Project Development Phase
Model Performance Test

TEAM:

B.V.V. MAHESHA CHANDRA

M.MANOJ BHASKAR

K. POORNA AVINASH

V.GOWRI SHANKAR

Model Performance Testing:

Our strategy for this machine learning project is centered on making use of classification models. For this machine learning project, we used logistic regression, random forest classifiers, decision tree classifiers, and Furthermore, we utilized Grid Search CV for hyper parameter adjustment.

Decision Tree Classifier:

Decision Tree Classifier

```
In [57]: from sklearn.tree import DecisionTreeClassifier

In [58]: model1 = DecisionTreeClassifier()

In [59]: model1.fit(x_train,y_train)
```

```
In [58]: model1 = DecisionTreeClassifier()

In [59]: model1.fit(x_train,y_train)
Out[59]: ▾ DecisionTreeClassifier
         DecisionTreeClassifier()

In [60]: y_predict = model1.predict(x_test)

In [61]: y_predict
Out[61]: array([0., 0., 0., ..., 0., 0., 0.])

In [62]: y_predict_train = model1.predict(x_train)

In [63]: from sklearn.metrics import accuracy_score,classification_report,confusion_matrix

In [64]: print('Testing Accuracy = ', accuracy_score(y_test,y_predict))
         print('Training Accuracy = ', accuracy_score(y_train,y_predict_train))

         Testing Accuracy =  0.7971197308945653
         Training Accuracy =  0.9945375501193855

In [65]: pd.crosstab(y_test,y_predict)
Out[65]:
```

col_0	0.0	1.0
Diabetes_binary		
0.0	57227	8123
1.0	7317	3437

```
In [66]: print(classification_report(y_test,y_predict))
                      precision    recall  f1-score   support

                 0.0       0.89      0.88      0.88     65350
                 1.0       0.30      0.32      0.31     10754

            accuracy                           0.80     76104
           macro avg       0.59      0.60      0.59     76104
        weighted avg       0.80      0.80      0.80     76104
```

Decision Tree Classifier After Hyper Parameter Turning

```
In [67]: from sklearn.model_selection import GridSearchCV
```

In [67]:
```
from sklearn.model_selection import GridSearchCV
```

In [68]:
```
parameters = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

In [69]:
```
clf = GridSearchCV(model1,param_grid = parameters,verbose =2)
```

In [70]:
```
clf.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 72 candidates, totalling 360 fits
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2; total time=   0.8s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2; total time=   0.9s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2; total time=   1.0s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2; total time=   1.0s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2; total time=   1.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5; total time=   0.9s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5; total time=   0.9s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5; total time=   0.7s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5; total time=   0.8s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5; total time=   0.7s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.8s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.7s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.7s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.7s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.7s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=2, min_samples_split=2; total time=   0.7s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=2, min_samples_split=2; total time=   0.8s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=2, min_samples_split=2; total time=   0.9s
```

In [71]:
```
clf.best_score_
```

Out[71]:
```
0.8612030994646263
```

In [72]:
```
clf.best_params_
```

Out[72]:
```
{'criterion': 'entropy',
 'max_depth': 10,
 'min_samples_leaf': 2,
 'min_samples_split': 2}
```

In [73]:
```
model2 = DecisionTreeClassifier(criterion='entropy',max_depth=10,min_samples_leaf=2,min_samples_split=5)
```

In [74]:
```
model2.fit(x_train,y_train)
```

```
                       min_samples_split=5)
```

In [75]:
```
y_ = model2.predict(x_test)
```

In [76]:
```
accuracy_score(y_test,y_)
```

Out[76]:
```
0.8611636707663197
```

In [77]:
```
pd.crosstab(y_test,y_)
```

Out[77]:

col_0	0.0	1.0
Diabetes_binary		
0.0	64190	1160
1.0	9406	1348

In [78]:
```
print(classification_report(y_test,y_))
```

```
              precision    recall  f1-score   support

         0.0       0.87      0.98      0.92     65350
         1.0       0.54      0.13      0.20     10754

    accuracy                           0.86     76104
   macro avg       0.70      0.55      0.56     76104
weighted avg       0.82      0.86      0.82     76104
```

Random Forest Classifier:

RandomForest Classifier

```
In [79]: from sklearn.ensemble import RandomForestClassifier

In [80]: model3 =RandomForestClassifier()

In [81]: model3.fit(x_train,y_train)

Out[81]: RandomForestClassifier()

In [82]: r_y_predict = model3.predict(x_test)
         r_y_predict_train = model3.predict(x_train)

In [83]: print('Testing Accuracy = ', accuracy_score(y_test,r_y_predict))
         print('Training Accuracy = ', accuracy_score(y_train,r_y_predict_train))
```

```
In [84]: pd.crosstab(y_test,r_y_predict)

Out[84]:
col_0              0.0   1.0
Diabetes_binary
0.0              63609  1741
1.0               9103  1651
```

```
In [85]: print(classification_report(y_test,r_y_predict))

              precision    recall  f1-score   support

         0.0       0.87      0.97      0.92     65350
         1.0       0.49      0.15      0.23     10754

    accuracy                           0.86     76104
   macro avg       0.68      0.56      0.58     76104
weighted avg       0.82      0.86      0.82     76104
```

Random Forest Classifier After Hyper Parameter Turning:

```
In [86]: from sklearn.model_selection import GridSearchCV

In [87]: parameters1 = {
             'n_estimators': [100, 200],  # Number of trees in the forest
             'criterion': ['gini', 'entropy'],  # Criterion for splitting
             'max_depth': [None, 10, 20],  # Maximum depth of trees
             'min_samples_split': [2, 5],  # Minimum samples required to split a node
             'min_samples_leaf': [1, 2],  # Minimum samples required to be in a leaf node
         }

In [88]: clf1 = GridSearchCV(model3,param_grid = parameters1,verbose =2)

In [89]: clf1.fit(x_train,y_train)

         Fitting 5 folds for each of 48 candidates, totalling 240 fits
         [CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=  24.9s
         [CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=  25.0s
         [CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=  25.9s
         [CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=  24.7s
         [CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=  25.8s
         [CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=  49.7s
         [CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=  49.2s
         [CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=  52.1s
         [CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=  49.4s
         [CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=  51.5s
         [CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=  21.5s
         [CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=  20.2s
         [CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=  19.9s
```

```
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=200; total time=  39.9s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=200; total time=  40.8s
```

In [90]:
```
clf1.best_score_
```

Out[90]:
```
0.864362310676569
```

In [91]:
```
clf1.best_params_
```

Out[91]:
```
{'criterion': 'entropy',
 'max_depth': 20,
 'min_samples_leaf': 2,
 'min_samples_split': 5,
 'n_estimators': 200}
```

In [92]:
```
model4 =RandomForestClassifier(criterion= 'gini',
 max_depth= 10,
 min_samples_leaf= 1,
 min_samples_split= 2,
 n_estimators= 200
)
```

In [93]:
```
model4.fit(x_train,y_train)
```

Out[93]:
```
RandomForestClassifier(max_depth=10, n_estimators=200)
```

In [94]:
```
r_y_predict1 = model4.predict(x_test)
r_y_predict_train1= model4.predict(x_train)
```

In [95]:
```
print('Testing Accuracy = ', accuracy_score(y_test,r_y_predict1))
print('Training Accuracy = ', accuracy_score(y_train,r_y_predict_train1))
```

```
Testing Accuracy =  0.8614921686113739
Training Accuracy =  0.8661192954002793
```

In [96]:
```
pd.crosstab(y_test,r_y_predict1)
```

Out[96]:

col_0	0.0	1.0
Diabetes_binary		
0.0	65074	276
1.0	10265	489

In [95]:
```
print(classification_report(y_test,r_y_predict1))
```

```
              precision    recall  f1-score   support

         0.0       0.86      1.00      0.93     65350
         1.0       0.64      0.05      0.09     10754

    accuracy                           0.86     76104
   macro avg       0.75      0.52      0.51     76104
weighted avg       0.83      0.86      0.81     76104
```

Logistic Regression:

```
###LogisticRegression

In [100]: from sklearn.linear_model import LogisticRegression
          model5 = LogisticRegression()

In [101]: model5.fit(x_train,y_train)

Out[101]: LogisticRegression()

In [102]: pred = model5.predict(x_test)
          pred

Out[102]: array([0., 0., 0., ..., 0., 0., 0.])

In [103]: from sklearn.metrics import accuracy_score, confusion_matrix,classification_report,roc_auc_score,roc_curve
```

```
In [82]: pd.crosstab(y_test,r_y_predict)
```

Out[82]:

col_0	0.0	1.0
Diabetes_binary		
0.0	63529	1821
1.0	9108	1646

```
In [83]: print(classification_report(y_test,r_y_predict))
```

```
              precision    recall  f1-score   support

         0.0       0.87      0.97      0.92     65350
         1.0       0.47      0.15      0.23     10754

    accuracy                           0.86     76104
   macro avg       0.67      0.56      0.58     76104
weighted avg       0.82      0.86      0.82     76104
```

Naïve Bayes

naive bayes

In [105]:
```
from sklearn.naive_bayes import GaussianNB
```

In [106]:
```
model3 = GaussianNB()
```

In [110]:
```
x_train_scaled= pd.DataFrame(scale.fit_transform(x_train),columns=x_train.columns)
x_test_scaled= pd.DataFrame(scale.fit_transform(x_test),columns=x_test.columns)
```

In [111]:
```
model3.fit(x_train_scaled,y_train)
```

Out[111]:
```
GaussianNB()
```

In [112]:
```
y_pred3 = model3.predict(x_test_scaled)
```

In [115]:
```
y_pred3_train = model3.predict(x_train_scaled)
```

In [116]:
```
print("Test accuracy", accuracy_score(y_test,y_pred3))
print("Train accuracy", accuracy_score(y_train,y_pred3_train))
```

```
Test accuracy 0.7721407547566488
Train accuracy 0.7710332477361805
```

In [117]:
```
pd.crosstab(y_test,y_pred3)
```

Out[117]:

col_0	0.0	1.0
Diabetes_binary		
0.0	52538	12812
1.0	4529	6225

In [118]:
```
print(classification_report(y_test,y_pred3))
```

```
              precision    recall  f1-score   support

         0.0       0.92      0.80      0.86     65350
         1.0       0.33      0.58      0.42     10754

    accuracy                           0.77     76104
   macro avg       0.62      0.69      0.64     76104
weighted avg       0.84      0.77      0.80     76104
```

Conclusion:

In conclusion, we set out to explore and construct a variety of models for our machine learning project. However, we meticulously evaluated each model's performance in an effort to achieve the highest accuracy. Consequently, we made the decision to use the Random Forest Classifier and a single model that continuously showed exceptional accuracy. This tactical decision reflects our dedication to providing the most accurate and trustworthy outcomes and is a prime example of data-driven decision-making.