INTERNSHIP REPORT

On

BUILDING A CLASSIFIER MODEL FOR DIABETICS

Bachelor of Technology

in

Department of Electronics and Computer Engineering

by

D.CHANDANA(180050012)

under the supervision of

Mr.N.Srinivasulu

Assistant Professor

(DEEMED TO BE UNIVERSITY)

Koneru Lakshmaiah Education Foundation

(Deemed to be University estd., u/s 3 of UGC Act 1956)

Greenfields, Vaddeswaram, Guntur (Dist.), Andhra Pradesh - 522502

August, 2020

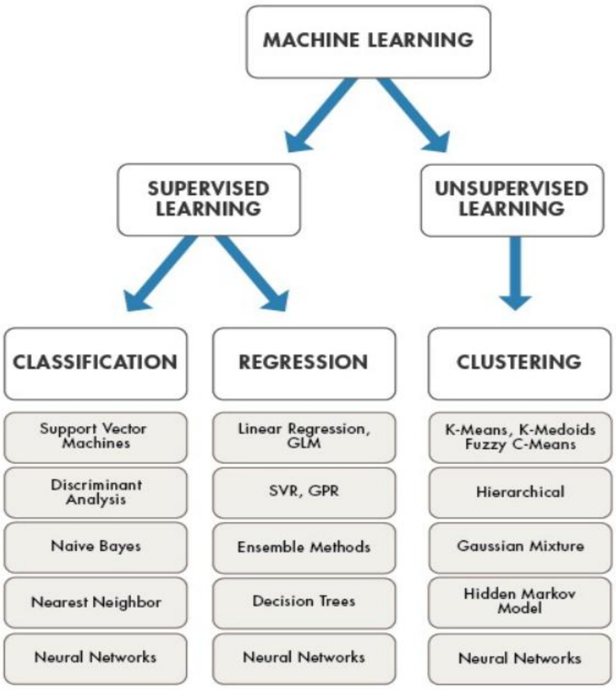
Table of Contents

ABSTRACT

In this project we built a predictive machine learning classifier model to predict diabetes based on the given diagnostics of the person. These are the revolution days of Artificial Intelligence. Machine Learning is a subset of Artificial Intelligence, it is area of computational science that focuses on analyzing and interpreting the patterns and structures of the data to enable learning, reasoning, and decision making for the machine outside of human interaction. In this project we gave a set of data to machine and built a classifier model to run, train, and test the data which makes the machine understands the input and output of the patients who are diabetic and non-diabetic. We take the data, firstly run the data, secondly visualize the data, thirdly split the data to train and test, then make that data undergo algorithms, and produce the results.

INTRODUCTION

The main Objective of this project to build a classifier model for diabetics to make the machine predict whether the patient has diabetes or not based on diagnostics which is trained with the classifier mode with machine learning using python. Machine learning is a subset of Artificial Intelligence, a scientific field dealing with the ways in which machines learn from experience. The purpose of machine learning is the construction of computer systems that can adapt and learn from their experience. There are three major types of Machine learning: Supervised Learning, Unsupervised Learning, and Reinforcement Learning.



Supervised Learning:

In Supervised learning, the system must “learn” inductively a function called target function, which is an expression of a model describing the data. The objective function is used to predict the value of a variable, called dependent variable or output variable, from a set of variables, called independent variables or input variables or characteristics or features. The set of possible input values of the function, i.e. its domain, are called instances.

There are mainly two kinds of learning tasks: Classification and regression

Most of the common classification techniques are:

1. Decision Trees
2. Instance Based Learning
3. K -Nearest Neighbors (KNN)
4. Genetic Algorithms
5. Artificial Neural Networks (ANN)
6. Support Vector Machine (SVM)
7. Naive Baye's Theorem
8. Logistic Regression

The techniques in Regression are:

1. Linear Regression
2. Polynomial Regression and etc..

The data should under go different processes to produce a accurate result

1. Data Collection:

The process of gathering data depends on the type of project we desire to make, we should collect real-time large amounts of data. But the collected data cannot be used directly for performing the analysis process as there might be a lot of missing data, extremely large values, unorganized text data or noisy data. To solve this data preparation is done. The data will be taken in forms of datasets.

2. Data Pre-processing:

Data pre-processing is a process of cleaning the raw data i.e. the data is collected in the real world and is converted to a clean data set. It is one of the most important steps in machine learning. It is the most important step that helps in building machine learning models more accurately.

In machine learning, there is an 80/20 rule. Every data scientist should spend 80% time for data pre-processing and 20% time to actually perform the analysis.

These steps are executed to convert the data into a small clean data set, this part of the process is called as data pre-processing. This is also called data preparation, we do splitting and normalization of data here.

3. Data Visualization:

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. The modules that are used to visualize the data are pandas, matplotlib, and seaborn.

4. Training and Testing the Data:

Before training and testing of data we should remove the null (if any). We split the data into 80:20 ratio i.e 80% data is used for training purpose and 20% of data is used for testing purpose.

5. Evaluating Machine learning model:

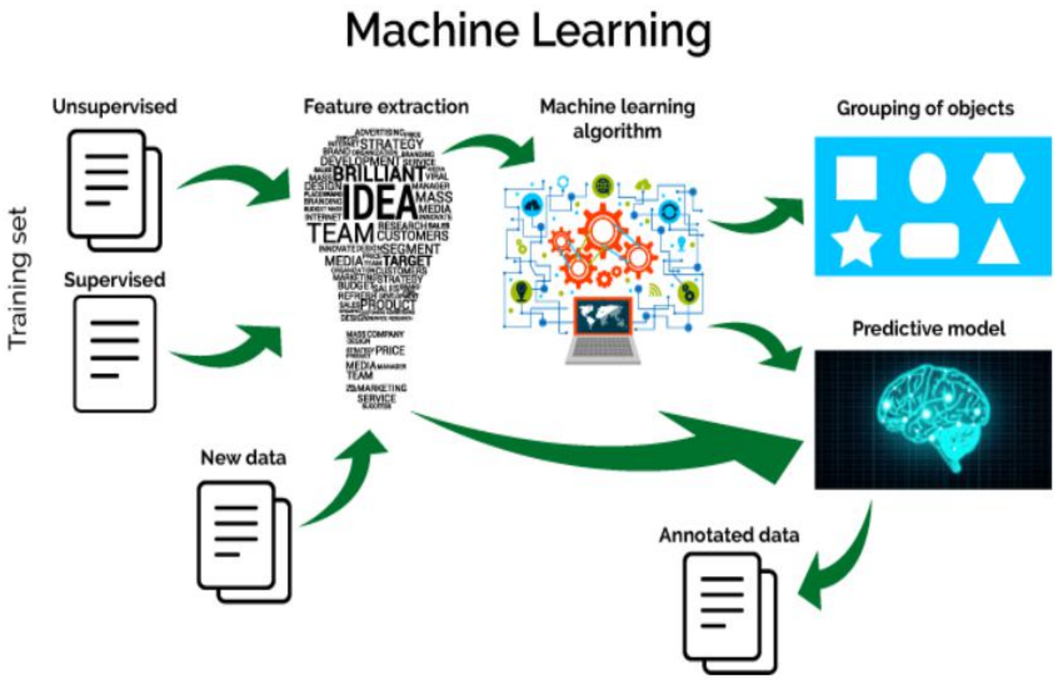
We can now train our classification model. We can use classification techniques like K-Nearest Neighbors, SVM, Naive baye's theorem, Random Forest, and Logistic regression.

Once the model is trained we can develop a confusion matrix, this tells us how well our model is trained. A confusion matrix has 4 parameters, which are 'True positives', 'True Negatives', 'False Positives' and 'False Negative'.

- **True positives** : These are cases in which we predicted TRUE and our predicted output is correct.

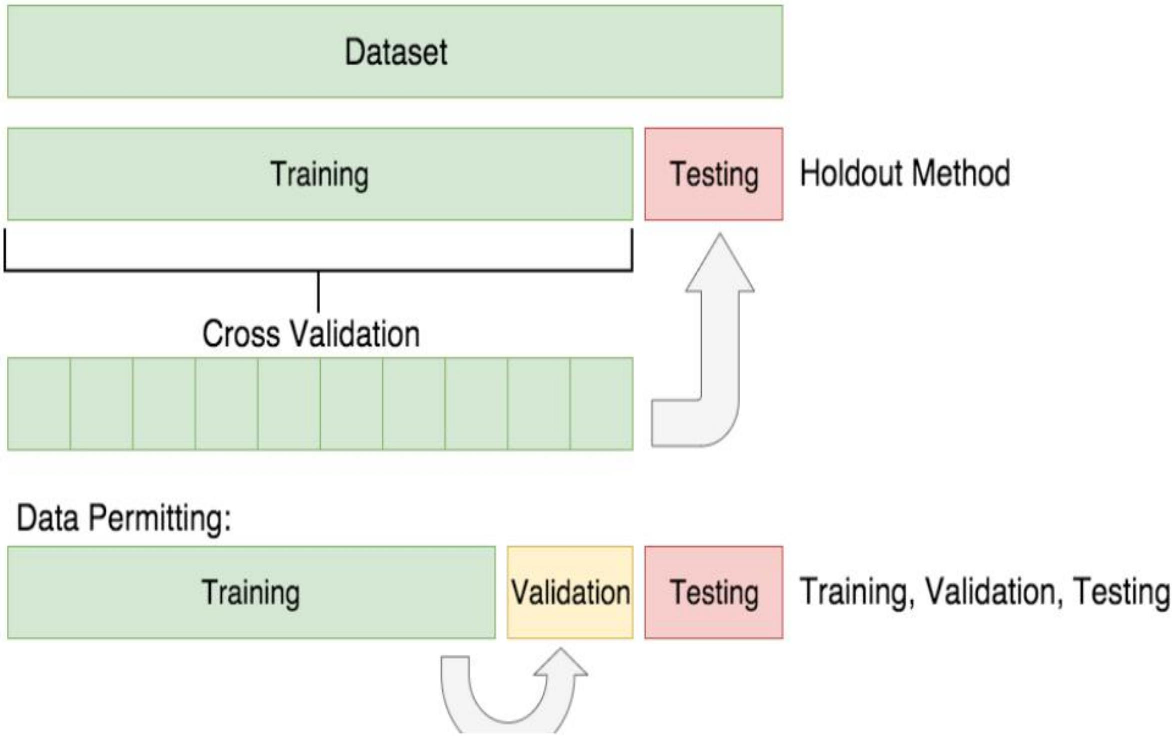
- **True negatives**:We predicted FALSE and our predicted output is correct.
- **False positives** :We predicted TRUE,but the actual predicted output is FALSE.
- **False negatives**:We predicted FALSE,but the actual predicted output is TRUE.

Accuracy = (True Positives +True Negatives) / (Total number of classes)

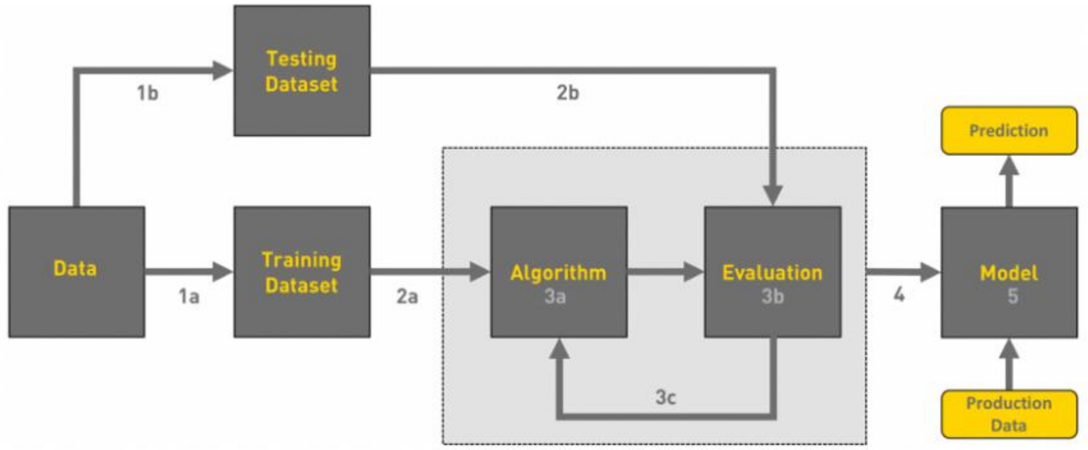


PROPOSED WORK

In the project we should apply or follow the flow of data mentioned in above section. The data should be collected first, then the data should undergo data pre-processing techniques, then data should be visualized for the sake of more understanding, then the data should be split into train and test set, and then the model should be evaluated, at last the accuracy test and the classifier model should be tested on unseen data



WORKING OF PROPOSED ARCHITECTURE

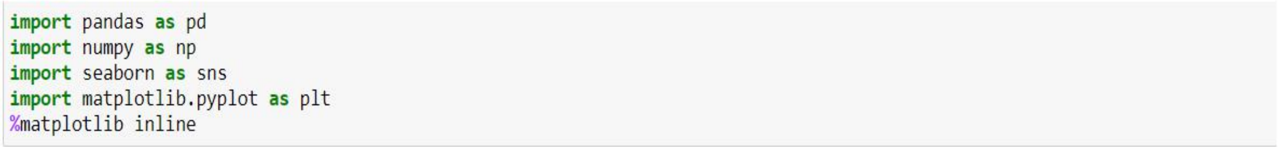


Overview of the Workflow of ML

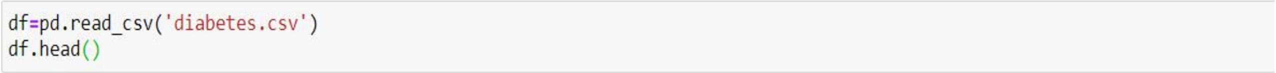
As shown in the figure the data which has already undergone through the first phases of the of the flow will be divided into two sets training and testing set. The training set will be used to train the data, the selected algorithms will be applied on the training set and that will be evaluated, We take many algorithms to train the data the evaluated data will be taken and tested for the accuracy of the algorithm and that algorithm will be picked. This will become a model and now we will send the testing set of data to test the model how much accurate results it is producing. Then this classifier model will be used to predict the diabetes. Here the accuracy score is measured using the confusion matrix and we also used ROC AUC i.e reciever operating characteristics and Area under curve is a performance measurement for classification problem at various thresholds settings. ROC is a probability curve and AUC represents degree or measure of separability. An excellent model has AUC near to the 1 which means it has good measure of separability.

RESULTS

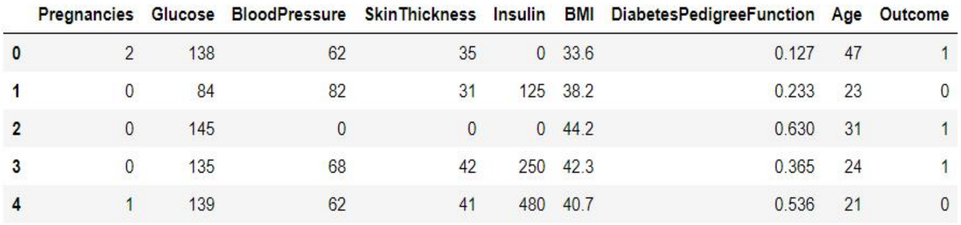
Importing modules and reading files:



```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

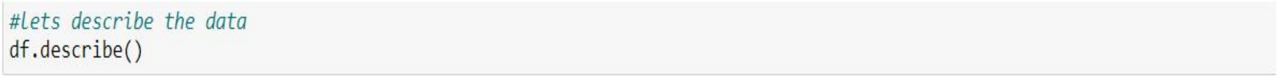


```
df=pd.read_csv('diabetes.csv')
df.head()
```

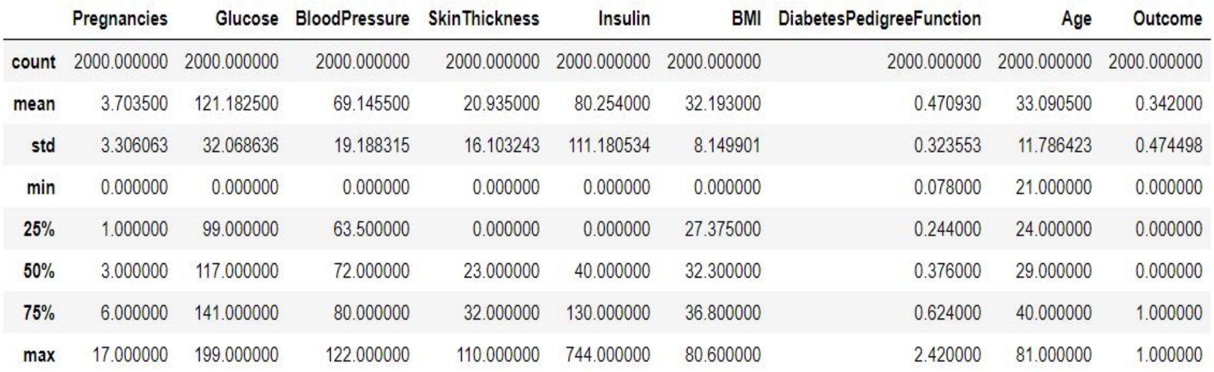


	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	2	138	62	35	0	33.6	0.127	47	1
1	0	84	82	31	125	38.2	0.233	23	0
2	0	145	0	0	0	44.2	0.630	31	1
3	0	135	68	42	250	42.3	0.365	24	1
4	1	139	62	41	480	40.7	0.536	21	0

Describing the data:

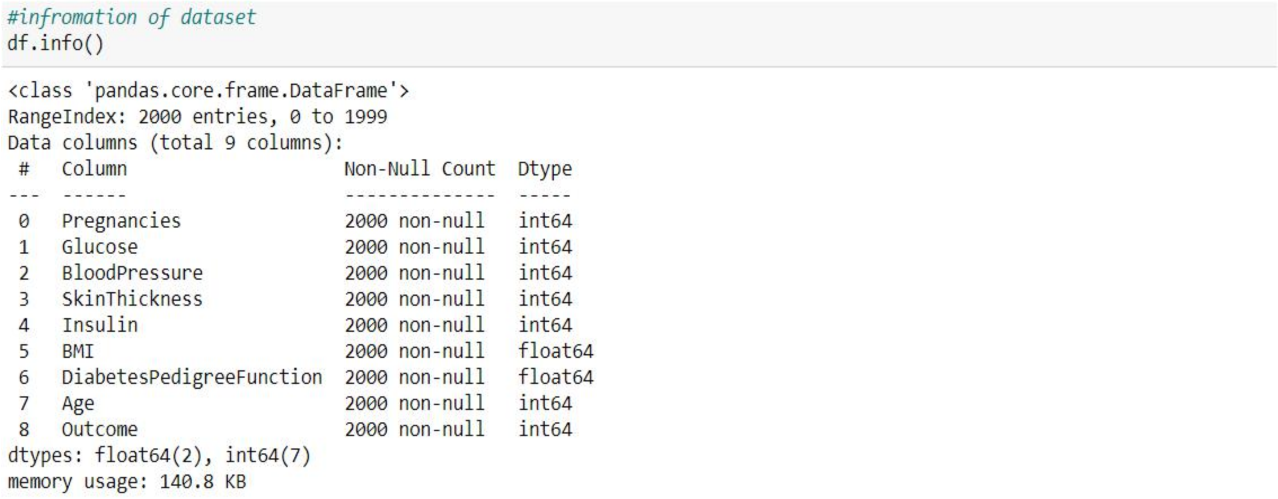


```
#lets describe the data
df.describe()
```



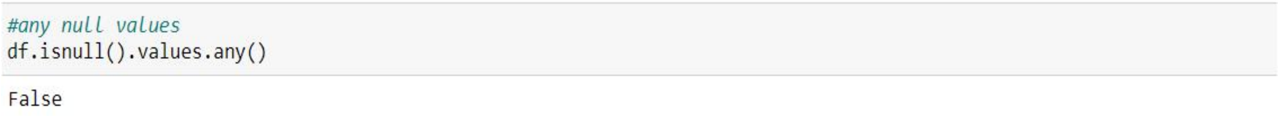
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	3.703500	121.182500	69.145500	20.935000	80.254000	32.193000	0.470930	33.090500	0.342000
std	3.306063	32.068636	19.188315	16.103243	111.180534	8.149901	0.323553	11.786423	0.474498
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	63.500000	0.000000	0.000000	27.375000	0.244000	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	40.000000	32.300000	0.376000	29.000000	0.000000
75%	6.000000	141.000000	80.000000	32.000000	130.000000	36.800000	0.624000	40.000000	1.000000
max	17.000000	199.000000	122.000000	110.000000	744.000000	80.600000	2.420000	81.000000	1.000000

Information about dataset and checking if any null values:



```
#infromation of dataset
df.info()
```

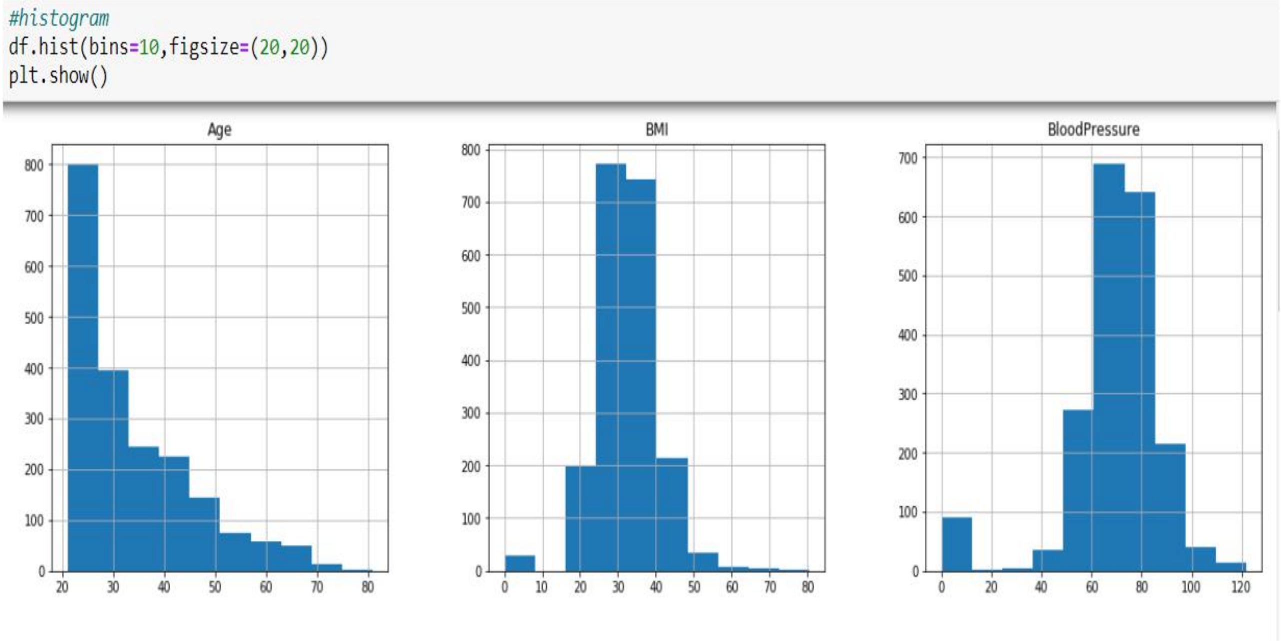
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               2000 non-null   int64  
 1   Glucose                   2000 non-null   int64  
 2   BloodPressure             2000 non-null   int64  
 3   SkinThickness             2000 non-null   int64  
 4   Insulin                   2000 non-null   int64  
 5   BMI                       2000 non-null   float64
 6   DiabetesPedigreeFunction  2000 non-null   float64
 7   Age                       2000 non-null   int64  
 8   Outcome                   2000 non-null   int64  
dtypes: float64(2), int64(7)
memory usage: 140.8 KB
```



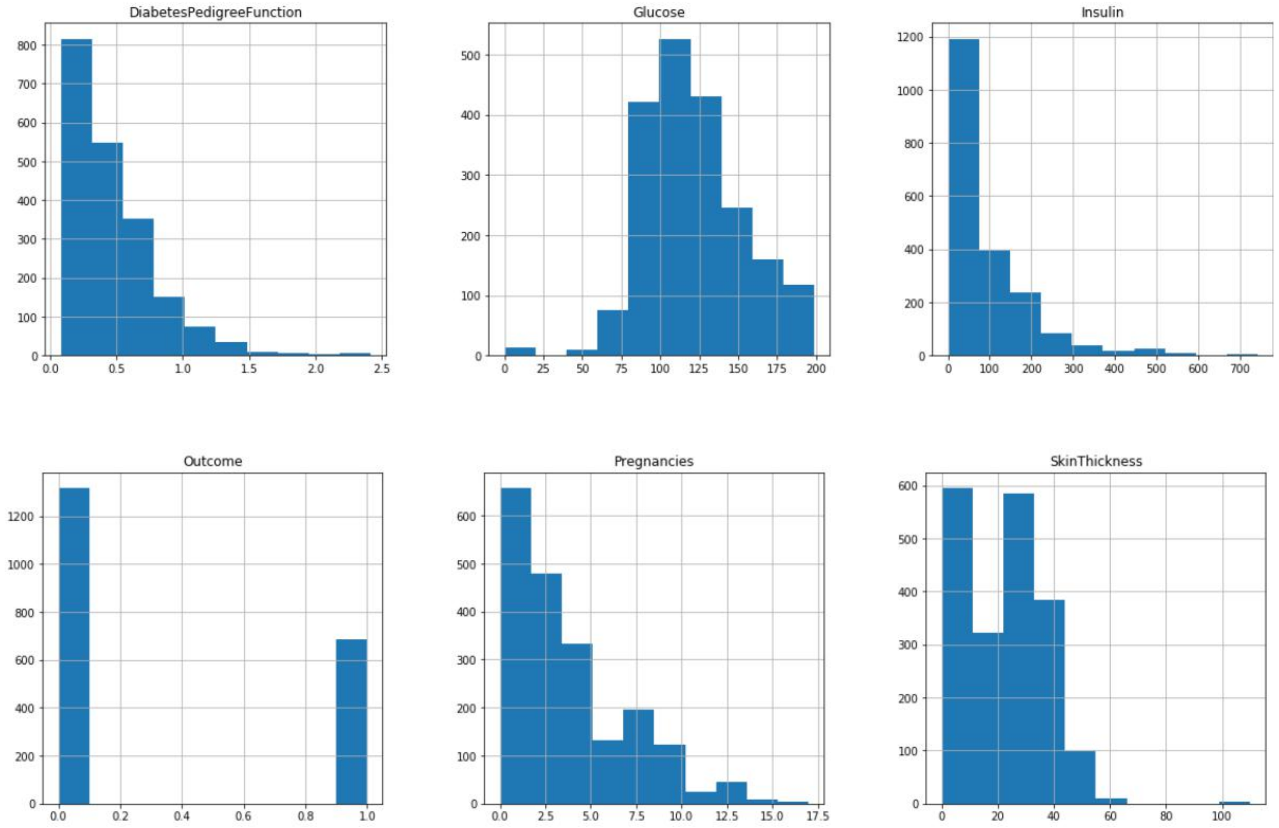
```
#any null values
df.isnull().values.any()
```

```
False
```

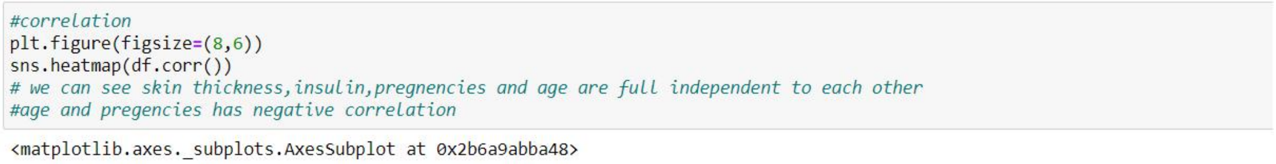
Data Visualization:



```
#histogram
df.hist(bins=10,figsize=(20,20))
plt.show()
```

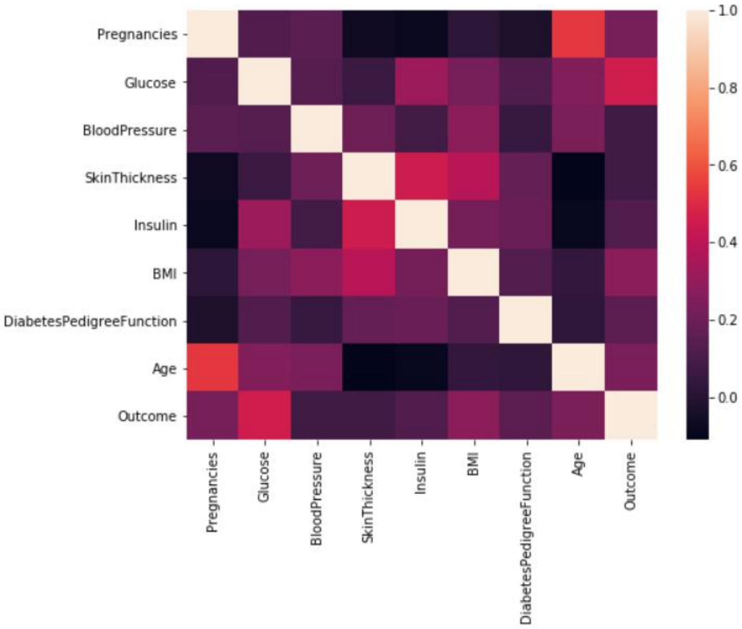


Correlation:

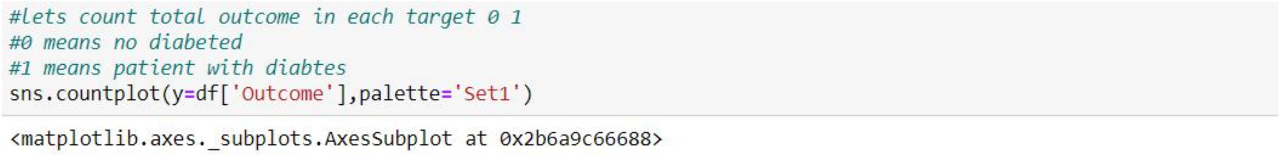


```
#correlation
plt.figure(figsize=(8,6))
sns.heatmap(df.corr())
# we can see skin thickness,insulin,pregnencies and age are full independent to each other
#age and pregencies has negative correlation
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2b6a9abba48>
```

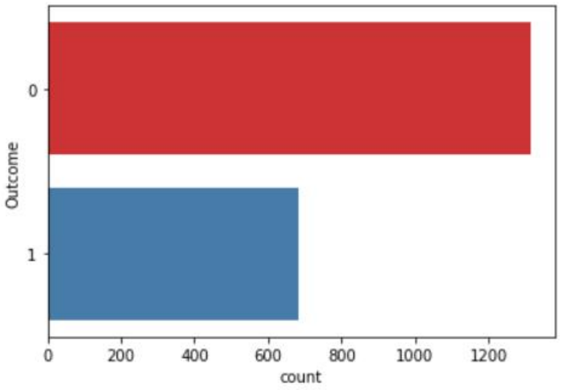


No.of diabetic and non-diabetic patients:

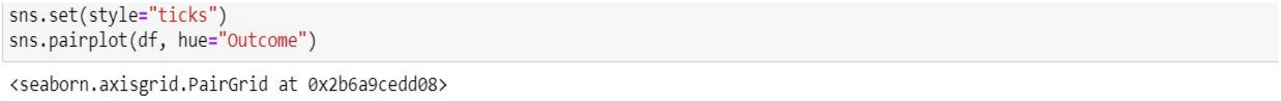


```
#lets count total outcome in each target 0 1
#0 means no diabeted
#1 means patient with diabtes
sns.countplot(y=df['Outcome'],palette='Set1')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2b6a9c66688>
```

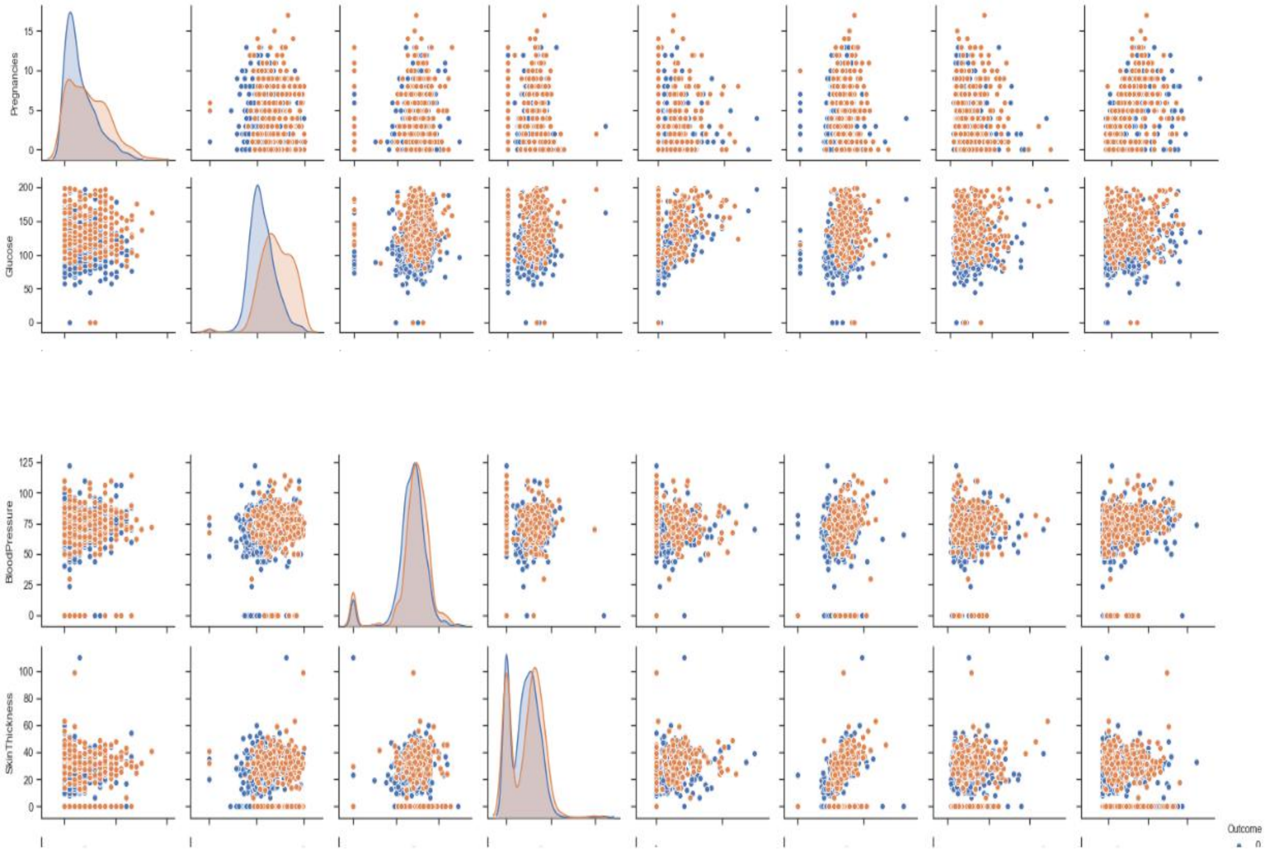


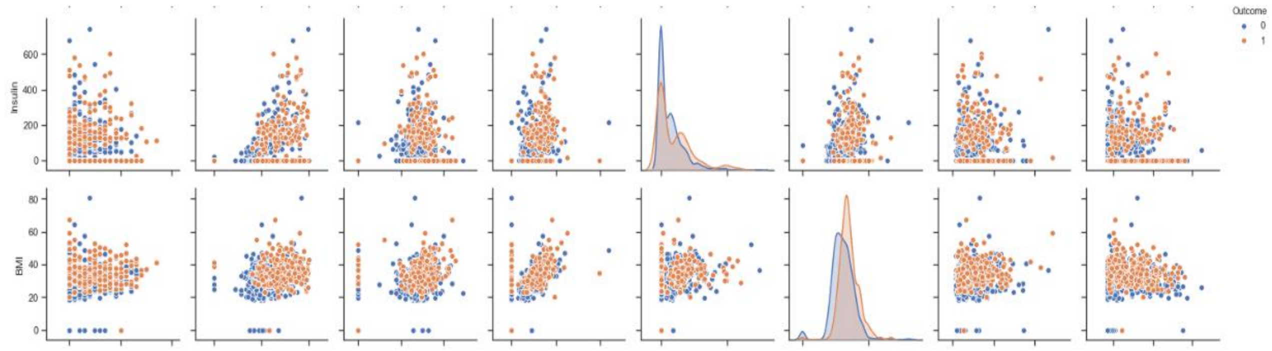
Scatter plot:

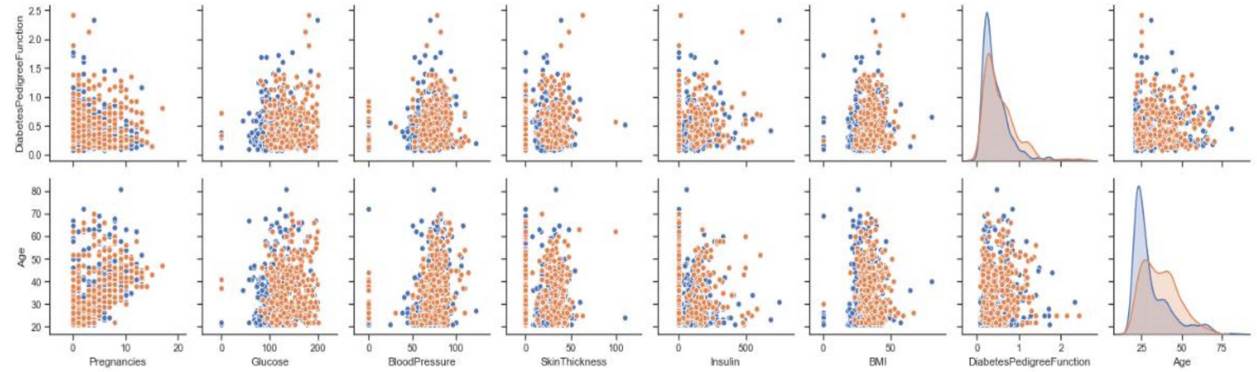


```
sns.set(style="ticks")
sns.pairplot(df, hue="Outcome")
```

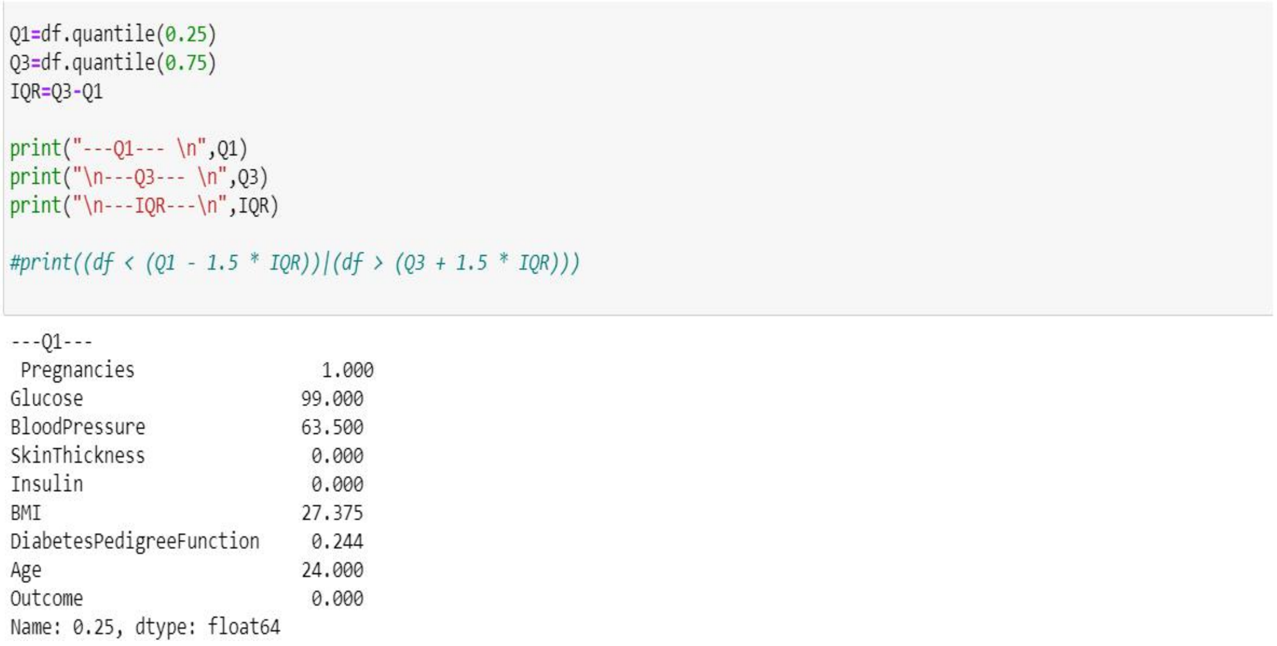
```
<seaborn.axisgrid.PairGrid at 0x2b6a9cedd08>
```





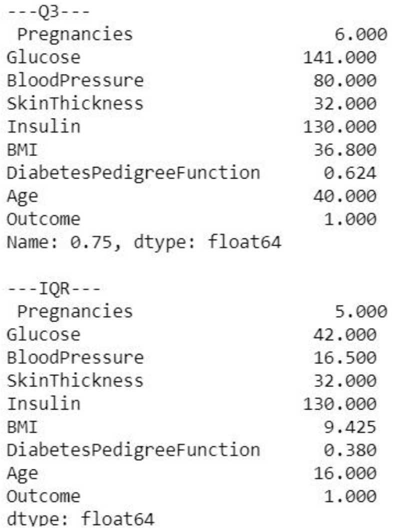


Calculating Inter-Quartile Range:(removing outlier)



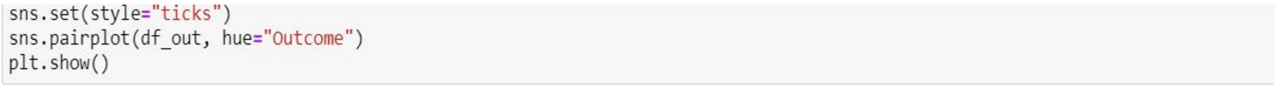
```
Q1=df.quantile(0.25)
Q3=df.quantile(0.75)
IQR=Q3-Q1

print("---Q1--- \n",Q1)
print("\n---Q3--- \n",Q3)
print("\n---IQR---\n",IQR)

#print((df < (Q1 - 1.5 * IQR))|(df > (Q3 + 1.5 * IQR)))
```

```
---Q1---
 Pregnancies                  1.000
Glucose                     99.000
BloodPressure               63.500
SkinThickness                0.000
Insulin                      0.000
BMI                         27.375
DiabetesPedigreeFunction     0.244
Age                         24.000
Outcome                      0.000
Name: 0.25, dtype: float64
```

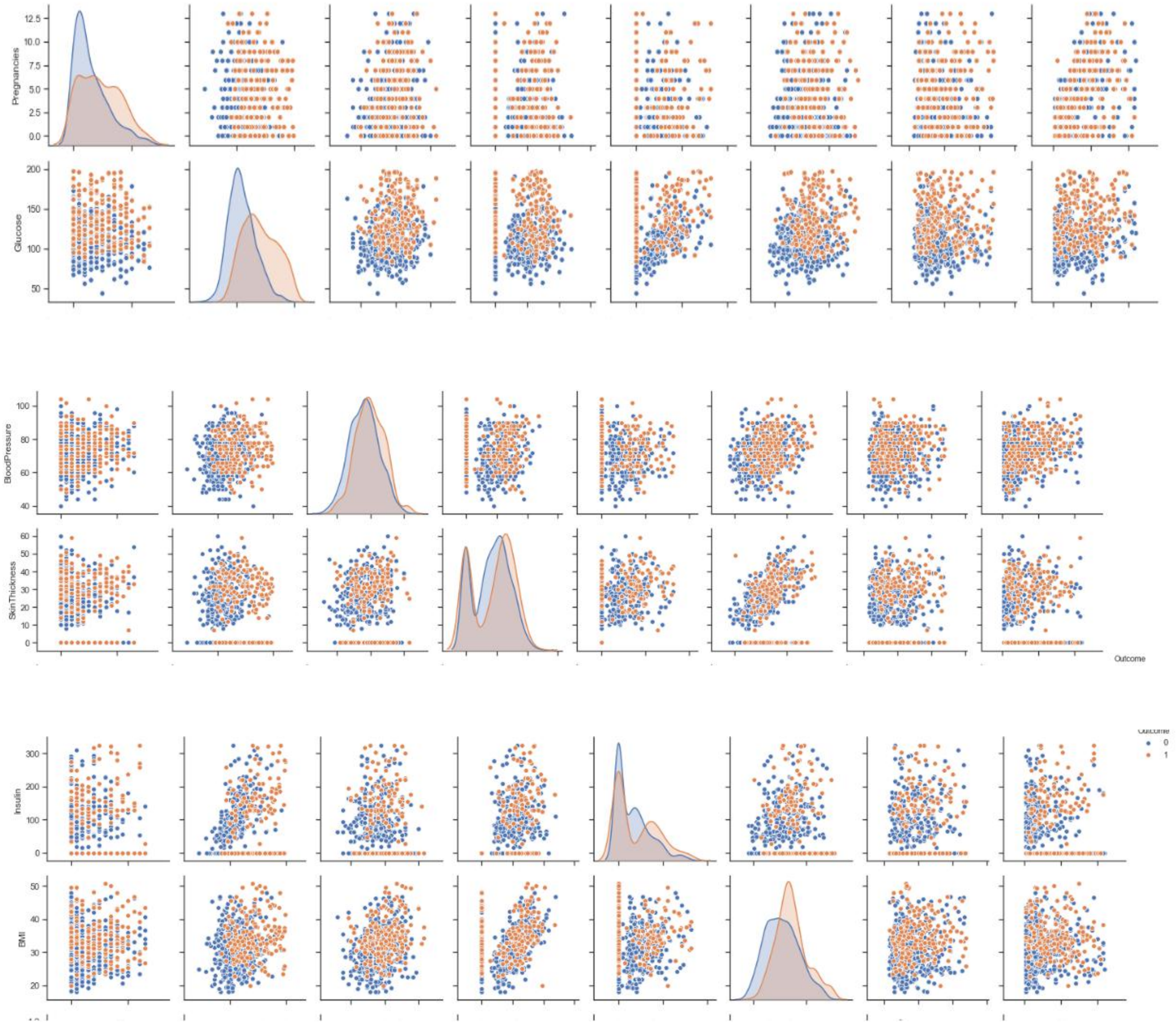


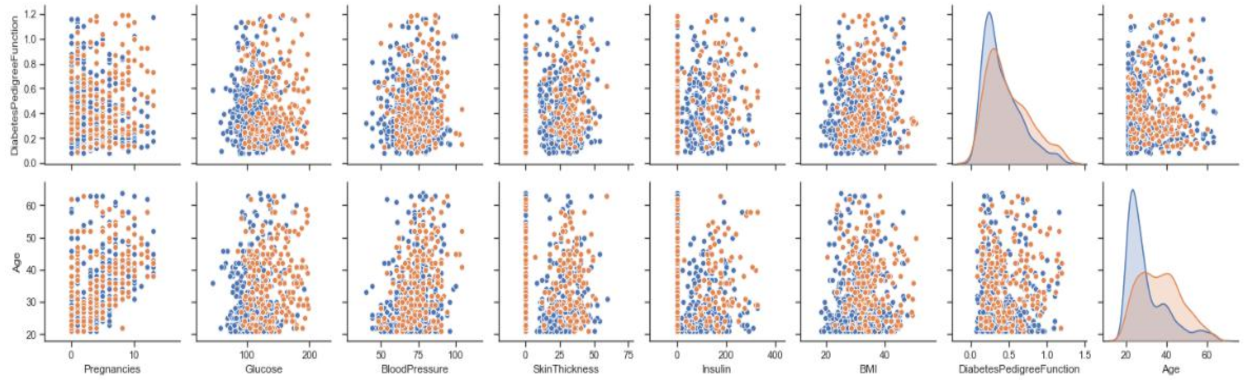
```
---Q3---
 Pregnancies                   6.000
Glucose                     141.000
BloodPressure                80.000
SkinThickness                32.000
Insulin                     130.000
BMI                          36.800
DiabetesPedigreeFunction      0.624
Age                          40.000
Outcome                       1.000
Name: 0.75, dtype: float64

---IQR---
 Pregnancies                   5.000
Glucose                      42.000
BloodPressure                16.500
SkinThickness                32.000
Insulin                     130.000
BMI                           9.425
DiabetesPedigreeFunction      0.380
Age                          16.000
Outcome                       1.000
dtype: float64
```

Scatter plot: (after removing outlier)

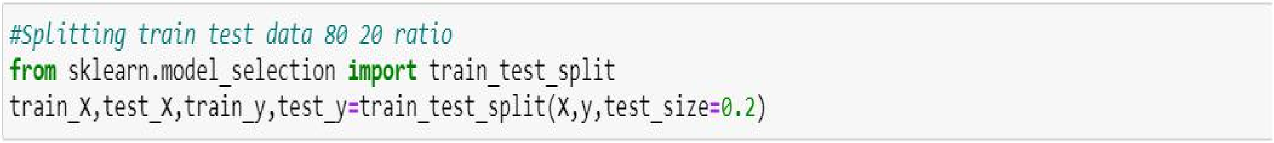


```
sns.set(style="ticks")
sns.pairplot(df_out, hue="Outcome")
plt.show()
```

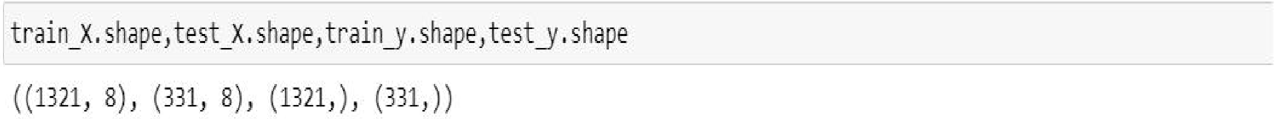




Splitting data into train and test set:



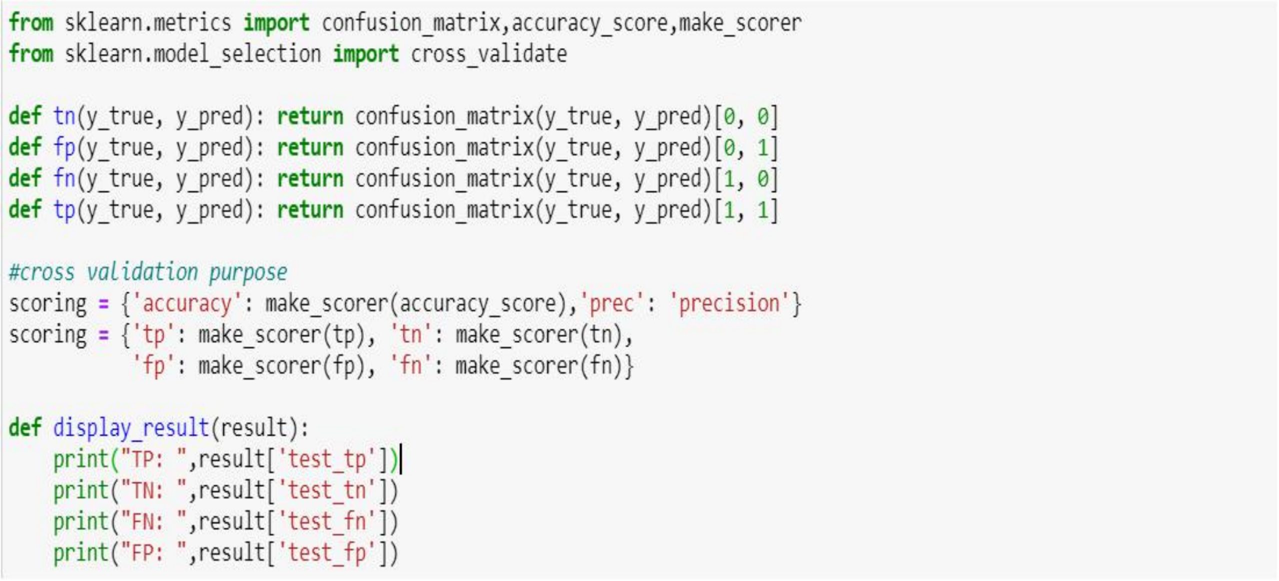
```
#Splitting train test data 80 20 ratio
from sklearn.model_selection import train_test_split
train_X,test_X,train_y,test_y=train_test_split(X,y,test_size=0.2)
```



```
train_X.shape,test_X.shape,train_y.shape,test_y.shape
```

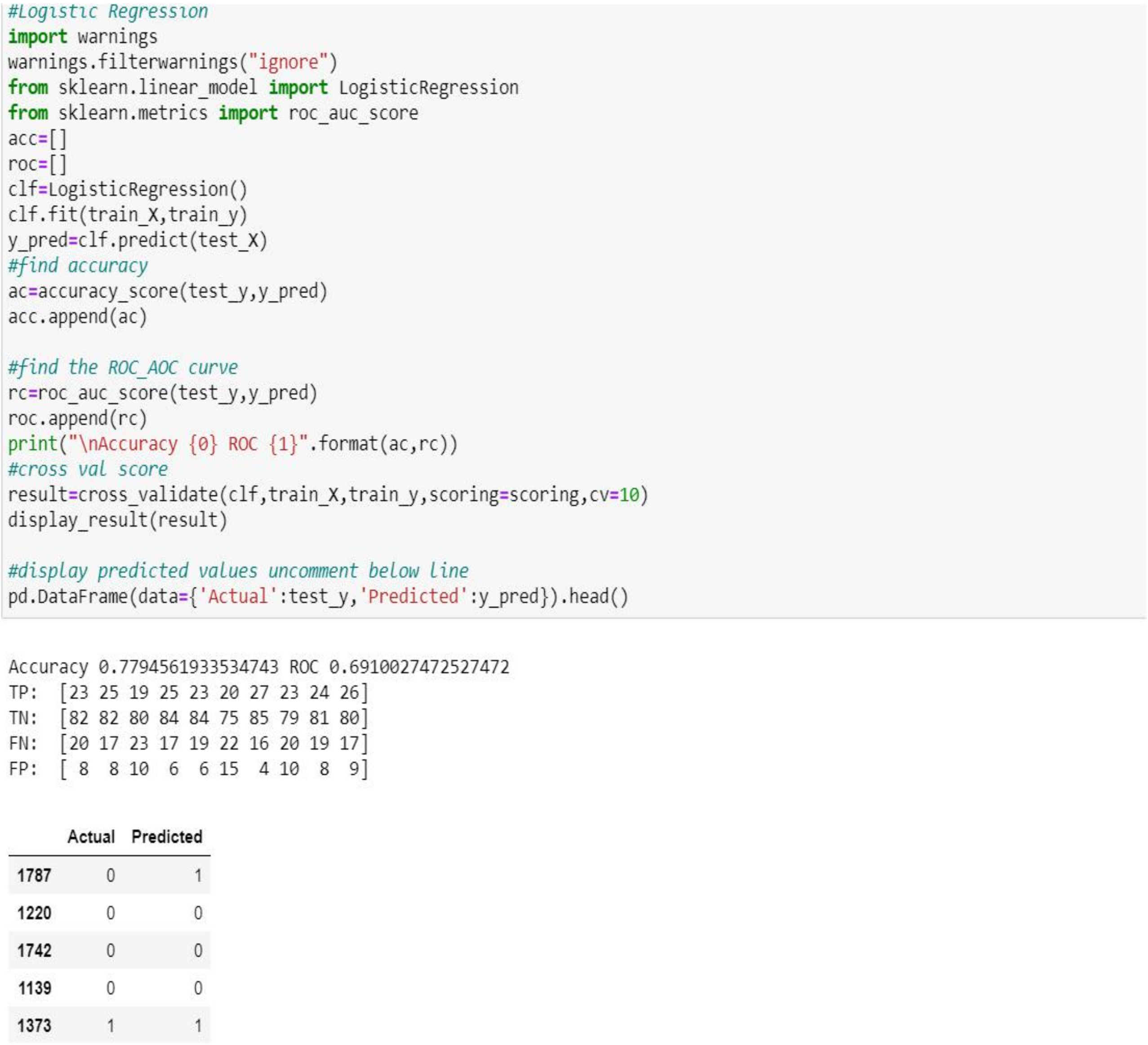
```
((1321, 8), (331, 8), (1321,), (331,))
```

Creating Confusion Matrix:



```
from sklearn.metrics import confusion_matrix,accuracy_score,make_scorer
from sklearn.model_selection import cross_validate

def tn(y_true, y_pred): return confusion_matrix(y_true, y_pred)[0, 0]
def fp(y_true, y_pred): return confusion_matrix(y_true, y_pred)[0, 1]
def fn(y_true, y_pred): return confusion_matrix(y_true, y_pred)[1, 0]
def tp(y_true, y_pred): return confusion_matrix(y_true, y_pred)[1, 1]

#cross validation purpose
scoring = {'accuracy': make_scorer(accuracy_score),'prec': 'precision'}
scoring = {'tp': make_scorer(tp), 'tn': make_scorer(tn),
           'fp': make_scorer(fp), 'fn': make_scorer(fn)}

def display_result(result):
    print("TP: ",result['test_tp'])
    print("TN: ",result['test_tn'])
    print("FN: ",result['test_fn'])
    print("FP: ",result['test_fp'])
```

Logistic Regression:

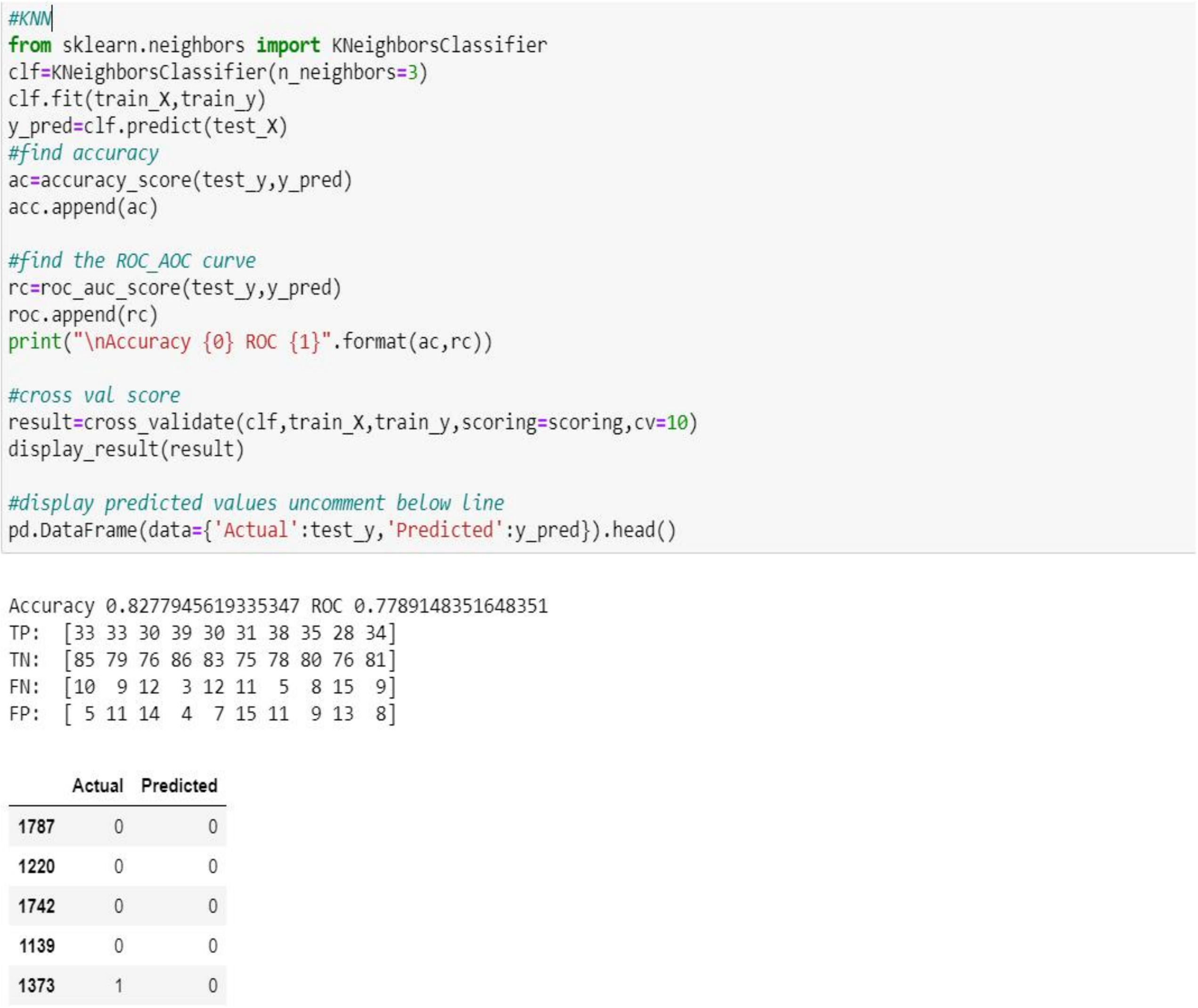


```
#Logistic Regression
import warnings
warnings.filterwarnings("ignore")
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
acc=[]
roc=[]
clf=LogisticRegression()
clf.fit(train_X,train_y)
y_pred=clf.predict(test_X)
#find accuracy
ac=accuracy_score(test_y,y_pred)
acc.append(ac)

#find the ROC_AOC curve
rc=roc_auc_score(test_y,y_pred)
roc.append(rc)
print("\nAccuracy {0} ROC {1}".format(ac,rc))
#cross val score
result=cross_validate(clf,train_X,train_y,scoring=scoring,cv=10)
display_result(result)

#display predicted values uncomment below line
pd.DataFrame(data={'Actual':test_y,'Predicted':y_pred}).head()
```

```
Accuracy 0.7794561933534743 ROC 0.6910027472527472
TP:  [23 25 19 25 23 20 27 23 24 26]
TN:  [82 82 80 84 84 75 85 79 81 80]
FN:  [20 17 23 17 19 22 16 20 19 17]
FP:  [ 8  8 10  6  6 15  4 10  8  9]
```

	Actual	Predicted
1787	0	1
1220	0	0
1742	0	0
1139	0	0
1373	1	1

K-Nearest Neighbors (KNN):

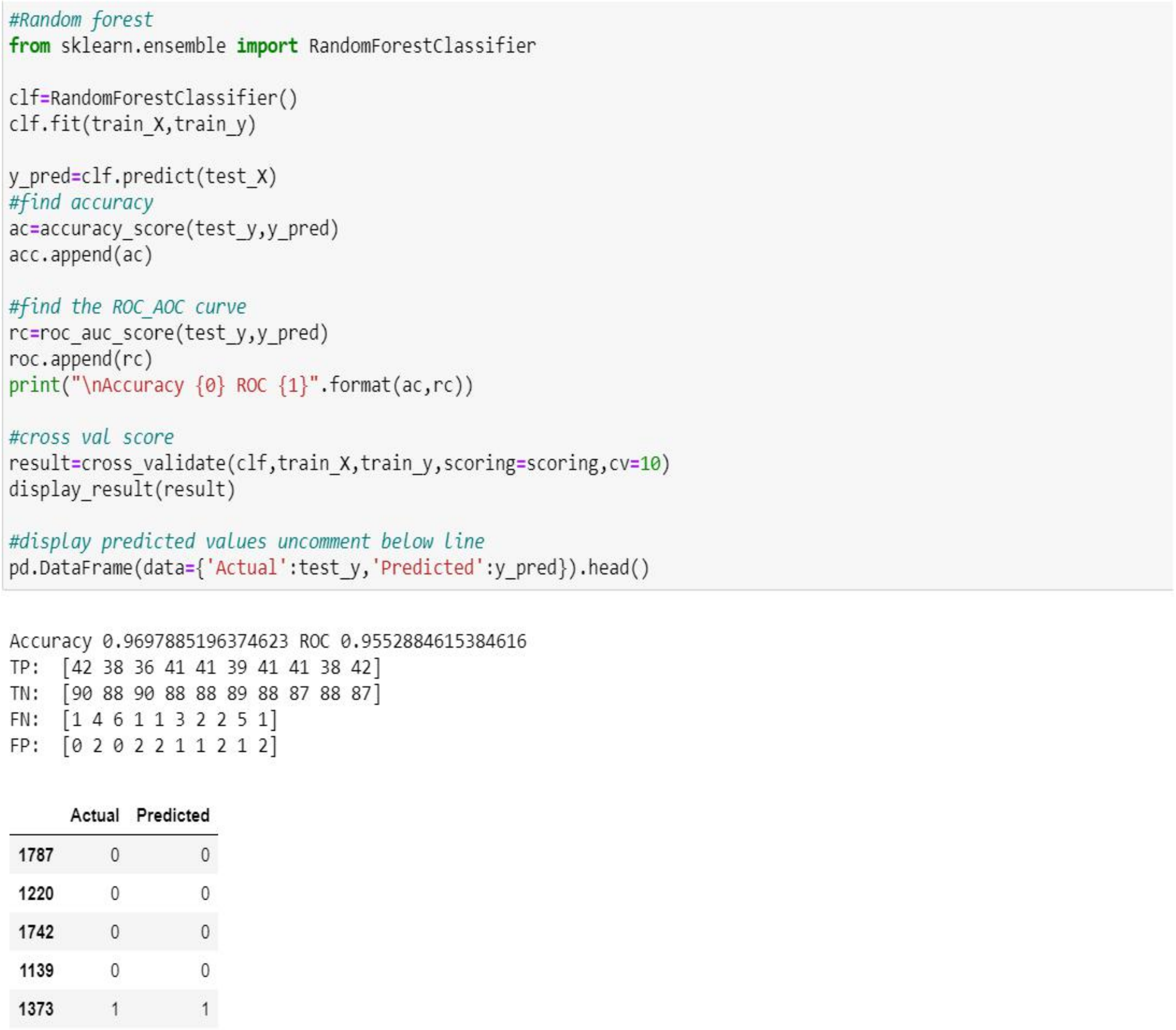


```
#KNN
from sklearn.neighbors import KNeighborsClassifier
clf=KNeighborsClassifier(n_neighbors=3)
clf.fit(train_X,train_y)
y_pred=clf.predict(test_X)
#find accuracy
ac=accuracy_score(test_y,y_pred)
acc.append(ac)

#find the ROC_AOC curve
rc=roc_auc_score(test_y,y_pred)
roc.append(rc)
print("\nAccuracy {0} ROC {1}".format(ac,rc))

#cross val score
result=cross_validate(clf,train_X,train_y,scoring=scoring,cv=10)
display_result(result)

#display predicted values uncomment below line
pd.DataFrame(data={'Actual':test_y,'Predicted':y_pred}).head()
```

```
Accuracy 0.8277945619335347 ROC 0.7789148351648351
TP:  [33 33 30 39 30 31 38 35 28 34]
TN:  [85 79 76 86 83 75 78 80 76 81]
FN:  [10  9 12  3 12 11  5  8 15  9]
FP:  [ 5 11 14  4  7 15 11  9 13  8]
```

	Actual	Predicted
1787	0	0
1220	0	0
1742	0	0
1139	0	0
1373	1	0

Random Forest Algorithm:

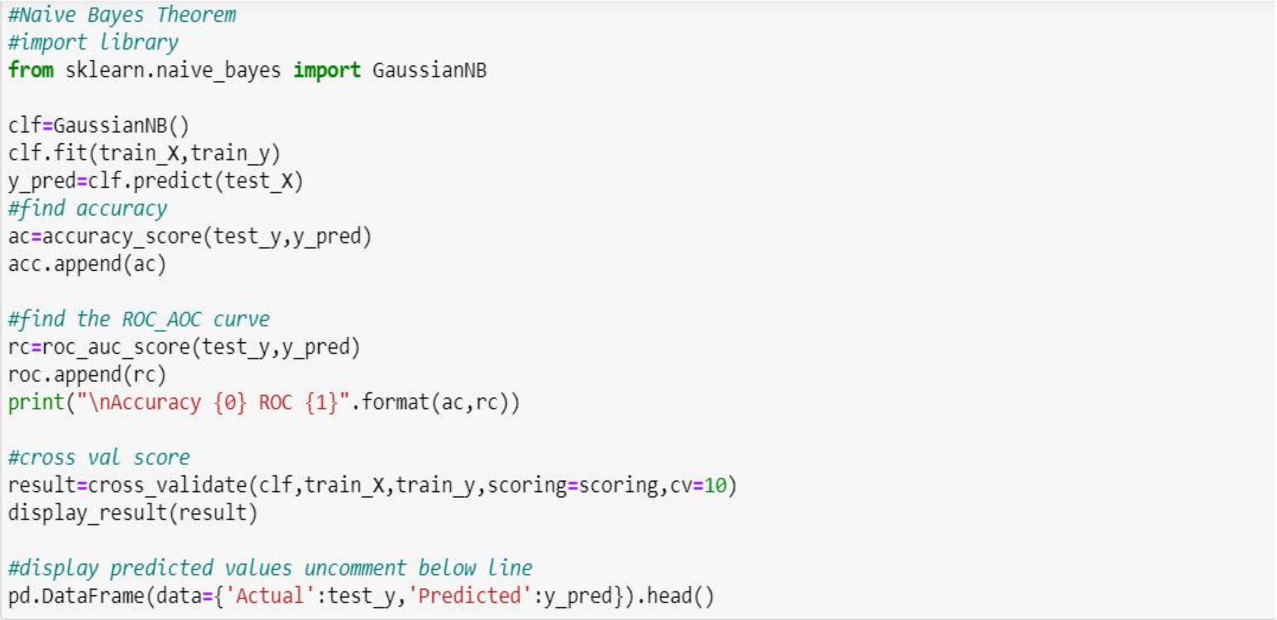


```
#Random forest
from sklearn.ensemble import RandomForestClassifier

clf=RandomForestClassifier()
clf.fit(train_X,train_y)

y_pred=clf.predict(test_X)
#find accuracy
ac=accuracy_score(test_y,y_pred)
acc.append(ac)

#find the ROC_AOC curve
rc=roc_auc_score(test_y,y_pred)
roc.append(rc)
print("\nAccuracy {0} ROC {1}".format(ac,rc))

#cross val score
result=cross_validate(clf,train_X,train_y,scoring=scoring,cv=10)
display_result(result)

#display predicted values uncomment below line
pd.DataFrame(data={'Actual':test_y,'Predicted':y_pred}).head()
```

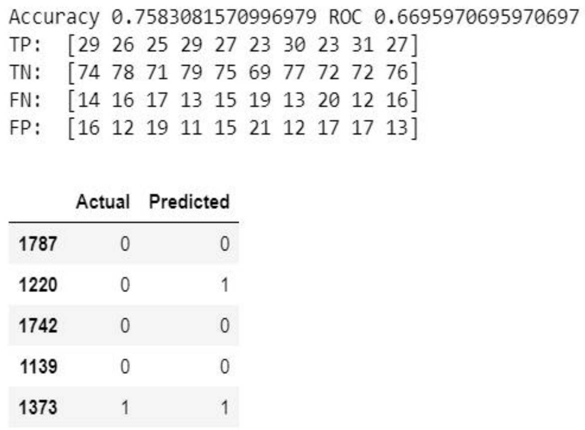
```
Accuracy 0.9697885196374623 ROC 0.9552884615384616
TP:  [42 38 36 41 41 39 41 41 38 42]
TN:  [90 88 90 88 88 89 88 87 88 87]
FN:  [1 4 6 1 1 3 2 2 5 1]
FP:  [0 2 0 2 2 1 1 2 1 2]
```

	Actual	Predicted
1787	0	0
1220	0	0
1742	0	0
1139	0	0
1373	1	1

Naive Baye’s Theorem:



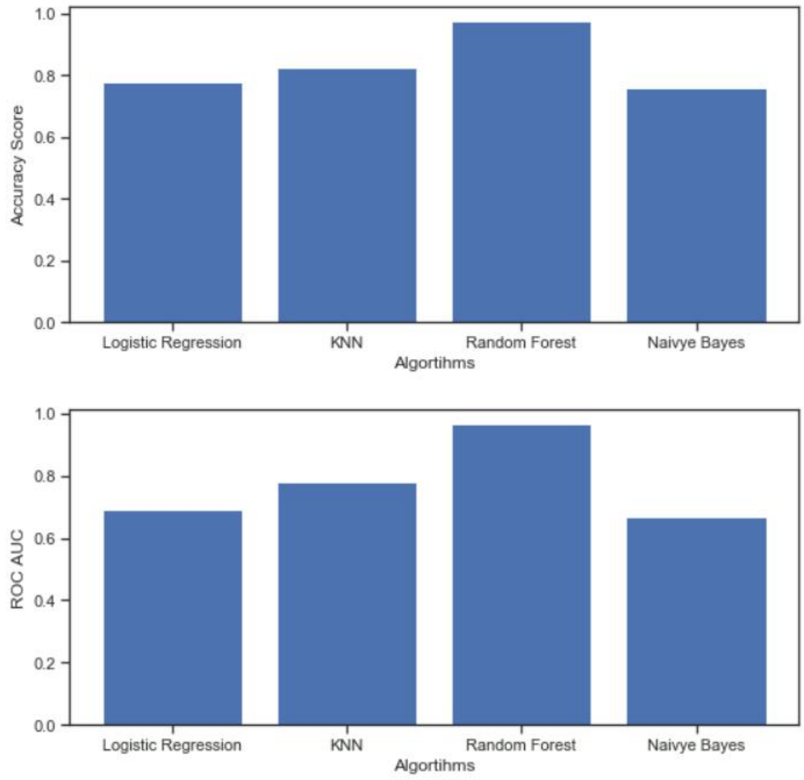
```
#Naive Bayes Theorem
#import library
from sklearn.naive_bayes import GaussianNB

clf=GaussianNB()
clf.fit(train_X,train_y)
y_pred=clf.predict(test_X)
#find accuracy
ac=accuracy_score(test_y,y_pred)
acc.append(ac)

#find the ROC_AOC curve
rc=roc_auc_score(test_y,y_pred)
roc.append(rc)
print("\nAccuracy {0} ROC {1}".format(ac,rc))

#cross val score
result=cross_validate(clf,train_X,train_y,scoring=scoring,cv=10)
display_result(result)

#display predicted values uncomment below line
pd.DataFrame(data={'Actual':test_y,'Predicted':y_pred}).head()
```



```
Accuracy 0.7583081570996979 ROC 0.6695970695970697
TP:  [29 26 25 29 27 23 30 23 31 27]
TN:  [74 78 71 79 75 69 77 72 72 76]
FN:  [14 16 17 13 15 19 13 20 12 16]
FP:  [16 12 19 11 15 21 12 17 17 13]
```

	Actual	Predicted
1787	0	0
1220	0	1
1742	0	0
1139	0	0
1373	1	1

Accuracy Measurement:



CONCLUSION

The project has been executed by using the KNN, Logistic Regression, Random Forest, Naive Baye’s Theorem. After testing the accuracy score with the confusion matrix and ROC AUC we observed the accuracy for the random forest algorithm is more compared to the others. We can implement this project by using other algorithms like SVM, Gradient boosting and some other Deep Learning algorithms which will result in producing the most accurate model to predict the diabetes.

REFERENCES

https://www.javatpoint.com/supervised-machine-learning

https://towardsdatascience.com/end-to-end-data-science-example-predicting-diabetes-with-logistic-regression-db9bc88b4d16

https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94

https://www.netapp.com/us/info/what-is-machine-learning-ml.aspx

APPENDIX

CODE:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

df=pd.read_csv('diabetes.csv')
df.head()

#lets describe the data
df.describe()
#infromation of dataset
df.info()
#any null values
df.isnull().values.any()

#data visualization
#histogram
df.hist(bins=10,figsize=(20,20))
plt.show()

#correlation
plt.figure(figsize=(8,6))
sns.heatmap(df.corr())

#lets count total outcome in each target 0 1
#0 means no diabeted
```

```
#1 means patient with diabtes
sns.countplot(y=df['Outcome'],palette='Set1')

sns.set(style="ticks")
sns.pairplot(df, hue="Outcome")

#outlier remove
Q1=df.quantile(0.25)
Q3=df.quantile(0.75)
IQR=Q3-Q1
print("---Q1--- \n",Q1)
print("\n---Q3--- \n",Q3)
print("\n---IQR---\n",IQR)
#print((df < (Q1 - 1.5 * IQR))|(df > (Q3 + 1.5 * IQR)))
df_out = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
df.shape,df_out.shape

#Scatter matrix after removing outlier
sns.set(style="ticks")
sns.pairplot(df_out, hue="Outcome")
plt.show()

#lets extract features and targets
X=df_out.drop(columns=['Outcome'])
y=df_out['Outcome']

#Splitting train test data 80 20 ratio
from sklearn.model_selection import train_test_split
train_X,test_X,train_y,test_y=train_test_split(X,y,test_size=0.2)
```

```
train_X.shape,test_X.shape,train_y.shape,test_y.shape

#creating confusion matrix
from sklearn.metrics import confusion_matrix,accuracy_score,make_scorer
from sklearn.model_selection import cross_validate

def tn(y_true, y_pred): return confusion_matrix(y_true, y_pred)[0, 0]
def fp(y_true, y_pred): return confusion_matrix(y_true, y_pred)[0, 1]
def fn(y_true, y_pred): return confusion_matrix(y_true, y_pred)[1, 0]
def tp(y_true, y_pred): return confusion_matrix(y_true, y_pred)[1, 1]

#cross validation purpose
scoring = {'accuracy': make_scorer(accuracy_score),'prec': 'precision'}
scoring = {'tp': make_scorer(tp), 'tn': make_scorer(tn),
           'fp': make_scorer(fp), 'fn': make_scorer(fn)}

def display_result(result):
   print("TP: ",result['test_tp'])
   print("TN: ",result['test_tn'])
   print("FN: ",result['test_fn'])
   print("FP: ",result['test_fp'])

#Lets build the model
#Logistic Regression
import warnings
warnings.filterwarnings("ignore")
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
acc=[]
```

```
roc=[]
clf=LogisticRegression()
clf.fit(train_X,train_y)
y_pred=clf.predict(test_X)
#find accuracy
ac=accuracy_score(test_y,y_pred)
acc.append(ac)

#find the ROC_AOC curve
rc=roc_auc_score(test_y,y_pred)
roc.append(rc)
print("\nAccuracy {0} ROC {1}".format(ac,rc))
#cross val score
result=cross_validate(clf,train_X,train_y,scoring=scoring,cv=10)
display_result(result)

#display predicted values uncomment below line
pd.DataFrame(data={'Actual':test_y,'Predicted':y_pred}).head()

#KNN
from sklearn.neighbors import KNeighborsClassifier
clf=KNeighborsClassifier(n_neighbors=3)
clf.fit(train_X,train_y)
y_pred=clf.predict(test_X)
#find accuracy
ac=accuracy_score(test_y,y_pred)
acc.append(ac)

#find the ROC_AOC curve
```

```
rc=roc_auc_score(test_y,y_pred)
roc.append(rc)
print("\nAccuracy {0} ROC {1}".format(ac,rc))

#cross val score
result=cross_validate(clf,train_X,train_y,scoring=scoring,cv=10)
display_result(result)

#display predicted values uncomment below line
pd.DataFrame(data={'Actual':test_y,'Predicted':y_pred}).head()

#Random forest
from sklearn.ensemble import RandomForestClassifier

clf=RandomForestClassifier()
clf.fit(train_X,train_y)

y_pred=clf.predict(test_X)
#find accuracy
ac=accuracy_score(test_y,y_pred)
acc.append(ac)

#find the ROC_AOC curve
rc=roc_auc_score(test_y,y_pred)
roc.append(rc)
print("\nAccuracy {0} ROC {1}".format(ac,rc))

#cross val score
result=cross_validate(clf,train_X,train_y,scoring=scoring,cv=10)
```

```
display_result(result)

#display predicted values uncomment below line
pd.DataFrame(data={'Actual':test_y,'Predicted':y_pred}).head()

#Naive Bayes Theorem
#import library
from sklearn.naive_bayes import GaussianNB

clf=GaussianNB()
clf.fit(train_X,train_y)
y_pred=clf.predict(test_X)
#find accuracy
ac=accuracy_score(test_y,y_pred)
acc.append(ac)

#find the ROC_AOC curve
rc=roc_auc_score(test_y,y_pred)
roc.append(rc)
print("\nAccuracy {0} ROC {1}".format(ac,rc))

#cross val score
result=cross_validate(clf,train_X,train_y,scoring=scoring,cv=10)
display_result(result)

#display predicted values uncomment below line
pd.DataFrame(data={'Actual':test_y,'Predicted':y_pred}).head()
```

```
#testing accuracy
#lets plot the bar graph

ax=plt.figure(figsize=(9,4))
plt.bar(['Logistic Regression','KNN','Random Forest','Naivye
Bayes'],acc,label='Accuracy')
plt.ylabel('Accuracy Score')
plt.xlabel('Algortihms')
plt.show()

ax=plt.figure(figsize=(9,4))
plt.bar(['Logistic Regression','KNN','Random Forest','Naivye Bayes'],roc,label='ROC
AUC')
plt.ylabel('ROC AUC')
plt.xlabel('Algortihms')
plt.show()
```