

Model Optimization and Tuning Phase Template

Date	15 July 2024
Team ID	team-740137
Project Title	Online Payments Fraud Detection
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining neural network models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (6 Marks):

Model	Tuned Hyperparameters	Optimal Values																
Random Forest Classifier	-	<div>1.Random Forest</div> <pre>[27]: rfc = RandomForestClassifier() rfc.fit(X_train,y_train) y_test_predict1 = rfc.predict(X_test) test_accuracy = accuracy_score(y_test,y_test_predict1) [28]: test_accuracy [28]: 0.990761581339245 [29]: y_train_predict1 = rfc.predict(X_train) train_accuracy = accuracy_score(y_train,y_train_predict1) train_accuracy [29]: 0.9999761581339245 [30]: pd.crosstab(y_test,y_test_predict1) [30]:</pre> <table><tr><th></th><th>col_0</th><th>0</th><th>1</th></tr><tr><td>Infraud</td><td></td><td></td><td></td></tr><tr><td>0</td><td>20840</td><td>4</td><td></td></tr><tr><td>1</td><td>46</td><td>175</td><td></td></tr></table> <pre>[31]: print(classification_report(y_test,y_test_predict1)) precision recall f1-score support 0 1.00 1.00 1.00 208404 1 0.99 0.79 0.88 221 accuracy 0.99 macro avg 0.99 0.90 0.94 208725 weighted avg 1.00 1.00 1.00 208725</pre>		col_0	0	1	Infraud				0	20840	4		1	46	175	
	col_0	0	1															
Infraud																		
0	20840	4																
1	46	175																

Decision Trees Classifier

2. Decision Tree

```
[32]: etc = DecisionTreeClassifier()
etc.fit(X_train,y_train)

y_test_predict3 = etc.predict(X_test)
test_accuracy = accuracy_score(y_test,y_test_predict3)
test_accuracy

[33]: 0.999813761335898

[34]: y_train_predict2 = etc.predict(X_train)
train_accuracy = accuracy_score(y_train,y_train_predict2)
train_accuracy

[35]: 1.0

[36]: pd.crosstab(y_test,y_test_predict2)

[37]: col_0    0    1
      -----
isfraud
0    209450  44
1      37    184

[38]: print(classification_report(y_test,y_test_predict2))

              precision    recall  f1-score   support

0       1.00        1.00        1.00    209494
1       0.81        0.83        0.82       221

accuracy: 0.999813761335898
macro avg: 0.9050000000000001 0.9150000000000001 0.9075000000000001 209715
weighted avg: 1.0000000000000001 1.0000000000000001 1.0000000000000001 209715
```

Extra Trees Classifier

3. ExtraTrees Classifier

```
[36]: etc = ExtraTreesClassifier()
etc.fit(X_train,y_train)

y_test_predict3 = etc.predict(X_test)
test_accuracy = accuracy_score(y_test,y_test_predict3)
test_accuracy

[37]: 0.99987276800316

[38]: y_train_predict3 = etc.predict(X_train)
train_accuracy = accuracy_score(y_train,y_train_predict3)
train_accuracy

[39]: 1.0

[40]: pd.crosstab(y_test,y_test_predict3)

[41]: col_0    0    1
      -----
isfraud
0    209492    2
1      51    170

[42]: print(classification_report(y_test,y_test_predict3))

              precision    recall  f1-score   support

0       1.00        1.00        1.00    209494
1       0.99        0.77        0.87       221

accuracy: 0.99987276800316
macro avg: 0.9950000000000001 0.8850000000000001 0.9375000000000001 209715
weighted avg: 1.0000000000000001 0.9900000000000001 0.9900000000000001 209715
```

SVM Classifier

4. SupportVectorMachine Classifier

```
[40]: svc = SVC()
svc.fit(X_train,y_train)

y_test_predict4 = svc.predict(X_test)
test_accuracy = accuracy_score(y_test,y_test_predict4)
test_accuracy

[41]: 0.9991758789295949

[42]: y_train_predict4 = svc.predict(X_train)
train_accuracy = accuracy_score(y_train,y_train_predict4)
train_accuracy

[43]: 0.999175894380408

[44]: pd.crosstab(y_test,y_test_predict4)

[45]: col_0    0    1
      -----
isfraud
0    209493    1
1      172    49

[46]: print(classification_report(y_test,y_test_predict4))

              precision    recall  f1-score   support

0       1.00        1.00        1.00    209494
1       0.30        0.22        0.26       221

accuracy: 0.9991758789295949
macro avg: 0.6500000000000001 0.6100000000000001 0.6300000000000001 209715
weighted avg: 1.0000000000000001 0.9900000000000001 0.9900000000000001 209715
```

XGboost

5.Xgboost Classifier

```
[47]: xgb1 = xgb.XGBClassifier()
xgb1.fit(X_train,y_train)

y_test_predict5 = xgb1.predict(X_test)
test_accuracy = accuracy_score(y_test,y_test_predict5)
test_accuracy

[47]: 0.9998235700832082

[48]: y_train_predict5 = xgb1.predict(X_train)
train_accuracy = accuracy_score(y_train,y_train_predict5)
train_accuracy

[48]: 0.9999356269222516

[49]: pd.crosstab(y_test,y_test_predict5)

[49]: col_0    0    1
fraud
0    204802    2
1         35    186

[50]: print(classification_report(y_test,y_test_predict5))

              precision    recall  f1-score   support

0               1.00         1.00         1.00      204804
1               0.99         0.94         0.95         221

 accuracy
macro avg   0.99         0.95         0.95      205025
weighted avg   1.00         1.00         1.00      205025
```

Performance Metrics Comparison Report (2 Marks):

Comparing the models

```
[51]: def compareModel():
    print("train accuracy for rfc",accuracy_score(y_train_predict1,y_train))
    print("test accuracy for rfc",accuracy_score(y_test_predict1,y_test))
    print("train accuracy for dtc",accuracy_score(y_train_predict2,y_train))
    print("test accuracy for dtc",accuracy_score(y_test_predict2,y_test))
    print("train accuracy for etc",accuracy_score(y_train_predict3,y_train))
    print("test accuracy for etc",accuracy_score(y_test_predict3,y_test))
    print("train accuracy for svc",accuracy_score(y_train_predict4,y_train))
    print("test accuracy for svc",accuracy_score(y_test_predict4,y_test))
    print("train accuracy for xgb1",accuracy_score(y_train_predict5,y_train1))
    print("test accuracy for xgb1",accuracy_score(y_test_predict5,y_test1))
compareModel()

train accuracy for rfc 0.9999976158119352
test accuracy for rfc 0.9997615811935245
train accuracy for dtc 1.0
test accuracy for dtc 0.9996137615335098
train accuracy for etc 1.0
test accuracy for etc 0.999747276065136
train accuracy for svc 0.9991178504160408
test accuracy for svc 0.9991750709295949
train accuracy for xgb1 0.9999356269222516
test accuracy for xgb1 0.9998235700832082
```

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Random Forest Classifier (RFC)	Performs exceptionally well with perfect accuracy metrics (Train accuracy: 1.000, Test accuracy: 1.000). It demonstrates excellent predictive performance and generalization ability, making it a robust choice for detecting fraudulent transactions.