

Project Development Phase Model Performance Test

Date	10 November 2022
Team ID	PNT2022TMID30811
Project Name	Project – Web Phishing Detection
Maximum Marks	10 Marks

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Metrics	Regression Model: Logistic Regression MAE – 0.26142017186793304 MSE - 0.5228403437358661 RMSE - 0.7230769971004928 R2 score - -2.888673182487615 Classification Model: Decision Tree Classifier Confusion Matrix - array([[61, 249], [26, 1875]]) Accuracy Score- 0.8756218905472637 Classification Report – refer screenshot	Attached Below
2.	Tune the Model	Hyperparameter Tuning - Validation Method -	Attached Below

1. METRICS:

REGRESSION MODEL: LOGISTIC REGRESSION

```
Working with Logistic Regression model

[35] #splitting data into train and test
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)

[30] #fitting the data
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(x_train,y_train)

LogisticRegression()

[36] pred=lr.predict(x_test)

[37] pred
array([1, 1, 1, ..., 1, 1, 1])
```

EVALUATION METRICS:

Here are some evaluation metrics used for regression they are,

- R2 Score
- Mean Square Error(MSE)
- RMSE(Root Mean Square Error)
- Mean Absolute Error(MAE)

```
evaluation metrics

[50] from sklearn.metrics import mean_squared_error,r2_score,mean_absolute_error
mse=mean_squared_error(pred,y_test)

[51] mean_absolute_error(pred,y_test)
0.26142017186793304

[39] mse
0.5228403437358661

[40] rmse=np.sqrt(mse)

[41] rmse
0.7230769971004928

[42] r2=r2_score(pred,y_test)

[43] r2
-2.888673182487615
```

CLASSIFICATION MODEL: DECISION TREE CLASSIFIER

```
building the Decision Tree Classifier model

[44] # Decision Tree model
from sklearn.tree import DecisionTreeClassifier

# instantiate the model
tree = DecisionTreeClassifier(max_depth = 5)
# fit the model
tree.fit(x_train, y_train)

DecisionTreeClassifier(max_depth=5)

[45] #prediction on test data
pred2=tree.predict(x_test)
pred2

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

EVALUATION METRICS:

Some of the evaluation metrics is as follows

- Confusion matrix
- Accuracy score
- Classification report

```
evaluation metrics

[63] from sklearn import metrics

[47] metrics.confusion_matrix(y_test,pred2)

array([[ 61, 249],
       [ 26, 1875]])

[53] print('DT model Accuracy Score:',metrics.accuracy_score(y_test,pred2))

DT model Accuracy Score: 0.8756218905472637

[54] acc=metrics.accuracy_score(y_test,pred2)
acc

0.8756218905472637

[55] #error
1-acc

0.12437810945273631
```

```
[65] from sklearn.metrics import classification_report

report = classification_report(y_test,pred2)
print("Classification report:")
print(report)

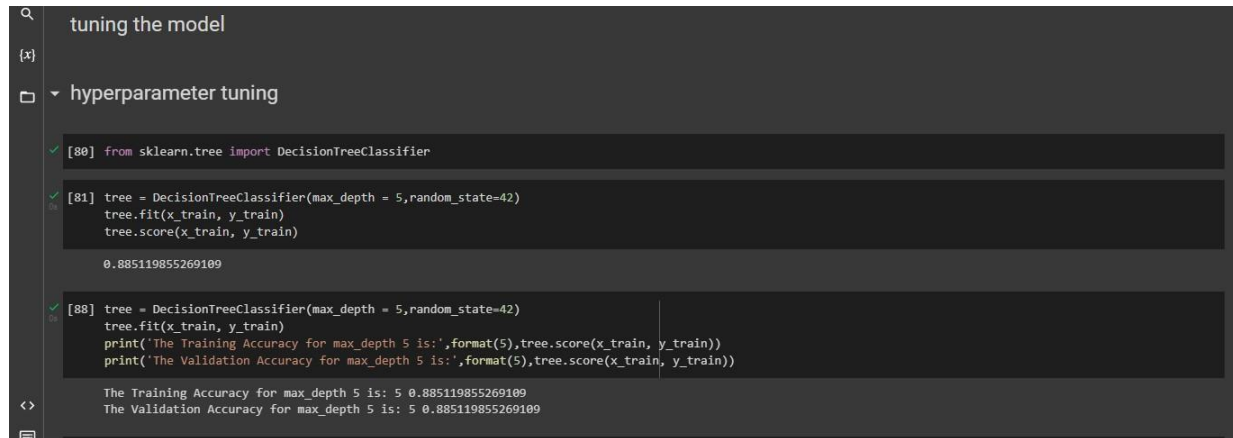
Classification report:
precision    recall  f1-score   support

   -1       0.70     0.20     0.31       310
    1       0.88     0.99     0.93      1901

 accuracy          0.88       2211
  macro avg       0.79     0.59     0.62       2211
 weighted avg     0.86     0.88     0.84       2211
```

2. TUNE THE MODEL: DECISION TREE CLASSIFIER

HYPERPARAMETER TUNING:



The image shows a Jupyter Notebook interface with a search bar at the top containing the text "tuning the model". Below the search bar, a sidebar on the left shows a file explorer with a folder named "hyperparameter tuning" which is currently selected. The main area of the notebook displays three code cells. The first cell imports the DecisionTreeClassifier from sklearn.tree. The second cell creates a DecisionTreeClassifier with max_depth = 5 and random_state = 42, fits it to the training data, and prints the training accuracy. The third cell does the same but also prints the validation accuracy. The output of the third cell shows the training accuracy as 0.885119855269109 and the validation accuracy as 0.885119855269109.

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

[81] tree = DecisionTreeClassifier(max_depth = 5, random_state=42)
tree.fit(x_train, y_train)
tree.score(x_train, y_train)

0.885119855269109

[88] tree = DecisionTreeClassifier(max_depth = 5, random_state=42)
tree.fit(x_train, y_train)
print('The Training Accuracy for max_depth 5 is:', format(5), tree.score(x_train, y_train))
print('The Validation Accuracy for max_depth 5 is:', format(5), tree.score(x_train, y_train))

The Training Accuracy for max_depth 5 is: 5 0.885119855269109
The Validation Accuracy for max_depth 5 is: 5 0.885119855269109
```