Project Development Phase
Model Performance Test

Konjerla Likhith
Pentapati Meher Baba
Mudunuri Harsha Vardhan Varma
Challa Sai Phanindra

Date	8 November 2023
Team ID	PNT2022TMIDxxxxxx
Project Name	Diabetes Prediction Using Machine Learning
Maximum Marks	10 Marks

Model Performance Testing:

Metrics:

In this machine learning project, our approach is focused on utilizing classification models.

In this machine learning project, we employed the Decision Tree Classifier, Random Forest Classifier, Logistic Regression, and Support Vector Classifier (SVC). Additionally, we harnessed GridSearchCV for hyperparameter tuning.

Decision Tree Classifier:

▾ **Decision Tree Classifier**

```
[ ] from sklearn.tree import DecisionTreeClassifier
```

```
[ ] model1 = DecisionTreeClassifier()
```

```
[ ] model1.fit(x_train,y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
[ ] y_predict = model1.predict(x_test)
```

```
[ ] y_predict
```

```
array([0., 0., 1., ..., 0., 0., 0.])
```

```
[ ] y_predict_train = model1.predict(x_train)
```

```
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
```

+ Code + Text

```
print('Testing Accuracy = ', accuracy_score(y_test,y_predict))
print('Training Accuracy = ', accuracy_score(y_train,y_predict_train))
```

```
Testing Accuracy =  0.7774294670846394
Training Accuracy =  0.9989617052464423
```

```
pd.crosstab(y_test,y_predict)
```

col_0	0.0	1.0
Diabetes_binary		
0.0	5164	881
1.0	681	292

```
print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

         0.0       0.88      0.85      0.87      6045
         1.0       0.25      0.30      0.27       973

    accuracy                           0.78      7018
   macro avg       0.57      0.58      0.57      7018
weighted avg       0.80      0.78      0.79      7018
```

Decision Tree Classifier After Hyper Parameter Turning:

```
from sklearn.model_selection import GridSearchCV
```

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

```
clf = GridSearchCV(model1,param_grid = parameters,verbose =2)
```

```
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=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
```

```
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=4, min_samples_split=10; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=4, min_samples_split=10; total time=   0.1s
```

GridSearchCV
estimator: DecisionTreeClassifier
DecisionTreeClassifier

```
clf.best_score_
```

```
0.8404693933699235
```

```
clf.best_params_
```

```
{'criterion': 'entropy',
 'max_depth': 10,
 'min_samples_leaf': 4,
 'min_samples_split': 5}
```

```
model2 = DecisionTreeClassifier(criterion='entropy',max_depth=10,min_samples_leaf=2,min_samples_split=5)
```

```
model2.fit(x_train,y_train)
```

DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=10, min_samples_leaf=2, min_samples_split=5)

```
y_ = model2.predict(x_test)
```

```
accuracy_score(y_test,y_)
```

```
0.8435451695639783
```

```
pd.crosstab(y_test,y_)
```

col_0	0.0	1.0
Diabetes_binary		
0.0	5745	300
1.0	798	175

```
print(classification_report(y_test,y_))
```

```
              precision    recall  f1-score   support

         0.0       0.88      0.95      0.91      6045
         1.0       0.37      0.18      0.24       973

    accuracy                           0.84      7018
   macro avg       0.62      0.57      0.58      7018
weighted avg       0.81      0.84      0.82      7018
```

Random Forest Classifier:

RandomForest

```
[ ] from sklearn.ensemble import RandomForestClassifier
```

```
[ ] model3 =RandomForestClassifier()
```

```
[ ] model3.fit(x_train,y_train)
```

```
RandomForestClassifier
RandomForestClassifier()
```

```
[ ] r_y_predict = model3.predict(x_test)
    r_y_predict_train = model3.predict(x_train)
```

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

```
Testing Accuracy =  0.8605015673981191
Training Accuracy =  0.9989617052464423
```

```
pd.crosstab(y_test,r_y_predict)
```

col_0	0.0	1.0
Diabetes_binary		
0.0	5911	134
1.0	845	128

```
[ ] print(classification_report(y_test,r_y_predict))
```

```
              precision    recall  f1-score   support

         0.0       0.87      0.98      0.92      6045
         1.0       0.49      0.13      0.21       973

    accuracy                           0.86      7018
   macro avg       0.68      0.55      0.57      7018
weighted avg       0.82      0.86      0.82      7018
```

Random Forest Classifier After Hyper Parameter Turning:

```
[ ] from sklearn.model_selection import GridSearchCV
```

```
[ ] 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
    }
```

```
clf1 = GridSearchCV(model3,param_grid = parameters1,verbose =2)
```

```
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=   1.3s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   1.3s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   1.3s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   1.5s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   1.9s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   2.6s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   2.5s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   2.6s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   2.5s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   3.5s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=   1.2s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=   1.2s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=   1.2s
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   2.6s
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   2.6s
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   2.4s
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   2.3s
▸ GridSearchCV
▸ estimator: RandomForestClassifier
  ▸ RandomForestClassifier
```

```
[ ] clf1.best_score_
```

```
0.8573262018121028
```

```
clf1.best_params_
```

```
{'criterion': 'gini',
 'max_depth': None,
 'min_samples_leaf': 2,
 'min_samples_split': 2,
 'n_estimators': 100}
```

+ Code + Text

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

```
[ ] model4.fit(x_train,y_train)
```

```
RandomForestClassifier(max_depth=10, n_estimators=200)
```

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

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

```
Testing Accuracy =  0.8639213451125677
Training Accuracy =  0.8849935870029927
```

```
[ ] pd.crosstab(y_test,r_y_predict1)
```

col_0	0.0	1.0
Diabetes_binary		
0.0	6003	42
1.0	913	60

```
print(classification_report(y_test,r_y_predict1))
```

```
              precision    recall  f1-score   support

         0.0       0.87      0.99      0.93      6045
         1.0       0.59      0.06      0.11       973

    accuracy                           0.86      7018
   macro avg       0.73      0.53      0.52      7018
weighted avg       0.83      0.86      0.81      7018
```

Logistic Regression:

▾ LogisticRegression

```
[ ] from sklearn.linear_model import LogisticRegression
    model5 = LogisticRegression()
```

```
[ ] model5.fit(x_train,y_train)
```

▾ LogisticRegression
LogisticRegression()

```
pred = model5.predict(x_test)
pred
```

```
array([0., 0., 0., ..., 0., 0., 0.])
```

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

```
[ ] accuracy_score(y_test,pred)
```

```
0.8587916785408949
```

```
pd.crosstab(y_test,pred)
```

col_0	0.0	1.0
Diabetes_binary		
0.0	5912	133
1.0	858	115

```
[ ] print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

         0.0       0.87      0.98      0.92      6045
         1.0       0.46      0.12      0.19       973

    accuracy                           0.86      7018
   macro avg       0.67      0.55      0.56      7018
weighted avg       0.82      0.86      0.82      7018
```

SVC:

▾ SVC

```
[ ] from sklearn.svm import SVC
```

```
[ ] model6 = SVC()
```

```
model6.fit(x_train,y_train)
```

▾ SVC
SVC()

```
[ ] y_pred1 = model6.predict(x_train)
```

```
[ ] y_pred = model6.predict(x_test)
```

```
[ ] from sklearn.metrics import accuracy_score,classification_report
```

```
[ ] print("Test accuracy", accuracy_score(y_test,y_pred))
    print("Train accuracy", accuracy_score(y_train,y_pred1))
```

```
Test accuracy 0.8610715303505272
Train accuracy 0.857570390276675
```

```
[ ] pd.crosstab(y_test,y_pred)
```

col_0	0.0	1.0
Diabetes_binary		
0.0	6029	16
1.0	959	14

```
[ ] print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.86      1.00      0.93      6045
         1.0       0.47      0.01      0.03       973

    accuracy                           0.86      7018
   macro avg       0.66      0.51      0.48      7018
weighted avg       0.81      0.86      0.80      7018
```

SVC After Hyper Parameter Turning:

```
[ ] from sklearn.model_selection import GridSearchCV
```

```
[ ] parameters2 ={'kernel':['linear','rbf'],
                  'C':[0.1,0.2,0.5,1.0],
                  'gamma':['scale','auto']}
```

```
[ ] clf2 = GridSearchCV(model6,param_grid = parameters2,verbose =2)
```

```
clf2.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
[CV] END ..................C=0.1, gamma=scale, kernel=linear; total time=   2.7s
[CV] END ..................C=0.1, gamma=scale, kernel=linear; total time=   2.9s
[CV] END ..................C=0.1, gamma=scale, kernel=linear; total time=   2.1s
[CV] END ..................C=0.1, gamma=scale, kernel=linear; total time=   2.0s
[CV] END ..................C=0.1, gamma=scale, kernel=linear; total time=   2.3s
[CV] END ......................C=0.1, gamma=scale, kernel=rbf; total time=   6.8s
[CV] END ......................C=0.1, gamma=scale, kernel=rbf; total time=   5.7s
[CV] END ......................C=0.1, gamma=scale, kernel=rbf; total time=   6.9s
[CV] END ......................C=0.1, gamma=scale, kernel=rbf; total time=   5.7s
[CV] END ......................C=0.1, gamma=scale, kernel=rbf; total time=   7.9s
[CV] END ...................C=0.1, gamma=auto, kernel=linear; total time=   2.8s
[CV] END ...................C=0.1, gamma=auto, kernel=linear; total time=   2.0s
[CV] END ...................C=0.1, gamma=auto, kernel=linear; total time=   2.1s
[CV] END ...................C=0.1, gamma=auto, kernel=linear; total time=   2.0s
[CV] END ...................C=0.1, gamma=auto, kernel=linear; total time=   2.6s
[CV] END .......................C=0.1, gamma=auto, kernel=rbf; total time=   5.5s
[CV] END .......................C=0.1, gamma=auto, kernel=rbf; total time=   4.5s
[CV] END .......................C=0.1, gamma=auto, kernel=rbf; total time=   5.5s
[CV] END .......................C=0.1, gamma=auto, kernel=rbf; total time=   4.3s
[CV] END .......................C=0.1, gamma=auto, kernel=rbf; total time=   4.3s
[CV] END ..................C=0.2, gamma=scale, kernel=linear; total time=   3.1s
[CV] END ..................C=0.2, gamma=scale, kernel=linear; total time=   2.3s
[CV] END ..................C=0.2, gamma=scale, kernel=linear; total time=   2.0s
[CV] END ..................C=0.2, gamma=scale, kernel=linear; total time=   2.2s
[CV] END ..................C=0.2, gamma=scale, kernel=linear; total time=   2.3s
[CV] END ......................C=0.2, gamma=scale, kernel=rbf; total time=   7.1s
```

```
[CV] END .....................C=1.0, gamma=auto, kernel=rbf; total time=   7.7s
[CV] END .....................C=1.0, gamma=auto, kernel=rbf; total time=   8.2s
[CV] END .....................C=1.0, gamma=auto, kernel=rbf; total time=   8.1s
[CV] END .....................C=1.0, gamma=auto, kernel=rbf; total time=   7.4s
[CV] END .....................C=1.0, gamma=auto, kernel=rbf; total time=   8.2s
```

▸ GridSearchCV
▸ estimator: SVC
▸ SVC

```
[ ] clf2.best_score_
```

```
0.8549441307176131
```

```
[ ] clf2.best_params_
```

```
{'C': 1.0, 'gamma': 'scale', 'kernel': 'rbf'}
```

```
[ ] model7 = SVC(C= 1.0, gamma= 'scale', kernel= 'rbf')
```

```
[ ] model7.fit(x_train,y_train)
```

▾ SVC
SVC()

```
y_pred2 = model7.predict(x_train)
```

```
[ ] y_pred3 = model7.predict(x_test)
```

```
[ ] from sklearn.metrics import accuracy_score,classification_report
```

```
[ ] print("Test accuracy", accuracy_score(y_test,y_pred3))
    print("Train accuracy", accuracy_score(y_train,y_pred2))
```

```
Test accuracy 0.8610715303505272
Train accuracy 0.857570390276675
```

```
[ ] pd.crosstab(y_test,y_pred3)
```

col_0	0.0	1.0
Diabetes_binary		
0.0	6029	16
1.0	959	14

```
print(classification_report(y_test,y_pred3))
```

```
              precision    recall  f1-score   support

         0.0       0.86      1.00      0.93      6045
         1.0       0.47      0.01      0.03       973

    accuracy                           0.86      7018
   macro avg       0.66      0.51      0.48      7018
weighted avg       0.81      0.86      0.80      7018
```

Conclusion:

In our machine learning project, we embarked on a journey of exploring and building various models. However, in the pursuit of achieving the highest accuracy, we carefully assessed each model's performance. As a result, we decided to select **Random Forest Classifier** and utilize a single model that consistently demonstrated remarkable accuracy. This strategic choice underscores our commitment to delivering the most precise and reliable results, and it exemplifies the essence of data-driven decision-making.