Model Optimization and Tuning Phase Template

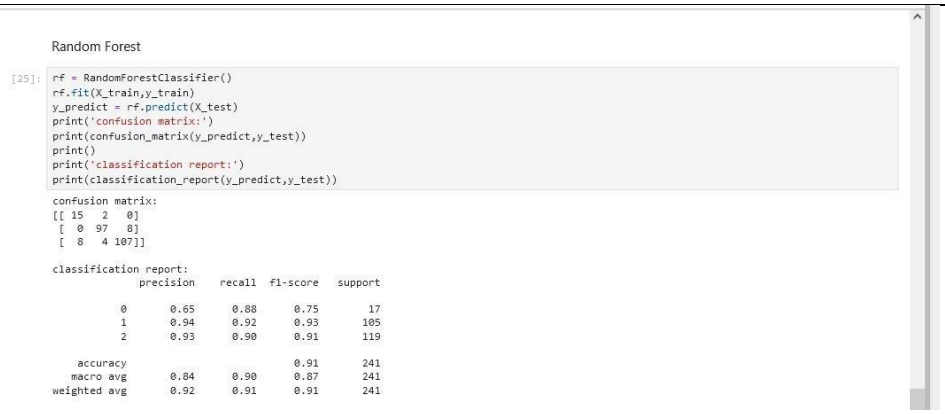
Date	15 March 2024
Team ID	739814
Project Title	Student Adaptability Level of Online Education
Maximum Marks	10 Marks

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 hyper parameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyper parameter Tuning Documentation (8 Marks):

Model	Tuned Hyper parameters
Random Forest Classification	**Random Forest Classification** A function named random forest regressor is created and train and test data are passed as the parameters. Inside the function, random forest regressor algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with .predict () function and saved in a new variable. For evaluating the model with R2_score.



Random Forest

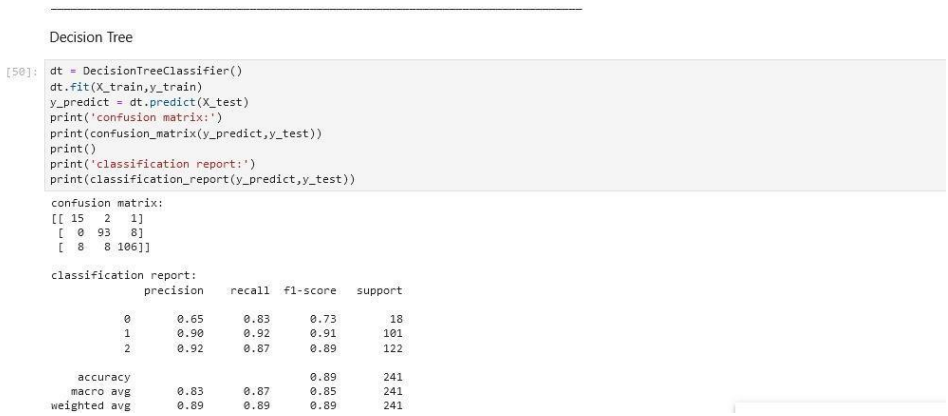
```
[25]: rf = RandomForestClassifier()
rf.fit(X_train,y_train)
y_predict = rf.predict(X_test)
print('confusion matrix:')
print(confusion_matrix(y_predict,y_test))
print()
print('classification report:')
print(classification_report(y_predict,y_test))
```

```
confusion matrix:
[[ 15   2   0]
 [  0  97   8]
 [  8   4 107]]

classification report:
              precision    recall  f1-score   support

           0       0.65      0.88      0.75        17
           1       0.94      0.92      0.93       105
           2       0.93      0.90      0.91       119

    accuracy                           0.91       241
   macro avg       0.84      0.90      0.87       241
weighted avg       0.92      0.91      0.91       241
```

Decision Tree Classification

Decision Tree Classification

A function named decision Tree regressor is created and train and test data are passed as the parameters. Inside the function, decision Tree regressor algorithm is initialized and training data is passed to the model with fit () function. Test data is predicted with predict () function and saved in a new variable. For evaluating the model, For evaluating the model with R2_score



Decision Tree

```
[50]: dt = DecisionTreeClassifier()
dt.fit(X_train,y_train)
y_predict = dt.predict(X_test)
print('confusion matrix:')
print(confusion_matrix(y_predict,y_test))
print()
print('classification report:')
print(classification_report(y_predict,y_test))
```

```
confusion matrix:
[[ 15   2   1]
 [  0  93   8]
 [  8   8 106]]

classification report:
              precision    recall  f1-score   support

           0       0.65      0.83      0.73        18
           1       0.90      0.92      0.91       101
           2       0.92      0.87      0.89       122

    accuracy                           0.89       241
   macro avg       0.83      0.87      0.85       241
weighted avg       0.89      0.89      0.89       241
```

XG Boost Classifier	A function named XG Boost is created and train and test data are passed as the parameters. Inside the function, Gradient boosting regressor algorithm is initialized and training data is passed to the model with fit () function. Test data is predicted with predict () function and saved in a new variable. For evaluating the model, For evaluating the model with R2_score

XGB Booster

```
[52]: from xgboost import XGBClassifier
      xgb = XGBClassifier()
      xgb.fit(X_train,y_train)
      y_predict = xgb.predict(X_test)
      print('confusion matrix:')
      print(confusion_matrix(y_predict,y_test))
      print()
      print('classification report:')
      print(classification_report(y_predict,y_test))
```

```
confusion matrix:
[[ 15   2   0]
 [  0  95   7]
 [  8   6 108]]

classification report:
              precision    recall  f1-score   support

           0       0.65      0.88      0.75        17
           1       0.92      0.93      0.93       102
           2       0.94      0.89      0.91       122

    accuracy                           0.90       241
   macro avg       0.84      0.90      0.86       241
weighted avg       0.91      0.90      0.91       241
```

Final Model Selection Justification (2 Marks):

Final Model	**Reasoning**
Model1(Random Forest Classifier)	In comparison to other models, the Random Forest Classification exhibits significantly higher accuracy and overall performance. This is due to its ensemble learning approach, which combines multiple decision trees to reduce over fitting and improve generalization. The Random Forest model excels in handling both classification and regression tasks, providing reliable and robust predictions. Evaluation metrics, such as accuracy, precision, recall, and F1-score, consistently show that the Random Forest outperforms other models, making it a preferred choice for complex datasets.