Model Development Phase Template

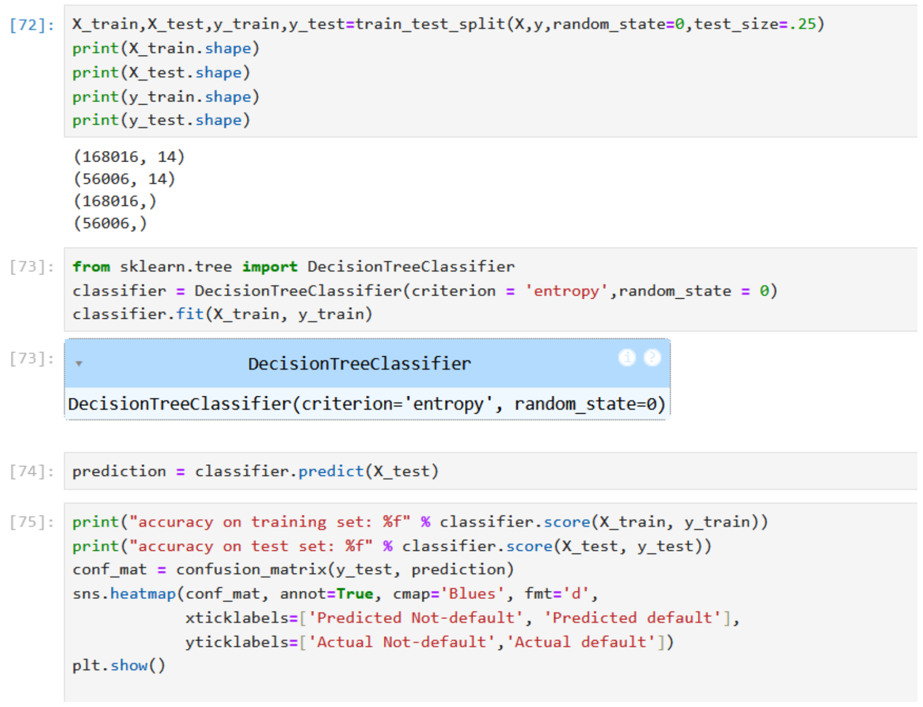
Date	05 June 2024
Team ID	737568
Project Title	AutoForesight : A Predictive Model for Streamlining Car Loan Repayment Planning
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

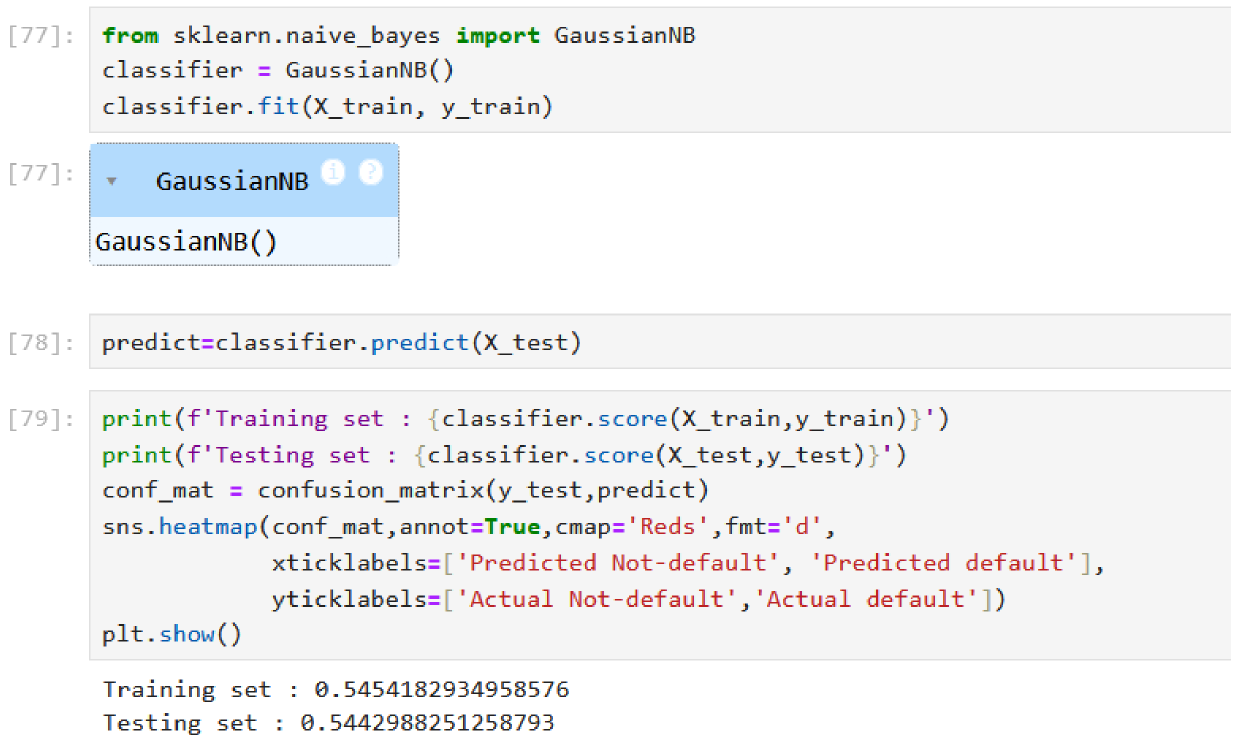
The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:

Paste the screenshot of the model training code

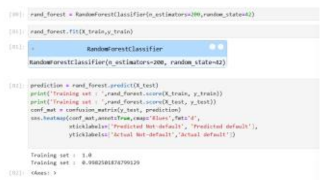
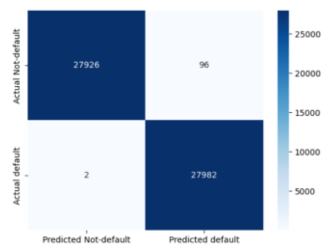
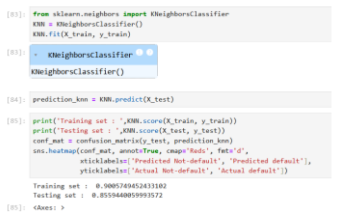
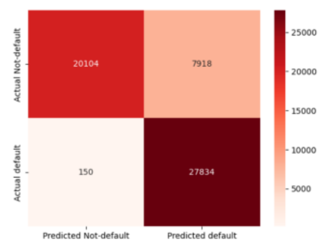


```
[72]: X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=0,test_size=.25)
      print(X_train.shape)
      print(X_test.shape)
      print(y_train.shape)
      print(y_test.shape)

      (168016, 14)
      (56006, 14)
      (168016,)
      (56006,)

[73]: from sklearn.tree import DecisionTreeClassifier
      classifier = DecisionTreeClassifier(criterion = 'entropy',random_state = 0)
      classifier.fit(X_train, y_train)

[73]: DecisionTreeClassifier
      DecisionTreeClassifier(criterion='entropy', random_state=0)

[74]: prediction = classifier.predict(X_test)

[75]: print("accuracy on training set: %f" % classifier.score(X_train, y_train))
      print("accuracy on test set: %f" % classifier.score(X_test, y_test))
      conf_mat = confusion_matrix(y_test, prediction)
      sns.heatmap(conf_mat, annot=True, cmap='Blues', fmt='d',
                  xticklabels=['Predicted Not-default', 'Predicted default'],
                  yticklabels=['Actual Not-default','Actual default'])
      plt.show()
```



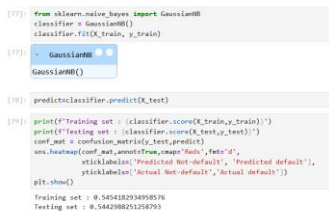
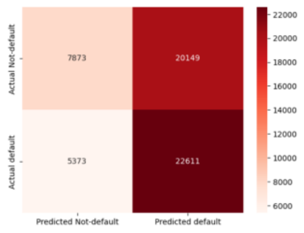
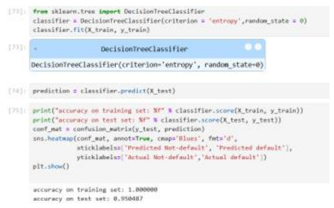
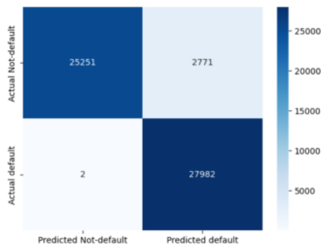
```
[77]: from sklearn.naive_bayes import GaussianNB
      classifier = GaussianNB()
      classifier.fit(X_train, y_train)

[77]:  ▾ GaussianNB
      GaussianNB()

[78]: predict=classifier.predict(X_test)

[79]: print(f'Training set : {classifier.score(X_train,y_train)}')
      print(f'Testing set : {classifier.score(X_test,y_test)}')
      conf_mat = confusion_matrix(y_test,predict)
      sns.heatmap(conf_mat,annot=True,cmap='Reds',fmt='d',
                  xticklabels=['Predicted Not-default', 'Predicted default'],
                  yticklabels=['Actual Not-default','Actual default'])
      plt.show()

      Training set : 0.5454182934958576
      Testing set : 0.5442988251258793
```

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix
Random Forest		99.8%	
K Nearest Neighbors		85.5%	

```
[80]: rand_forest = RandomForestClassifier(n_estimators=200,random_state=42)

[81]: rand_forest.fit(X_train,y_train)

[81]:  RandomForestClassifier
      RandomForestClassifier(n_estimators=200, random_state=42)

[82]: prediction = rand_forest.predict(X_test)
      print('Training set : ',rand_forest.score(X_train, y_train))
      print('Training set : ',rand_forest.score(X_test, y_test))
      conf_mat = confusion_matrix(y_test, prediction)
      sns.heatmap(conf_mat,annot=True,cmap='Blues',fmt='d',
                  xticklabels=['Predicted Not-default', 'Predicted default'],
                  yticklabels=['Actual Not-default','Actual default'])

      Training set :  1.0
      Training set :  0.9982501874799129
[82]: <Axes: >
```

```
[83]: from sklearn.neighbors import KNeighborsClassifier
      KNN = KNeighborsClassifier()
      KNN.fit(X_train, y_train)

[83]:  ▾ KNeighborsClassifier
      KNeighborsClassifier()

[84]: prediction_knn = KNN.predict(X_test)

[85]: print('Training set : ',KNN.score(X_train, y_train))
      print('Testing set : ',KNN.score(X_test, y_test))
      conf_mat = confusion_matrix(y_test, prediction_knn)
      sns.heatmap(conf_mat, annot=True, cmap='Reds', fmt='d',
                  xticklabels=['Predicted Not-default', 'Predicted default'],
                  yticklabels=['Actual Not-default', 'Actual default'])

      Training set :  0.9005749452433102
      Testing set :  0.8559440059993572
[85]: <Axes: >
```

Gaussian NB		54.4%	
Decision Tree Classifier		99.8%	

```
[77]: from sklearn.naive_bayes import GaussianNB
      classifier = GaussianNB()
      classifier.fit(X_train, y_train)

[77]: GaussianNB
      GaussianNB()

[78]: predict=classifier.predict(X_test)

[79]: print(f'Training set : {classifier.score(X_train,y_train)}')
      print(f'Testing set : {classifier.score(X_test,y_test)}')
      conf_mat = confusion_matrix(y_test,predict)
      sns.heatmap(conf_mat,annot=True,cmap='Reds',fmt='d',
                  xticklabels=['Predicted Not-default', 'Predicted default'],
                  yticklabels=['Actual Not-default','Actual default'])
      plt.show()

      Training set : 0.5454182934958576
      Testing set : 0.5442988251258793
```

```
[73]: from sklearn.tree import DecisionTreeClassifier
      classifier = DecisionTreeClassifier(criterion = 'entropy',random_state = 0)
      classifier.fit(X_train, y_train)

[73]: DecisionTreeClassifier
      DecisionTreeClassifier(criterion='entropy', random_state=0)

[74]: prediction = classifier.predict(X_test)

[75]: print("accuracy on training set: %f" % classifier.score(X_train, y_train))
      print("accuracy on test set: %f" % classifier.score(X_test, y_test))
      conf_mat = confusion_matrix(y_test, prediction)
      sns.heatmap(conf_mat, annot=True, cmap='Blues', fmt='d',
                  xticklabels=['Predicted Not-default', 'Predicted default'],
                  yticklabels=['Actual Not-default','Actual default'])
      plt.show()

      accuracy on training set: 1.000000
      accuracy on test set: 0.950487
```