

Model Development Phase Template

Date	11 July 2024
Team ID	740023
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:

```
#decision tree model
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report
def DecisionTree(X_train,X_test,Y_train,Y_test):
    # Indent the code within the function
    dt=DecisionTreeClassifier()
    dt.fit(X_train,Y_train)
    yPred=dt.predict(X_test)
    acc=accuracy_score(yPred,Y_test)
    yPred_train = dt.predict(X_train)
    acc1=accuracy_score(yPred_train,Y_train)
    print('***DecisionTreeClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(Y_test,yPred))
    print('Classification report')
    print(classification_report(Y_test,yPred))
    print('Accuracy Score of testing: ', acc)
    print('Accuracy Score of training: ', acc1)

[ ] #printing the train accuracy and test accuracy respectively
DecisionTree(X_train,X_test,Y_train,Y_test)
```

```

▶ #Random forest model
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report # Import necessary functions
def RandomForest(X_train,X_test,Y_train,Y_test):
    rf=RandomForestClassifier()
    rf.fit(X_train,Y_train)
    yPred=rf.predict(X_test)
    acc=accuracy_score(yPred,Y_test)
    yPred_train = rf.predict(X_train)
    acc1=accuracy_score(yPred_train,Y_train)
    print('***RandomForestClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(Y_test,yPred))
    print('Classification report')
    print(classification_report(Y_test,yPred))
    print('Accuracy Score of testing: ', acc)
    print('Accuracy Score of training: ', acc1)

```

```

[ ] #printing the train accuracy and test accuracy respectively
RandomForest(X_train,X_test,Y_train,Y_test)

```

```

▶ #KNN model
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, classification_report
def KNN(X_train,X_test,Y_train,Y_test):
    knn=KNeighborsClassifier()
    knn.fit(X_train,Y_train)
    yPred=knn.predict(X_test)
    acc=accuracy_score(yPred,Y_test)
    yPred_train = knn.predict(X_train)
    acc1=accuracy_score(yPred_train,Y_train)
    print('***KNeighborsClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(Y_test,yPred))
    print('Classification report')
    print(classification_report(Y_test,yPred))
    print('Accuracy Score of testing: ', acc)
    print('Accuracy Score of training: ', acc1)

```

```

[97] #printing the train accuracy and test accuracy respectively
KNN(X_train,X_test,Y_train,Y_test)

```

```

▶ #xgboost model
from sklearn.metrics import accuracy_score
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import confusion_matrix, classification_report
def xgboost(X_train,X_test,Y_train,Y_test):
    xg=GradientBoostingClassifier()
    xg.fit(X_train,Y_train)
    yPred=xg.predict(X_test)
    acc=accuracy_score(yPred,Y_test)
    yPred_train = xg.predict(X_train)
    acc1=accuracy_score(yPred_train,Y_train)
    print('***GradientBoostingClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(Y_test,yPred))
    print('Classification report')
    print(classification_report(Y_test,yPred))
    print('Accuracy Score of testing: ', acc)
    print('Accuracy Score of training: ', acc1)


```

```

[99] #printing the train accuracy and test accuracy respectively
xgboost(X_train,X_test,Y_train,Y_test)

```

Model Validation and Evaluation Report:

Model	Classification Report	F1 Score	Confusion Matrix																														
Decision Tree	<div><div></div><pre>print(classification_report(Y_test,yPred))</pre><div>Classification report</div><table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>0</td><td>0.60</td><td>0.75</td><td>0.67</td><td>99</td></tr><tr><td>1</td><td>0.75</td><td>0.61</td><td>0.67</td><td>125</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.67</td><td>224</td></tr><tr><td>macro avg</td><td>0.68</td><td>0.68</td><td>0.67</td><td>224</td></tr><tr><td>weighted avg</td><td>0.69</td><td>0.67</td><td>0.67</td><td>224</td></tr></tbody></table><div>Accuracy Score of testing: 0.6696428571428571</div><div>Accuracy Score of training: 1.0</div></div>		precision	recall	f1-score	support	0	0.60	0.75	0.67	99	1	0.75	0.61	0.67	125	accuracy			0.67	224	macro avg	0.68	0.68	0.67	224	weighted avg	0.69	0.67	0.67	224	67%	<pre>print(confusion_matrix(Y_test,yPred))</pre> <div>Confusion matrix</div> <pre>[[74 25] [49 76]]</pre>
	precision	recall	f1-score	support																													
0	0.60	0.75	0.67	99																													
1	0.75	0.61	0.67	125																													
accuracy			0.67	224																													
macro avg	0.68	0.68	0.67	224																													
weighted avg	0.69	0.67	0.67	224																													

Random Forest	<pre>print(classification_report(Y_test,yPred))</pre> <table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>0</td><td>0.69</td><td>0.79</td><td>0.74</td><td>99</td></tr><tr><td>1</td><td>0.81</td><td>0.72</td><td>0.76</td><td>125</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.75</td><td>224</td></tr><tr><td>macro avg</td><td>0.75</td><td>0.75</td><td>0.75</td><td>224</td></tr><tr><td>weighted avg</td><td>0.76</td><td>0.75</td><td>0.75</td><td>224</td></tr></tbody></table> <p>Accuracy Score of testing: 0.75 Accuracy Score of training: 1.0</p>		precision	recall	f1-score	support	0	0.69	0.79	0.74	99	1	0.81	0.72	0.76	125	accuracy			0.75	224	macro avg	0.75	0.75	0.75	224	weighted avg	0.76	0.75	0.75	224	75%	<pre>print(confusion_matrix(Y_test,yPred))</pre> <p>Confusion matrix</p> <pre>[[78 21] [35 90]]</pre>
	precision	recall	f1-score	support																													
0	0.69	0.79	0.74	99																													
1	0.81	0.72	0.76	125																													
accuracy			0.75	224																													
macro avg	0.75	0.75	0.75	224																													
weighted avg	0.76	0.75	0.75	224																													
KNN	<pre>print(classification_report(Y_test,yPred))</pre> <table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>0</td><td>0.62</td><td>0.67</td><td>0.65</td><td>101</td></tr><tr><td>1</td><td>0.69</td><td>0.64</td><td>0.66</td><td>114</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.66</td><td>215</td></tr><tr><td>macro avg</td><td>0.66</td><td>0.66</td><td>0.66</td><td>215</td></tr><tr><td>weighted avg</td><td>0.66</td><td>0.66</td><td>0.66</td><td>215</td></tr></tbody></table>		precision	recall	f1-score	support	0	0.62	0.67	0.65	101	1	0.69	0.64	0.66	114	accuracy			0.66	215	macro avg	0.66	0.66	0.66	215	weighted avg	0.66	0.66	0.66	215	66%	<pre>print(confusion_matrix(Y_test,yPred))</pre> <p>Confusion matrix</p> <pre>[[68 33] [41 73]]</pre>
	precision	recall	f1-score	support																													
0	0.62	0.67	0.65	101																													
1	0.69	0.64	0.66	114																													
accuracy			0.66	215																													
macro avg	0.66	0.66	0.66	215																													
weighted avg	0.66	0.66	0.66	215																													

Gradient Boosting

print(classification_report(Y_test,yPred))

Classification report

	precision	recall	f1-score	support
0	0.63	0.74	0.68	99
1	0.76	0.66	0.71	125
accuracy			0.70	224
macro avg	0.70	0.70	0.70	224
weighted avg	0.71	0.70	0.70	224

Accuracy Score of testing: 0.6964285714285714
Accuracy Score of training: 0.9337748344370861

70%

print(confusion_matrix(Y_test,yPred))

Confusion matrix

[[73 26]
[42 83]]