

## Model Development Phase Template

Date	15 <sup>th</sup> July 2024
Team ID	739924
Project Title	Auto Foresight : 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)

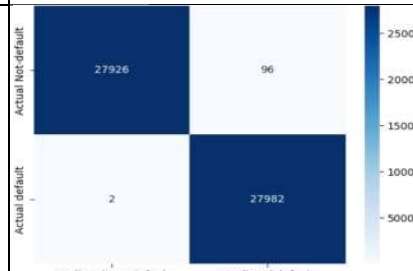
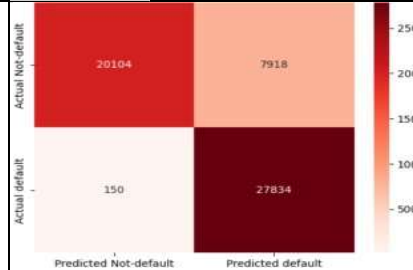


[77]: GaussianNB
      GaussianNB()

[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	<pre>[80]: rand_forest = RandomForestClassifier(n_estimators=100,random_state=42) [81]: rand_forest.fit(X_train,y_train) [82]: RandomForestClassifier RandomForestClassifier(n_estimators=200, random_state=42)  [83]: 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 : 0.9982181878706129 [84]: (Axes: 1)</pre>	99.8%	 <table border="1"><thead><tr><th></th><th>Predicted Not-default</th><th>Predicted default</th></tr></thead><tbody><tr><th>Actual Not-default</th><td>27926</td><td>96</td></tr><tr><th>Actual default</th><td>2</td><td>27982</td></tr></tbody></table>		Predicted Not-default	Predicted default	Actual Not-default	27926	96	Actual default	2	27982
	Predicted Not-default	Predicted default										
Actual Not-default	27926	96										
Actual default	2	27982										
K Nearest Neighbors	<pre>[83]: from sklearn.neighbors import KNeighborsClassifier KNN = KNeighborsClassifier() KNN.fit(X_train, y_train) [84]: 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 : 0.9005740452431102 Testing set : 0.8559440059993572 [86]: (Axes: 1)</pre>	85.5%	 <table border="1"><thead><tr><th></th><th>Predicted Not-default</th><th>Predicted default</th></tr></thead><tbody><tr><th>Actual Not-default</th><td>20104</td><td>7918</td></tr><tr><th>Actual default</th><td>150</td><td>27834</td></tr></tbody></table>		Predicted Not-default	Predicted default	Actual Not-default	20104	7918	Actual default	150	27834
	Predicted Not-default	Predicted default										
Actual Not-default	20104	7918										
Actual default	150	27834										
Gaussian NB	<pre>[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</pre>	54.4%	 <table border="1"><thead><tr><th></th><th>Predicted Not-default</th><th>Predicted default</th></tr></thead><tbody><tr><th>Actual Not-default</th><td>7873</td><td>20149</td></tr><tr><th>Actual default</th><td>5373</td><td>22611</td></tr></tbody></table>		Predicted Not-default	Predicted default	Actual Not-default	7873	20149	Actual default	5373	22611
	Predicted Not-default	Predicted default										
Actual Not-default	7873	20149										
Actual default	5373	22611										
Decision Tree Classifier	<pre>[77]: from sklearn.tree import DecisionTreeClassifier classifier = DecisionTreeClassifier(criterion = 'entropy',random_state = 0) classifier.fit(X_train, y_train) [77]: DecisionTreeClassifier DecisionTreeClassifier(criterion='entropy', random_state=0)  [78]: prediction = classifier.predict(X_test)  [79]: 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</pre>	99.8%	 <table border="1"><thead><tr><th></th><th>Predicted Not-default</th><th>Predicted default</th></tr></thead><tbody><tr><th>Actual Not-default</th><td>25251</td><td>2771</td></tr><tr><th>Actual default</th><td>2</td><td>27982</td></tr></tbody></table>		Predicted Not-default	Predicted default	Actual Not-default	25251	2771	Actual default	2	27982
	Predicted Not-default	Predicted default										
Actual Not-default	25251	2771										
Actual default	2	27982										