRPLOT WITH HUE)

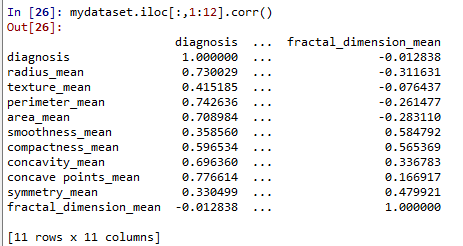
1. NEW DATA’S HEADER PART

#print the first 5 rows of the new data



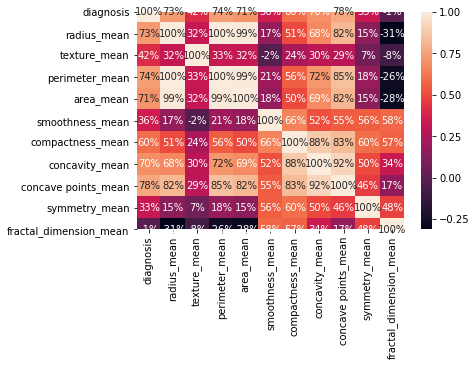
(FIG 24: HEADER)

1. THE CORRELATION OF THE COLUMNS



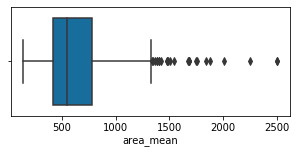
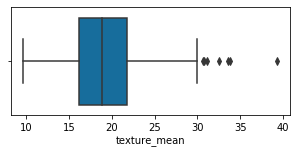
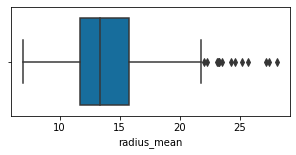
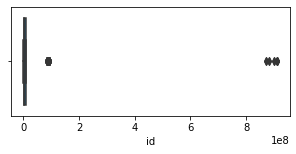
(FIG 25: CO-RELATION)

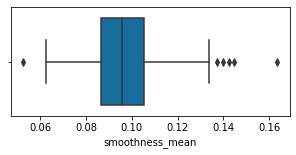
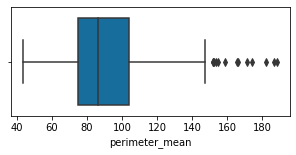
1. VISUALIZATION OF CO-RELATION

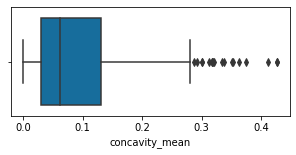
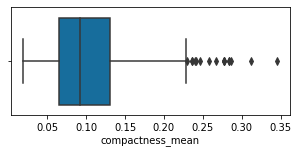


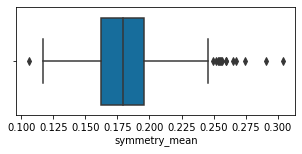
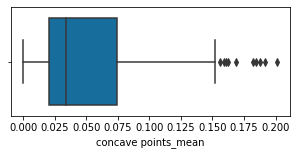
(FIG 26: VISUALIZATION OF CO-RELATION)

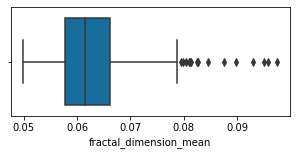
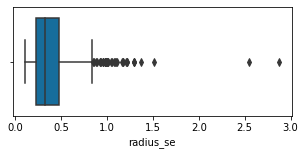
1. BOX PLOT TO CHECK OUTLIER IN EACH CATEGORY

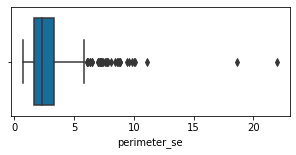
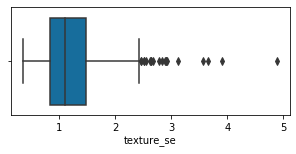


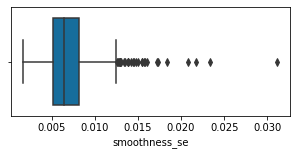
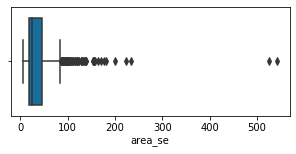


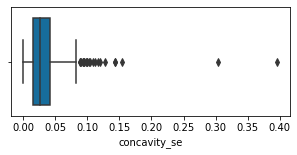
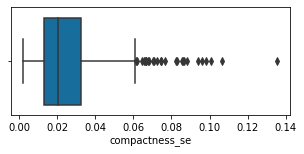


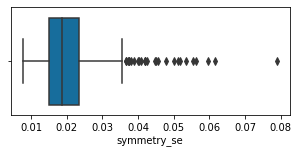
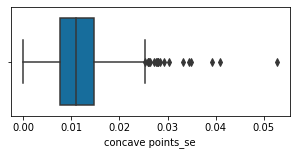


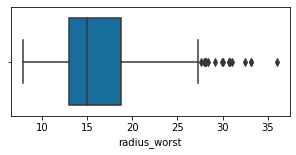
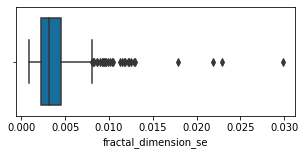


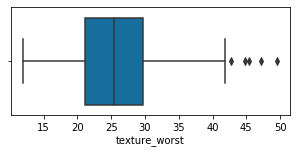
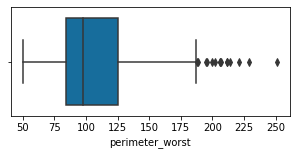


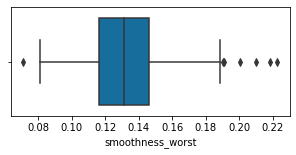
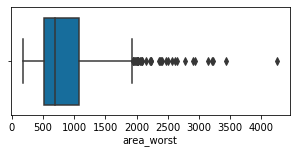


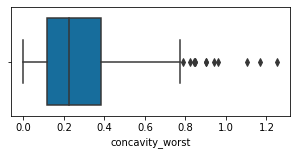
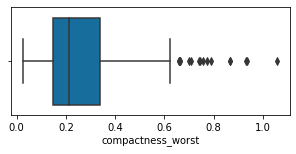


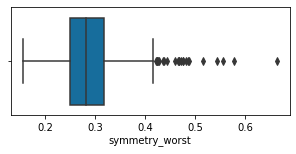
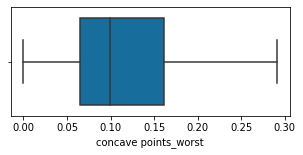


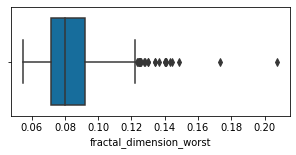




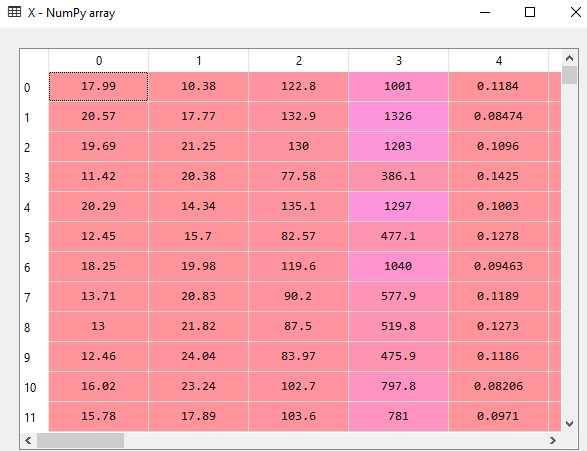


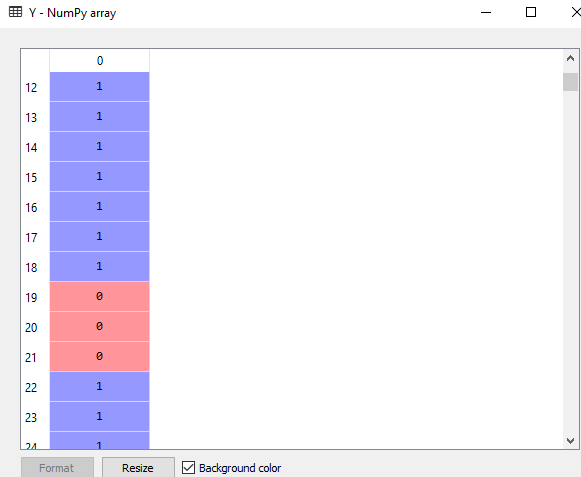




(FIG 27: BOXPLOT)

1. split the data set into independent (X) and dependent (y) data sets



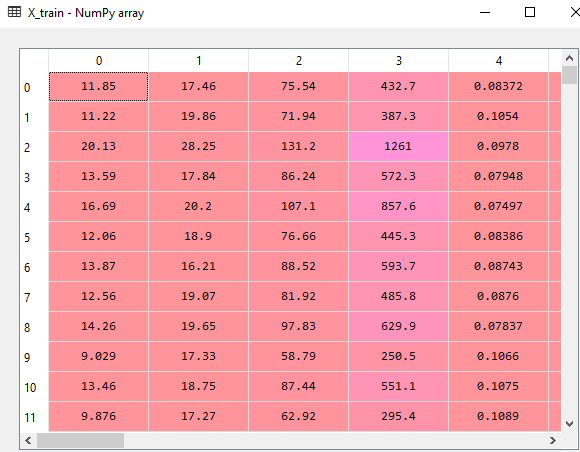


(FIG 28: DATASET OF X-SPLIT AND Y SPLIT)

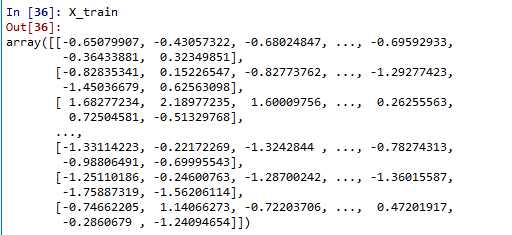
1. X-TEST AND Y-TEST:



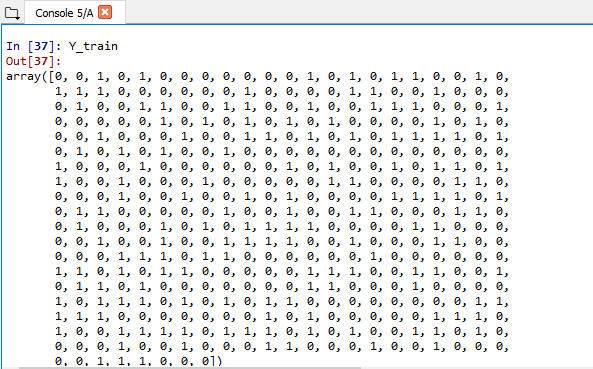
(FIG 29: 75% TRAIN AND 25% TEST)



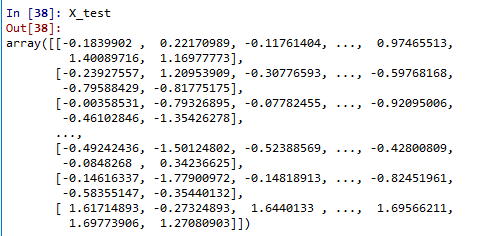
(FIG 30: X-TRAIN)



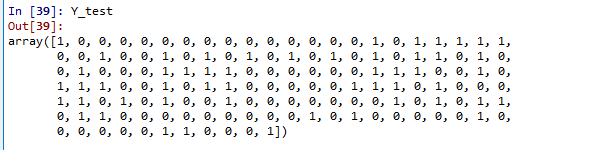
(FIG 31: X-TRAIN SCALE)



(FI 32: Y-TRAIN)

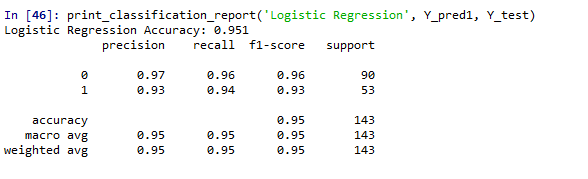


(FIG 33: X TEST)



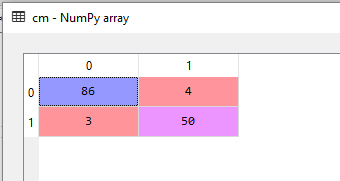
(FIG 34: Y TEST)

1. ACCURACY LEVELOF LOGISTIC REGRESSION:

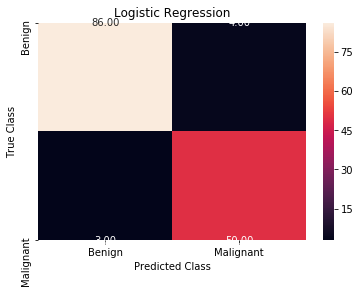


(FIG 35: ACCURACY LEVEL OF LR)

1. CONFUSION MATRIX OF LOGISTIC REGRESSION:

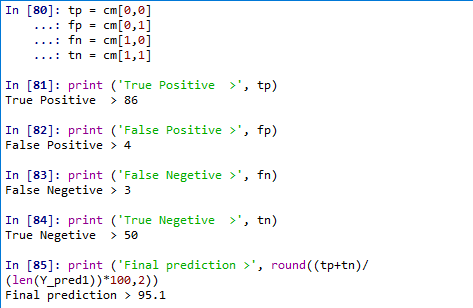
(FIG 36 CM)

1. PLOT OF LOGISTIC REGRESSION:



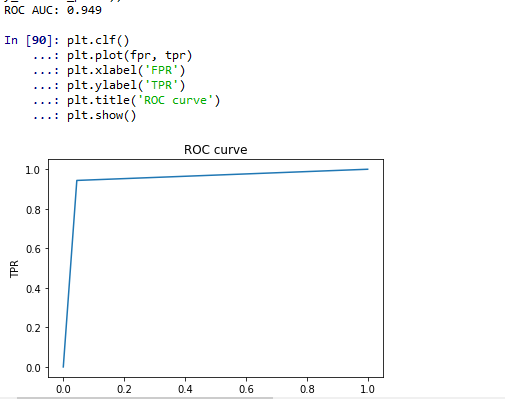
(FIG 37: PLOT OF LR)

1. FINAL PREDICTION OF LR IS: 95.1



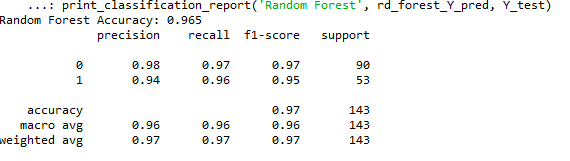
(FIG 38: FINAL PREDICTION)

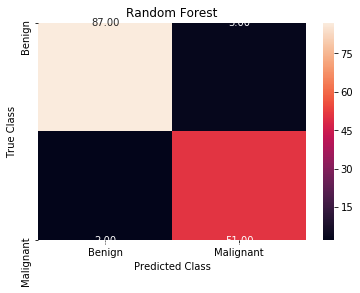
1. ROC AND AUC VALUE OF LOGISTIC REGRESSION:



(FIG 39: ROC AND AUC OF LR)

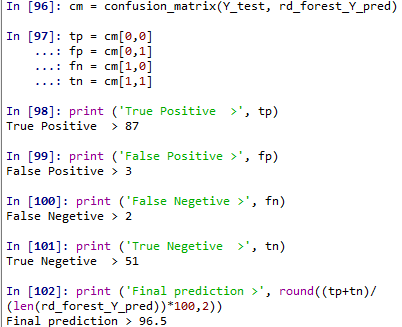
1. ACCURACY LEVEL OF RANDOM FOREST:





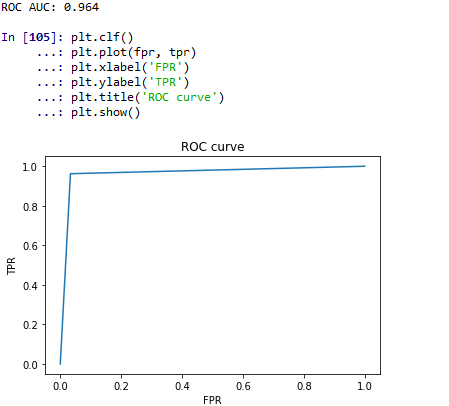
(FIG 40: ACCURACY LEVEL AND PLOT OF RF)

1. CONFUSION MATIX AND FINAL PREDICTION OF RF(96.5)



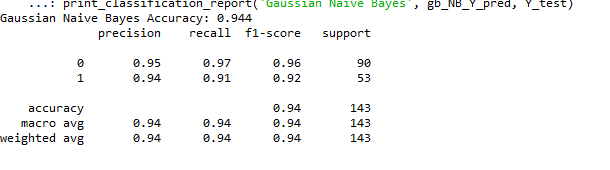
(FIG 41: FINAL PREDICTION OF RF)

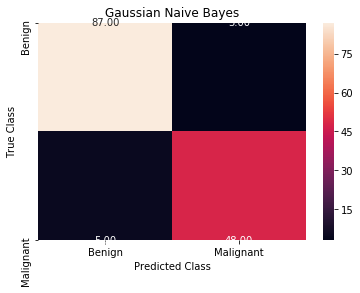
1. ROC CURVE AND AUC FOR RF:



(FIG 42: ROC CURVE AND AUC SCORE)

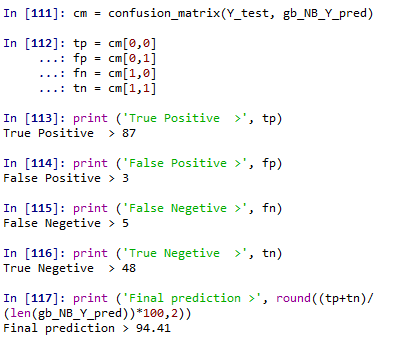
1. ACCURACY LEVEL OF GAUSSION NAVIE BAYES:





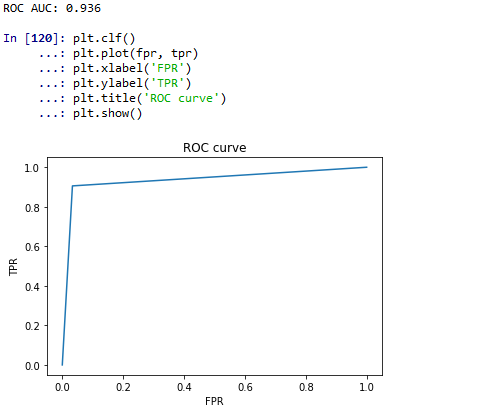
(FIG 43: GAUSSION NAVIE BAYES ACCURACY & PLOT OF NB)

1. CONFUSION MATRIX AND FINAL PREDICTION OF GNB



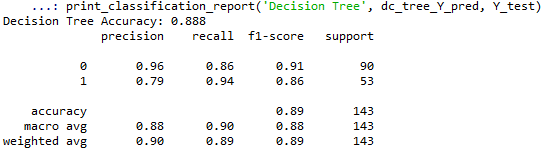
(FIG 44: FINAL PREDICTION OF GNB)

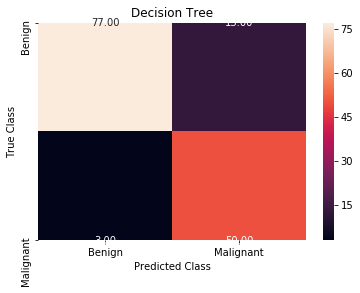
1. ROC CURVE FOR GNB AND AUC SCORE(0.936)



(FIG 45: ROC AND AUC SCORE OF GNB)

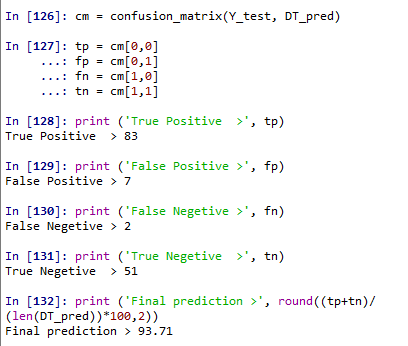
1. ACCURACY OF DECISION TREE





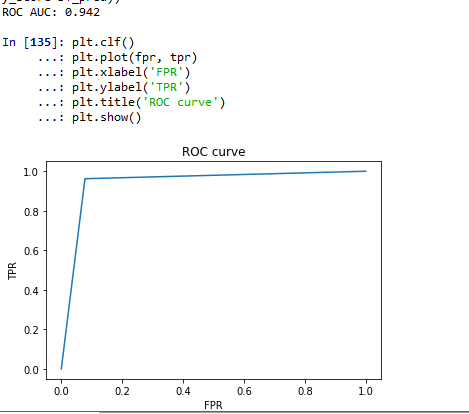
(fig 46: decision tree ACCURACY AND PLOT)

1. CONFUSION MATRIX AND FINAL PREDICTION OG DECISION TREE(93.71)



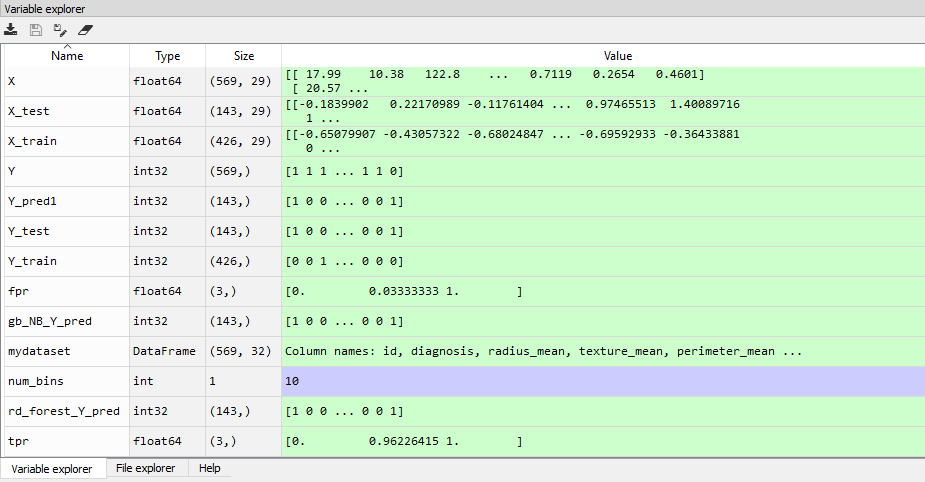
(FIG 47: FINAL PREDICTION OF DECISION TREE)

1. ROC AND AUC SCORE FOR DT:



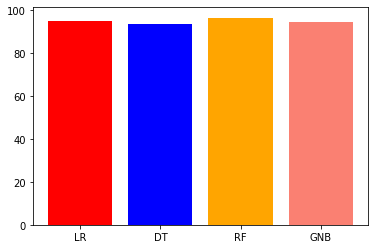
(FIG 48: ROC AND AUC SCORE FOR DT)

1. VARIABLE EXPLORES:



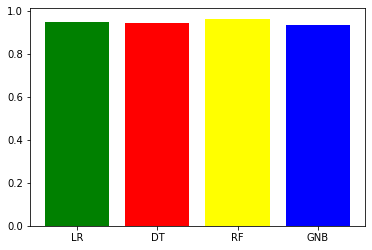
(FIG 49: VARIABLE EXPLORER)

1. GRAPH OF ACCURACY SCORE:



(FIG 50: GRAPH OF ACCURACY SCORE)

1. GRAPH OF ROC\_AUC SCORE:



(FIG 51: GRAPH OF ROC\_AUC\_ SCORE