Model Optimization and Tuning Phase Template

Date	July 2024
Team ID	740664
Project Title	Drug classification using machine learning
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining machine learning 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

Decision tree	`def decisionTree(x_train, x_test, y_train, y_test):`
	`dt=DecisionTreeClassifier()`
	`dt.fit(x_train,y_train)`
	`yPred = dt.predict(x_test)`
	`print('***DecisionTreeClassifier***')`
	`print('Confusion matrix')`
	`print(confusion_matrix(y_test,yPred))`
	`print('Classification report')`
	`print(classification_report(y_test,yPred))`

Random forest	```def randomForest(x_train, x_test, y_train, y_test):
	    rf = RandomForestClassifier()
	    rf.fit(x_train,y_train)
	    yPred = rf.predict(x_test)
	    print('***RandomForestClassifier***')
	    print('Confusion matrix')
	    print(confusion_matrix(y_test,yPred))
	    print('Classification report')
	    print(classification_report(y_test,yPred))```

Kneighbors	```def KNN(x_train, x_test, y_train, y_test):
	    knn = KNeighborsClassifier()
	    knn.fit(x_train,y_train)
	    yPred = knn.predict(x_test)
	    print('***KNeighborsClassifier***')
	    print('Confusion matrix')
	    print(confusion_matrix(y_test,yPred))
	    print('Classification report')
	    print(classification_report(y_test,yPred))```

Gradient boosting	```def xgboost (x_train, x_test, y_train, y_test):
	    xg = GradientBoostingClassifier()
	    xg.fit(x_train,y_train)
	    yPred = xg.predict(x_test)
	    print('***Gradient BoostingClassifier***')
	    print('Confusion matrix')
	    print(confusion_matrix(y_test,yPred))
	    print('Classification report')
	    print(classification_report (y_test, yPred))```

Performance Metrics Comparison Report (2 Marks):

Model	Baseline Metric	Optimized Metric

Decision tree	accuracy 0.97 60 macro avg 0.96 0.93 0.93 60 weighted avg 0.97 0.97 0.97 60	***DecisionTreeClassifier*** Confusion matrix [[25 0 0 0 0] [0 7 0 0 0] [0 2 4 0 0] [0 0 0 7 0] [0 0 0 0 15]] Classification report precision recall f1-score support DrugY 1.00 1.00 1.00 25 drugA 0.78 1.00 0.88 7 drugB 1.00 0.67 0.80 6 drugC 1.00 1.00 1.00 7 drugX 1.00 1.00 1.00 15
Random forest	accuracy 0.95 60 macro avg 0.94 0.90 0.91 60 weighted avg 0.96 0.95 0.95 60	***RandomForestClassifier*** Confusion matrix [[25 0 0 0 0] [0 7 0 0 0] [0 2 4 0 0] [0 0 0 6 1] [0 0 0 0 15]] Classification report precision recall f1-score support DrugY 1.00 1.00 1.00 25 drugA 0.78 1.00 0.88 7 drugB 1.00 0.67 0.80 6 drugC 1.00 0.86 0.92 7 drugX 0.94 1.00 0.97 15
Kneighbors	accuracy 0.37 60 macro avg 0.23 0.24 0.22 60 weighted avg 0.28 0.37 0.30 60	***KNeighborsClassifier*** Confusion matrix [[18 2 1 0 4] [6 0 0 0 1] [3 0 2 0 1] [5 0 0 0 2] [10 1 1 1 2]] Classification report precision recall f1-score support DrugY 0.43 0.72 0.54 25 drugA 0.00 0.00 0.00 7 drugB 0.50 0.33 0.40 6 drugC 0.00 0.00 0.00 7 drugX 0.20 0.13 0.16 15
Gradient boosting	accuracy 0.93 60 macro avg 0.93 0.87 0.88 60 weighted avg 0.94 0.93 0.93 60	***Gradient BoostingClassifier*** Confusion matrix [[25 0 0 0 0] [0 7 0 0 0] [0 2 3 0 1] [0 0 0 6 1] [0 0 0 0 15]] Classification report precision recall f1-score support DrugY 1.00 1.00 1.00 25 drugA 0.78 1.00 0.88 7 drugB 1.00 0.50 0.67 6 drugC 1.00 0.86 0.92 7 drugX 0.88 1.00 0.94 15

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Random forest	The Random forest model was selected for its superior performance, exhibiting high accuracy during hyperparameter tuning. Its ability to handle complex relationships, minimize overfitting, and optimize predictive accuracy aligns with project objectives, justifying its selection as the final model

```
# Decision tree and Random forest performs well
from sklearn.model_selection import cross_val_score
```

```
# Random forest model is selected
rf = RandomForestClassifier()
rf.fit(x_train,y_train)
yPred=rf.predict(x_test)
```

```
f1_score (yPred, y_test, average='weighted')
```

```
0.9516222084367246
```

```
cv = cross_val_score(rf,x,y,cv=5)
np.mean(cv)
```

```
0.985
```