# Model Development Phase Template

| | |
|---|---|
| Date | July 2024 |
| Team ID | team-739778 |
| 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))
```

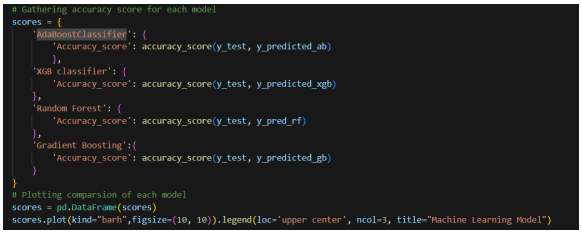
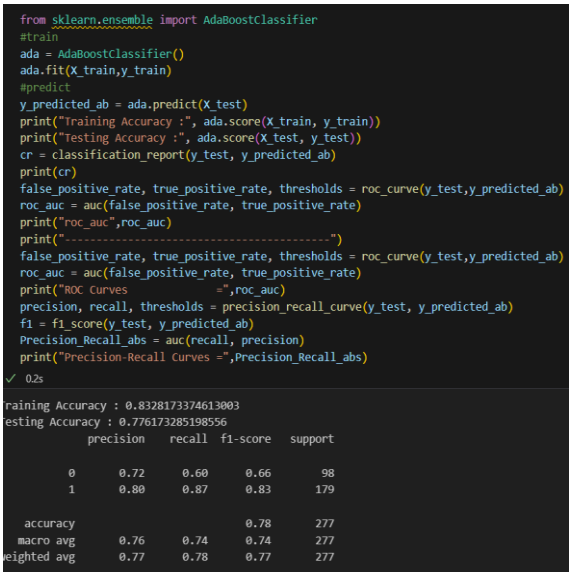
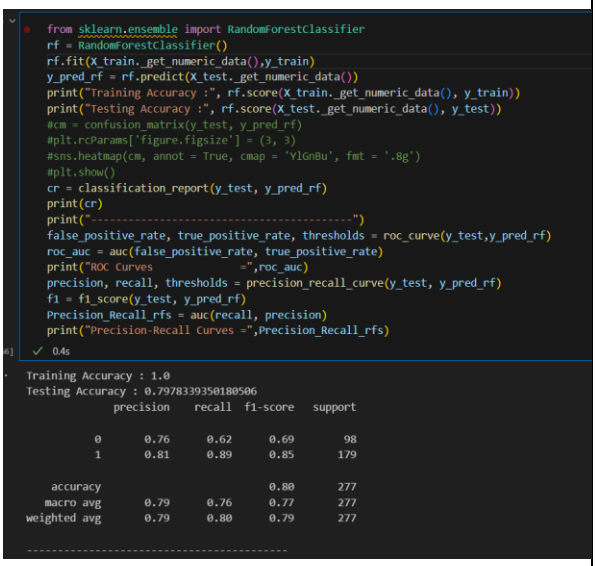
```
#importing and building the GradientBoostingClassifier model
from sklearn.ensemble import GradientBoostingClassifier
#train
gbc = GradientBoostingClassifier(learning_rate=0.02,
                                 max_depth=4,
                                 random_state=100, n_estimators=1000)
gbc.fit(X_train,y_train)
#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))
```

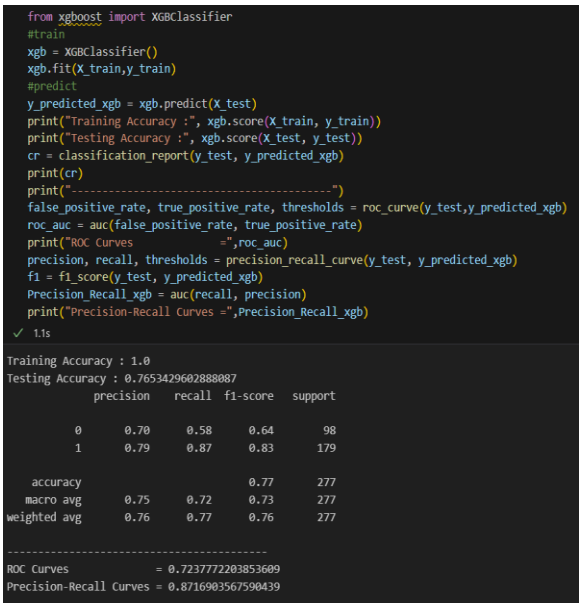
```
#importing and building the XGBClassifier model
from xgboost import XGBClassifier
#train
xgb = XGBClassifier()
xgb.fit(X_train,y_train)
#predict
y_predicted_xgb = xgb.predict(X_test)
print("Training Accuracy :", xgb.score(X_train, y_train))
print("Testing Accuracy :", xgb.score(X_test, y_test))
```

```
#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 comparsion 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. |  |
| Model 2 | AdaBoost classifier model commonly include accuracy, precision, recall, F1 score which help assess the model's prediction accuracy and generalizability |  |
| Model 3 | Random forest classifier model often encompass accuracy, precision, recall, F1 score to measure its prediction quality and robustness. |  |

| | | |
|---|---|---|
| Model 4 | XGB Classifier model typically include accuracy, precision, recall, F1 score to evaluate its prediction performance and generalization ability |  |

```
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))
cr = classification_report(y_test, y_predicted_xgb)
print(cr)
print("----------------------------------------")
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)
```

✓ 1.1s

```
Training Accuracy : 1.0
Testing Accuracy : 0.7653429602888087
              precision    recall  f1-score   support

           0       0.70      0.58      0.64        98
           1       0.79      0.87      0.83       179

    accuracy                           0.77       277
   macro avg       0.75      0.72      0.73       277
weighted avg       0.76      0.77      0.76       277

----------------------------------------
ROC Curves              = 0.7237772203853609
Precision-Recall Curves = 0.8716903567590439
```