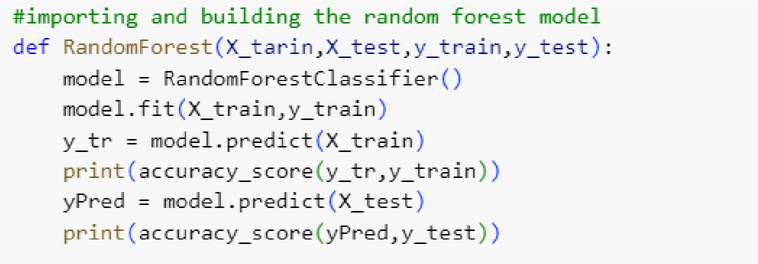
Model Development Phase Template

Date	02 may 2024
Team ID	738323
Project Title	SmartLender - Applicant Credibility Prediction for Loan Approval
Maximum Marks	4 Marks

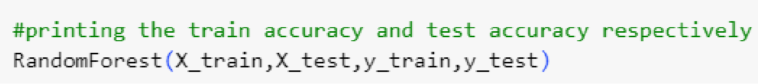
Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

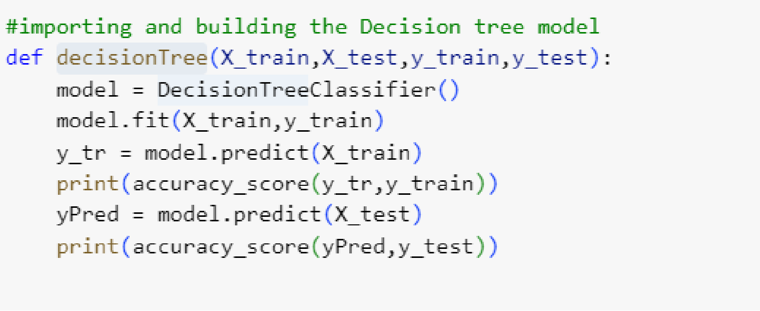
Initial Model Training Code:



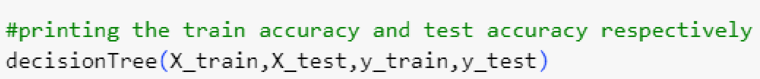
```
#importing and building the random forest model
def RandomForest(X_tarin,X_test,y_train,y_test):
    model = RandomForestClassifier()
    model.fit(X_train,y_train)
    y_tr = model.predict(X_train)
    print(accuracy_score(y_tr,y_train))
    yPred = model.predict(X_test)
    print(accuracy_score(yPred,y_test))
```



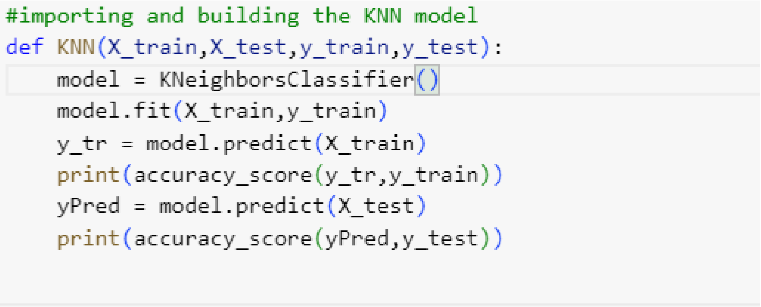
```
#printing the train accuracy and test accuracy respectively
RandomForest(X_train,X_test,y_train,y_test)
```



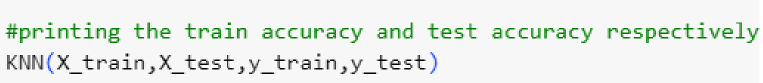
```
#importing and building the Decision tree model
def decisionTree(X_train,X_test,y_train,y_test):
    model = DecisionTreeClassifier()
    model.fit(X_train,y_train)
    y_tr = model.predict(X_train)
    print(accuracy_score(y_tr,y_train))
    yPred = model.predict(X_test)
    print(accuracy_score(yPred,y_test))
```



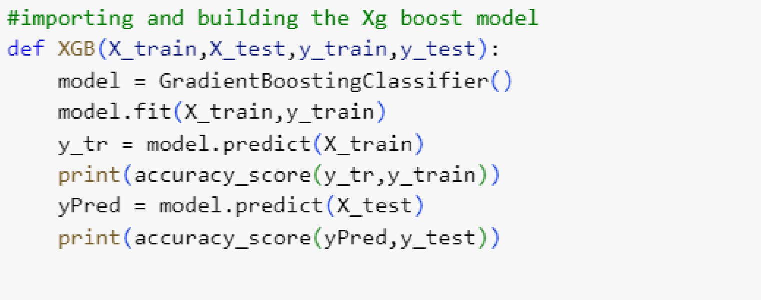
```
#printing the train accuracy and test accuracy respectively
decisionTree(X_train,X_test,y_train,y_test)
```



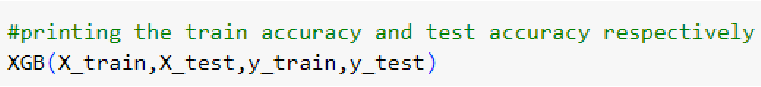
```
#importing and building the KNN model
def KNN(X_train,X_test,y_train,y_test):
    model = KNeighborsClassifier()
    model.fit(X_train,y_train)
    y_tr = model.predict(X_train)
    print(accuracy_score(y_tr,y_train))
    yPred = model.predict(X_test)
    print(accuracy_score(yPred,y_test))
```



```
#printing the train accuracy and test accuracy respectively
KNN(X_train,X_test,y_train,y_test)
```

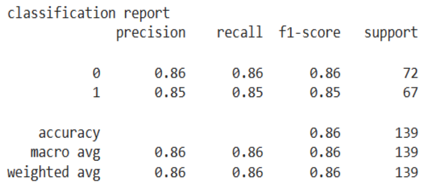
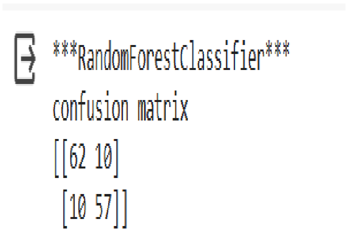


```
#importing and building the Xg boost model
def XGB(X_train,X_test,y_train,y_test):
    model = GradientBoostingClassifier()
    model.fit(X_train,y_train)
    y_tr = model.predict(X_train)
    print(accuracy_score(y_tr,y_train))
    yPred = model.predict(X_test)
    print(accuracy_score(yPred,y_test))
```



```
#printing the train accuracy and test accuracy respectively
XGB(X_train,X_test,y_train,y_test)
```

Model Validation and Evaluation Report:

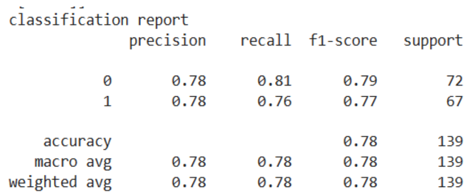
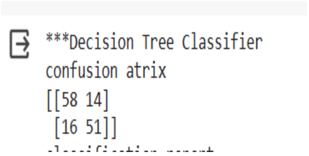
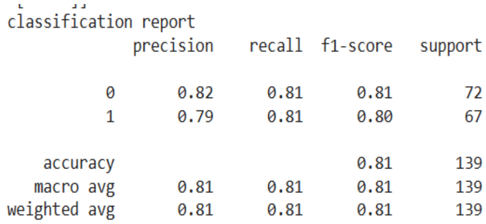
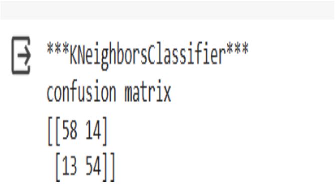
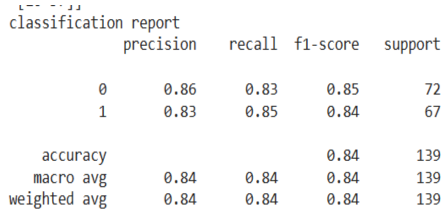
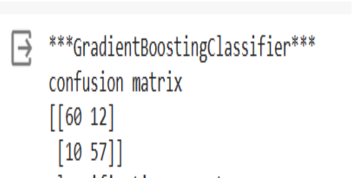
Model	Classification Report	F1 Score	Confusion Matrix
Random Forest		86%	

```
classification report
              precision    recall  f1-score   support

           0       0.86      0.86      0.86        72
           1       0.85      0.85      0.85        67

    accuracy                           0.86       139
   macro avg       0.86      0.86      0.86       139
weighted avg       0.86      0.86      0.86       139
```

```
***RandomForestClassifier***
confusion matrix
[[62 10]
 [10 57]]
```

Decision Tree	 classification report precision recall f1-score support 0 0.78 0.81 0.79 72 1 0.78 0.76 0.77 67 accuracy 0.78 139 macro avg 0.78 0.78 0.78 139 weighted avg 0.78 0.78 0.78 139	78%	 ***Decision Tree Classifier confusion atrix [[58 14] [16 51]]
KNN	 classification report precision recall f1-score support 0 0.82 0.81 0.81 72 1 0.79 0.81 0.80 67 accuracy 0.81 139 macro avg 0.81 0.81 0.81 139 weighted avg 0.81 0.81 0.81 139	81%	 ***KNeighborsClassifier*** confusion matrix [[58 14] [13 54]]
Gradient Boosting	 classification report precision recall f1-score support 0 0.86 0.83 0.85 72 1 0.83 0.85 0.84 67 accuracy 0.84 139 macro avg 0.84 0.84 0.84 139 weighted avg 0.84 0.84 0.84 139	85%	 ***GradientBoostingClassifier*** confusion matrix [[60 12] [10 57]]