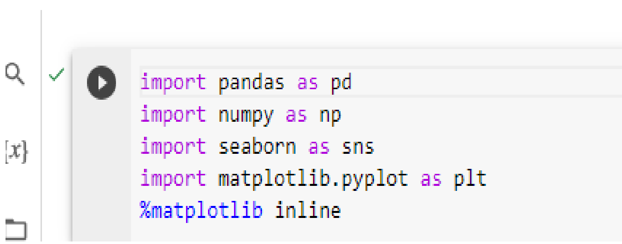
Project Name: AI Based Diabetes Prediction System

Project Code:203476

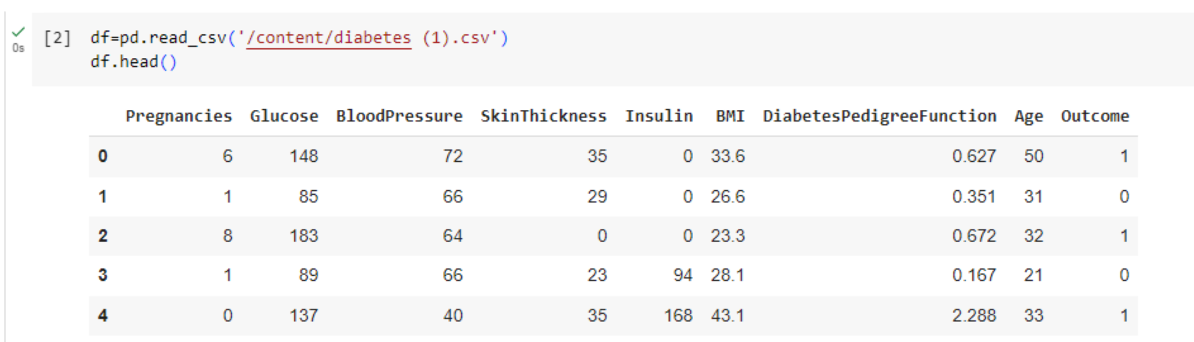
Coding:

```python
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline
```



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

```python
df=pd.read_csv('/content/diabetes (1).csv')

df.head()
```

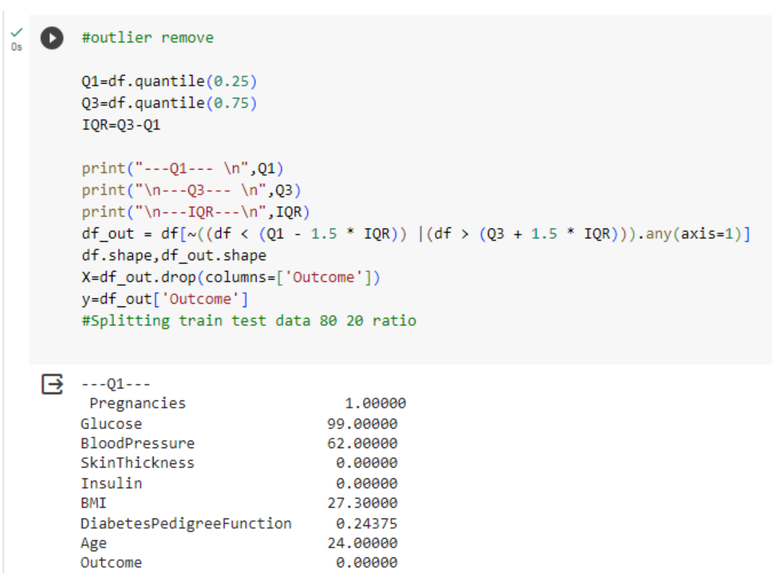


```python
[2] df=pd.read_csv('/content/diabetes (1).csv')
    df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```python
#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)

df_out = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]

df.shape,df_out.shape

X=df_out.drop(columns=['Outcome'])

y=df_out['Outcome']

#Splitting train test data 80 20 ratio
```



```
#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)
df_out = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
df.shape,df_out.shape
X=df_out.drop(columns=['Outcome'])
y=df_out['Outcome']
#Splitting train test data 80 20 ratio
```

```
---Q1---
 Pregnancies                  1.00000
Glucose                      99.00000
BloodPressure                62.00000
SkinThickness                 0.00000
Insulin                       0.00000
BMI                          27.30000
DiabetesPedigreeFunction      0.24375
Age                          24.00000
Outcome                       0.00000
Name: 0.25, dtype: float64
```

```
Outcome                        0.00000
Name: 0.25, dtype: float64

---Q3---
 Pregnancies                     6.00000
Glucose                      140.25000
BloodPressure                 80.00000
SkinThickness                 32.00000
Insulin                      127.25000
BMI                           36.60000
DiabetesPedigreeFunction       0.62625
Age                           41.00000
Outcome                        1.00000
Name: 0.75, dtype: float64

---IQR---
 Pregnancies                     5.0000
Glucose                       41.2500
BloodPressure                 18.0000
SkinThickness                 32.0000
Insulin                      127.2500
BMI                            9.3000
DiabetesPedigreeFunction       0.3825
Age                           17.0000
Outcome                        1.0000
dtype: float64
```

```python
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
```

```
[5] 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

    ((511, 8), (128, 8), (511,), (128,))
```

```python
from sklearn.metrics import confusion_matrix,accuracy_score,make_scorer

from sklearn.model_selection import cross_validate
```

```python
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]
```

```python
#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
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)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Accuracy 0.8515625 ROC 0.7956821026282853
TP:  [ 3  8 10 12  7 11  8  9  9  9]
TN:  [30 30 28 30 25 33 31 31 34 34]
FN:  [14  9  7  5 10  6  8  7  7  7]
FP:  [5 4 6 4 9 1 4 4 1 1]
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
```

```python
#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)
```

```python
#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)
```

```
Accuracy 0.796875 ROC 0.7819072313454336
TP:  [10 11  8 10  7  8 10 11  7 11]
TN:  [32 26 26 32 28 28 31 31 31 27]
FN:  [7 5 8 6 9 8 6 5 9 5]
FP:  [3 9 9 3 7 7 4 4 4 8]
```