

Model Development Phase Template

Date	July 2024
Team ID	739865
Project Title	Prosperity Prognosticator : Machine Learning for Startup success Prediction
Maximum Marks	10 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 a summary and training and validation performance metrics for multiple models, presented through respective screenshots.

Initial Model Training Code (5 marks):

```
#importing and building the random forest classifier model
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train.get_numeric_data(), y_train)
y_pred_rf = rf.predict(X_test.get_numeric_data())
print("Training Accuracy :", rf.score(X_train.get_numeric_data(), y_train))
print("Testing Accuracy :", rf.score(X_test.get_numeric_data(), y_test))
```

```
#importing and building the GradientBoostingClassifier model
from sklearn.ensemble import GradientBoostingClassifier
#train
gbc = GradientBoostingClassifier(learning_rate=0.02,
                                max_depth=4,
                                random_state=100, n_estimators=1000)
gbc.fit(X_train, y_train)
#predict
y_predicted_gb = gbc.predict(X_test)
print("Training Accuracy :", gbc.score(X_train, y_train))
print("Testing Accuracy :", gbc.score(X_test, y_test))
```

```
#importing and building the XGBClassifier model
from xgboost import XGBClassifier
#train
xgb = XGBClassifier()
xgb.fit(X_train,y_train)
#predict
y_predicted_xgb = xgb.predict(X_test)
print("Training Accuracy :", xgb.score(X_train, y_train))
print("Testing Accuracy :", xgb.score(X_test, y_test))
```

```
#importing and building the AdaBoostClassifier model
from sklearn.ensemble import AdaBoostClassifier
#train
ada = AdaBoostClassifier()
ada.fit(X_train,y_train)
#predict
y_predicted_ab = ada.predict(X_test)
print("Training Accuracy :", ada.score(X_train, y_train))
print("Testing Accuracy :", ada.score(X_test, y_test))
```

```
# Gathering accuracy score for each model
scores = {
    'AdaBoostClassifier': {
        'Accuracy_score': accuracy_score(y_test, y_predicted_ab)
    },
    'XGB classifier': {
        'Accuracy_score': accuracy_score(y_test, y_predicted_xgb)
    },
    'Random Forest': {
        'Accuracy_score': accuracy_score(y_test, y_pred_rf)
    },
    'Gradient Boosting':{
        'Accuracy_score': accuracy_score(y_test, y_predicted_gb)
    }
}
# Plotting comparison of each model
scores = pd.DataFrame(scores)
scores.plot(kind="barh",figsize=(10, 10)).legend(loc='upper center', ncol=3, title="Machine Learning Model")
```

Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics																														
Model 1	Gradient Boosting Classifier model typically include accuracy, precision, recall, F1 score to evaluate its predictive performance and generalization capability.	<pre># gathering accuracy score for each model scores = { 'AdaBoostClassifier': { 'Accuracy_score': accuracy_score(y_test, y_predicted_ab) }, 'XGB Classifier': { 'Accuracy_score': accuracy_score(y_test, y_predicted_xgb) }, 'Random Forest': { 'Accuracy_score': accuracy_score(y_test, y_pred_rf) }, 'Gradient Boosting':{ 'Accuracy_score': accuracy_score(y_test, y_predicted_gb) } } # plotting comparison of each model scores = pd.DataFrame(scores) scores.plot(kind="barh",figsize=(10, 10),legend(loc='upper center', ncol=3), title='Machine Learning Model')</pre>																														
Model 2	AdaBoost classifier model commonly include accuracy, precision, recall, F1 score which help assess the model's prediction accuracy and generalizability	<pre>from sklearn.ensemble import AdaBoostClassifier from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score ada = AdaBoostClassifier() ada.fit(X_train,y_train) y_predicted_ab = ada.predict(X_test) print("Training Accuracy :", ada.score(X_train, y_train)) print("Testing Accuracy :", ada.score(X_test, y_test)) cr = classification_report(y_test, y_predicted_ab) print(cr) false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, y_predicted_ab) roc_auc = auc(false_positive_rate, true_positive_rate) print("ROC AUC",roc_auc) precision, recall, thresholds = precision_recall_curve(y_test, y_predicted_ab) f1 = f1_score(y_test, y_predicted_ab) precision_recall_auc = auc(recall, precision) print("Precision-Recall AUC",precision_recall_auc) # 62s Training Accuracy : 0.8328171178613883 Testing Accuracy : 0.776173285198534</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>0.72</td><td>0.68</td><td>0.69</td><td>98</td></tr><tr><td>1</td><td>0.88</td><td>0.87</td><td>0.88</td><td>179</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.78</td><td>277</td></tr><tr><td>macro avg</td><td>0.76</td><td>0.78</td><td>0.74</td><td>277</td></tr><tr><td>weighted avg</td><td>0.77</td><td>0.78</td><td>0.77</td><td>277</td></tr></tbody></table>		precision	recall	f1-score	support	0	0.72	0.68	0.69	98	1	0.88	0.87	0.88	179	accuracy			0.78	277	macro avg	0.76	0.78	0.74	277	weighted avg	0.77	0.78	0.77	277
	precision	recall	f1-score	support																												
0	0.72	0.68	0.69	98																												
1	0.88	0.87	0.88	179																												
accuracy			0.78	277																												
macro avg	0.76	0.78	0.74	277																												
weighted avg	0.77	0.78	0.77	277																												
Model 3	Random forest classifier model often encompass accuracy, precision, recall, F1 score to measure its prediction quality and robustness.	<pre>from sklearn.ensemble import RandomForestClassifier from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score rf = RandomForestClassifier() rf.fit(X_train,y_train) y_pred_rf = rf.predict(X_test) print("Training Accuracy :", rf.score(X_train, y_train)) print("Testing Accuracy :", rf.score(X_test, y_test)) cm = confusion_matrix(y_test, y_pred_rf) plt.figure(figsize=(10, 10)) sns.heatmap(cm, annot=True, cmap = 'Blues', fmt = '.2g') plt.show() cr = classification_report(y_test, y_pred_rf) print(cr) false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, y_pred_rf) roc_auc = auc(false_positive_rate, true_positive_rate) print("ROC AUC",roc_auc) precision, recall, thresholds = precision_recall_curve(y_test, y_pred_rf) f1 = f1_score(y_test, y_pred_rf) precision_recall_auc = auc(recall, precision) print("Precision-Recall AUC",precision_recall_auc) # 64s Training Accuracy : 1.0 Testing Accuracy : 0.792813650180906</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>0.70</td><td>0.62</td><td>0.66</td><td>98</td></tr><tr><td>1</td><td>0.81</td><td>0.89</td><td>0.85</td><td>179</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.80</td><td>277</td></tr><tr><td>macro avg</td><td>0.70</td><td>0.76</td><td>0.77</td><td>277</td></tr><tr><td>weighted avg</td><td>0.79</td><td>0.80</td><td>0.79</td><td>277</td></tr></tbody></table>		precision	recall	f1-score	support	0	0.70	0.62	0.66	98	1	0.81	0.89	0.85	179	accuracy			0.80	277	macro avg	0.70	0.76	0.77	277	weighted avg	0.79	0.80	0.79	277
	precision	recall	f1-score	support																												
0	0.70	0.62	0.66	98																												
1	0.81	0.89	0.85	179																												
accuracy			0.80	277																												
macro avg	0.70	0.76	0.77	277																												
weighted avg	0.79	0.80	0.79	277																												

Model 4

XGB Classifier model typically include accuracy, precision, recall, F1 score to evaluate its prediction performance and generalization ability

```
from xgboost import XGBClassifier
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
xgb = XGBClassifier()
xgb.fit(x_train, y_train)
y_predicted_xgb = xgb.predict(x_test)
print("Training Accuracy :", xgb.score(x_train, y_train))
print("Testing Accuracy :", xgb.score(x_test, y_test))
cr = ClassificationReport(y_test, y_predicted_xgb)
print(cr)
print("\n")
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, y_predicted_xgb)
roc_auc = auc(false_positive_rate, true_positive_rate)
print("ROC Curves = ", roc_auc)
precision, recall, thresholds = precision_recall_curve(y_test, y_predicted_xgb)
f1 = f1_score(y_test, y_predicted_xgb)
Precision_Recall_xgb = auc(recall, precision)
print("Precision-Recall Curves = ", Precision_Recall_xgb)
```

✓ 1/6

Training Accuracy : 1.0
Testing Accuracy : 0.703429603888887

	precision	recall	f1-score	support
0	0.70	0.50	0.60	10
1	0.79	0.87	0.83	170
accuracy			0.77	177
macro avg	0.75	0.72	0.73	177
weighted avg	0.70	0.77	0.76	177

ROC Curves = 0.7237772203431609
Precision-Recall Curves = 0.8716001567500439