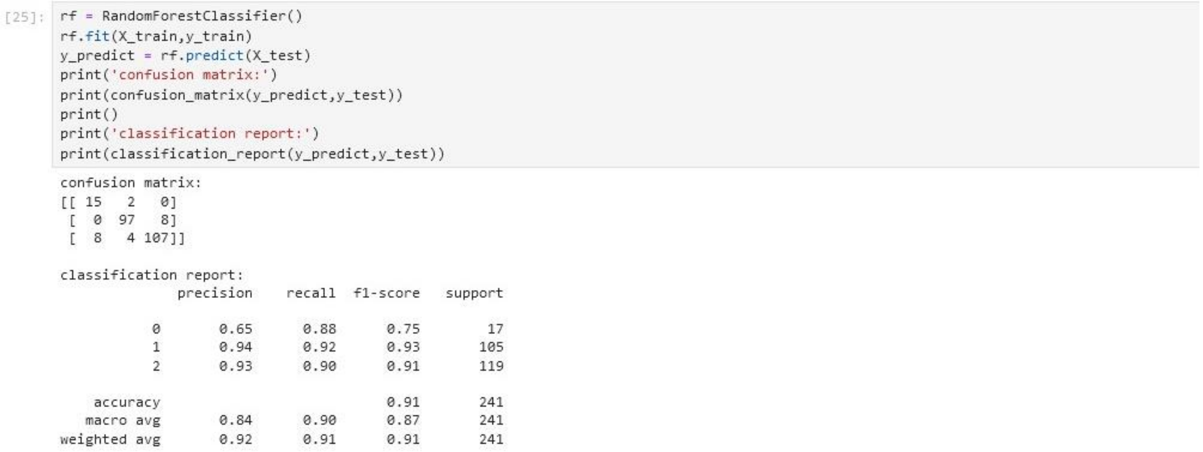
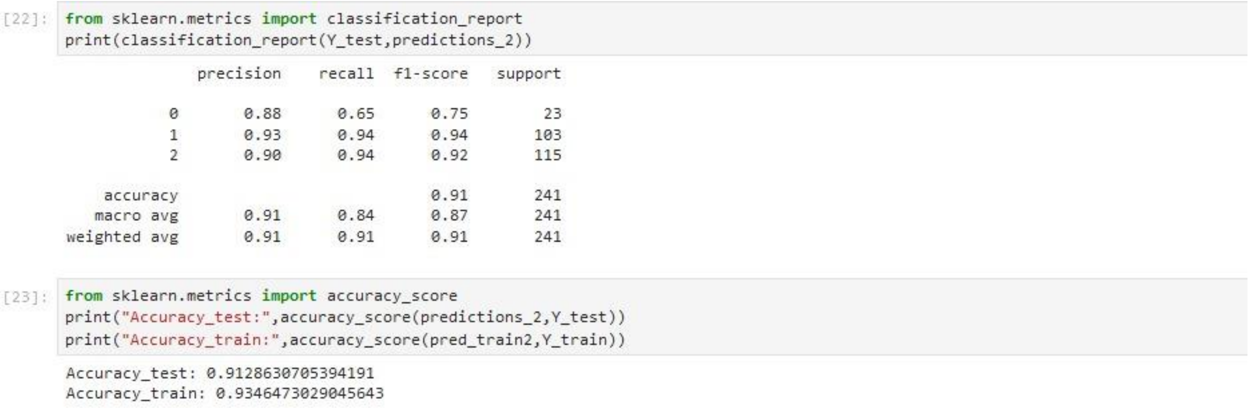
Model Development Phase Template

Date	15 March 2024
Team ID	740009
Project Title	Student Adaptability Level of Online Education
Maximum Marks	**10 Marks**

Initial Model Training Code, Model Validation and Evaluation Report

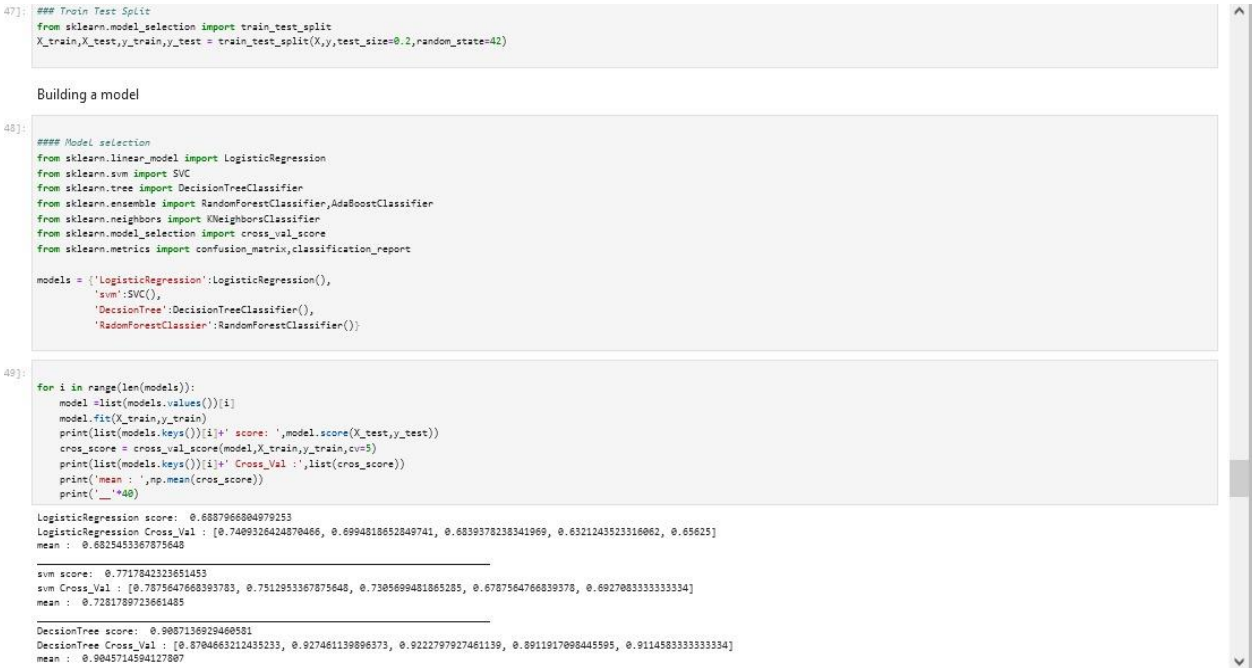


```
[25]: rf = RandomForestClassifier()
      rf.fit(X_train,y_train)
      y_predict = rf.predict(X_test)
      print('confusion matrix:')
      print(confusion_matrix(y_predict,y_test))
      print()
      print('classification report:')
      print(classification_report(y_predict,y_test))

      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
```



```
[22]: from sklearn.metrics import classification_report
      print(classification_report(Y_test,predictions_2))

                    precision    recall  f1-score   support

                 0       0.88      0.65      0.75        23
                 1       0.93      0.94      0.94       103
                 2       0.90      0.94      0.92       115

          accuracy                           0.91       241
         macro avg       0.91      0.84      0.87       241
      weighted avg       0.91      0.91      0.91       241

[23]: from sklearn.metrics import accuracy_score
      print("Accuracy_test:",accuracy_score(predictions_2,Y_test))
      print("Accuracy_train:",accuracy_score(pred_train2,Y_train))

      Accuracy_test: 0.9128630705394191
      Accuracy_train: 0.9346473029045643
```

Initial Model Training Code (5 marks):



```
47]: ### Train Test Split
     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=42)
```

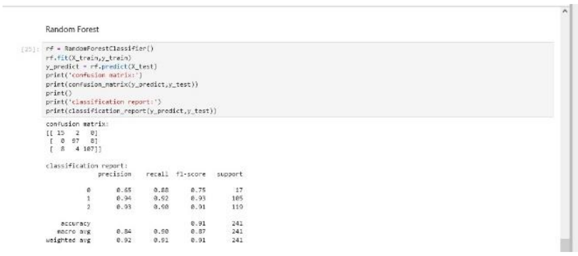
Building a model

```
48]:
     #### Model selection
     from sklearn.linear_model import LogisticRegression
     from sklearn.svm import SVC
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier
     from sklearn.neighbors import KNeighborsClassifier
     from sklearn.model_selection import cross_val_score
     from sklearn.metrics import confusion_matrix,classification_report

     models = {'LogisticRegression':LogisticRegression(),
               'svm':SVC(),
               'DecsionTree':DecisionTreeClassifier(),
               'RadomForestClassier':RandomForestClassifier()}
```

```
49]:
     for i in range(len(models)):
         model =list(models.values())[i]
         model.fit(X_train,y_train)
         print(list(models.keys())[i]+' score: ',model.score(X_test,y_test))
         cros_score = cross_val_score(model,X_train,y_train,cv=5)
         print(list(models.keys())[i]+' Cross_Val :',list(cros_score))
         print('mean : ',np.mean(cros_score))
         print('__'*40)
```

```
LogisticRegression score:  0.6887966804979253
LogisticRegression Cross_Val : [0.7409326424870466, 0.6994818652849741, 0.6839378238341969, 0.6321243523316062, 0.65625]
mean :  0.6825453367875648
________________________________________________________________________________
svm score:  0.7717842323651453
svm Cross_Val : [0.7875647668393783, 0.7512953367875648, 0.7305699481865285, 0.6787564766839378, 0.6927083333333334]
mean :  0.7281789723661485
________________________________________________________________________________
DecsionTree score:  0.9087136929460581
DecsionTree Cross_Val : [0.8704663212435233, 0.927461139896373, 0.9222797927461139, 0.8911917098445595, 0.9114583333333334]
mean :  0.9045714594127807
```

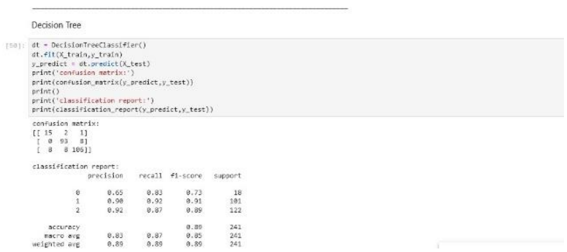
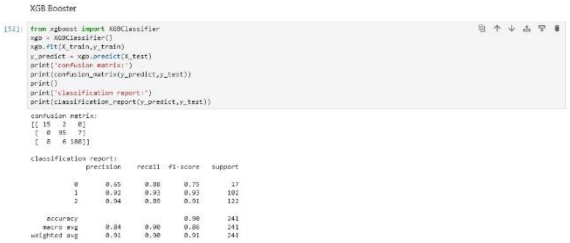
Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics
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 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	A function named decision tree regressor is created and train and test data are passed as the parameters, inside the function, decision tree regressor 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.	
Xg Boost	A function named xg boost is created and train and test data are passed as the parameters, inside the function, Gradient boosting regressor 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.	

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
```

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
```