

Model Development Phase Template

Date	February 2026
Team ID	LTVIP2026TMIDS50820
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.

```
#Gathering accuracy score for each model
scores = {
    'Adaboost classifier': (
        '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 2 AdaBoost classifier model commonly include accuracy, precision, recall, F1 score which help assess the model's prediction accuracy and generalizability

```
from sklearn.ensemble import AdaBoostClassifier
strata
ada = AdaBoostClassifier()
ada.fit(X_train,y_train)
y_predict_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)
print('-----')
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 curves : ',roc_auc)
precision, recall, thresholds = precision_recall_curve(y_test, y_predicted_ab)
f1 = f1_score(y_test, y_predicted_ab)
Precision_Recall_Abs = auc(recall, precision)
print('Precision-Recall curves : ',Precision_Recall_Abs)
# Ab
training Accuracy : 0.8120123376013883
testing Accuracy : 0.77012485108554
precision recall f1-score support
0 0.77 0.68 0.66 98
1 0.69 0.87 0.83 179
accuracy 0.76 0.76 0.76 277
macro avg 0.76 0.76 0.74 277
weighted avg 0.77 0.76 0.77 277
```

Model 3 Random forest classifier model often encompass accuracy, precision, recall, F1 score to measure its prediction quality and robustness.

```
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))
tn, fp, fn, tp = confusion_matrix(y_test, y_pred_rf)
fnr = fp / tn
fnr
fnr * 100
fnr_percent = True_map['Ylabel'].mean() * 100
fnr_percent
mpf.show()
cr = classification_report(y_test, y_pred_rf)
print(cr)
print('-----')
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 curves : ',roc_auc)
precision, recall, thresholds = precision_recall_curve(y_test, y_pred_rf)
f1 = f1_score(y_test, y_pred_rf)
Precision_Recall_Abs = auc(recall, precision)
print('Precision-Recall curves : ',Precision_Recall_Abs)
# Ab
training Accuracy : 1.0
testing Accuracy : 0.797810502499
precision recall f1-score support
0 0.26 0.62 0.39 98
1 0.81 0.69 0.77 179
accuracy 0.79 0.79 0.77 277
macro avg 0.79 0.79 0.77 277
weighted avg 0.79 0.79 0.79 277
```

Model 4	<p>XGB Classifier model typically include accuracy, precision, recall, F1 score to evaluate its prediction performance and generalization ability</p>	<pre> from xgboost import XGBClassifier import numpy as np xgb = XGBClassifier() xgb.fit(X_train,y_train) y_pred=xgb.predict(X_test) print("Training Accuracy : ",xgb.score(X_train, y_train)) print("Testing Accuracy : ",xgb.score(X_test, y_test)) cr=classification_report(y_test, y_pred_xgb) print(cr) print("-----") roc_auc = AUC(false_positive_rate, true_positive_rate, thresholds = roc_curve.y_test,y_pred_xgb) print("ROC curve = ",roc_auc) gridk("ROC curves", "roc_auc") precision, recall, thresholds = precision_recall_curve(y_test, y_pred_xgb) f1 = f1_score(y_test, y_pred_xgb) Precision_Recall_xgb = auc(recall, precision) print("Precision-Recall curves = ",Precision_Recall_xgb) print("-----") </pre> <p>Training Accuracy : 1.0 Testing Accuracy : 0.70534294603888487</p> <table border="1"> <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>0.50</td> <td>0.60</td> <td>18</td> </tr> <tr> <td>1</td> <td>0.70</td> <td>0.87</td> <td>0.83</td> <td>179</td> </tr> </tbody> </table> <p>accuracy = 0.70 macro avg = 0.75 weighted avg = 0.70</p> <p>-----</p> <p>AUC curves = 0.7237772203623609 Precision-Recall curves = 0.8216081567386439</p>		precision	recall	f1-score	support	0	0.70	0.50	0.60	18	1	0.70	0.87	0.83	179
	precision	recall	f1-score	support													
0	0.70	0.50	0.60	18													
1	0.70	0.87	0.83	179													