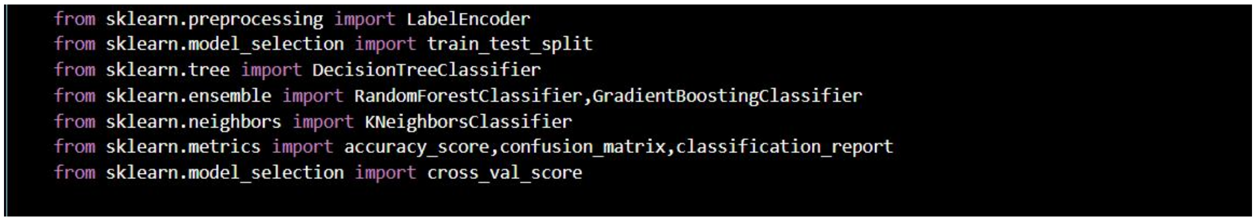
Model Development Phase Template

Date	9 July 2024
Team ID	SWTID1720104839
Project Title	Human Resource Management: Predicting Employee Promotions Using Machine Learning
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

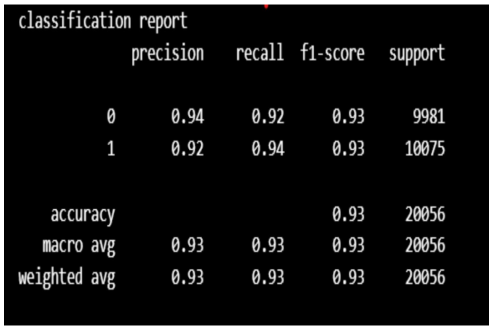
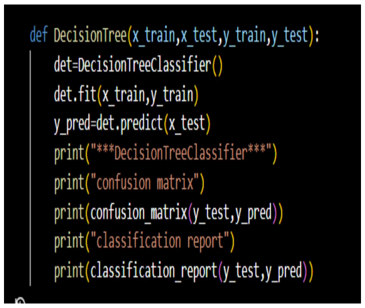
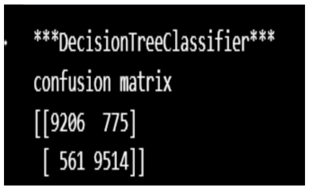
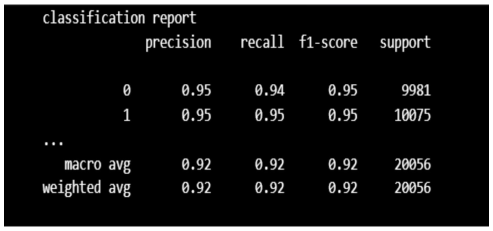
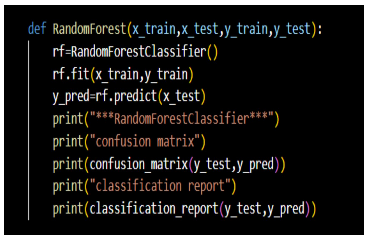
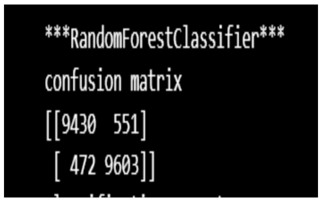
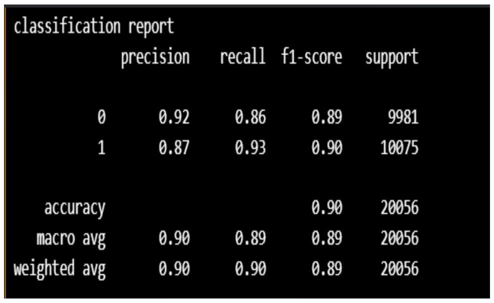
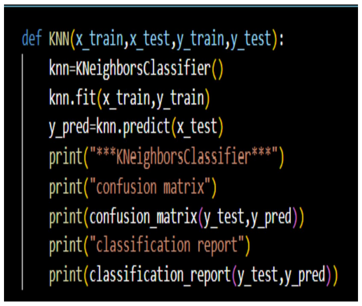
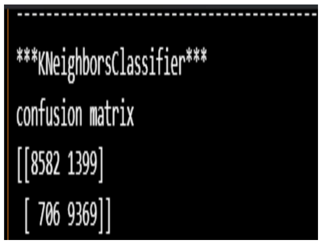
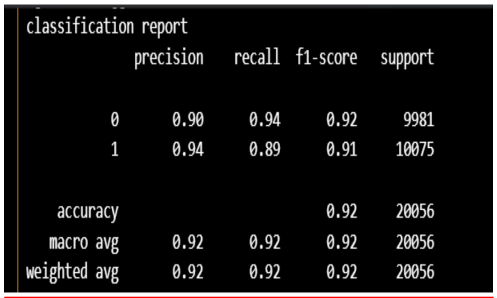
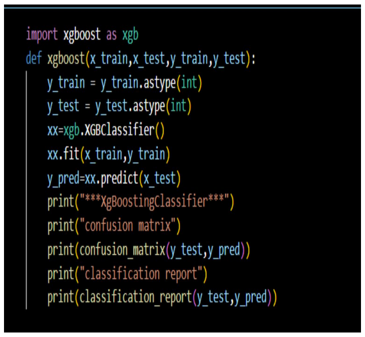
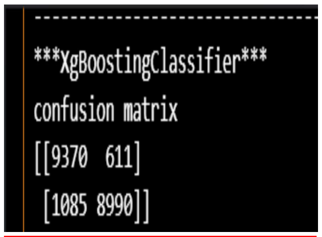
The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:



```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.model_selection import cross_val_score
```

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix
Decision Tree			
Random Forest			
KNN			
XG Boost			

Decision Tree – Classification Report:

```
classification report
              precision    recall  f1-score   support

           0       0.94      0.92      0.93      9981
           1       0.92      0.94      0.93     10075

    accuracy                           0.93     20056
   macro avg       0.93      0.93      0.93     20056
weighted avg       0.93      0.93      0.93     20056
```

Decision Tree – Accuracy:

```
def DecisionTree(x_train,x_test,y_train,y_test):
    det=DecisionTreeClassifier()
    det.fit(x_train,y_train)
    y_pred=det.predict(x_test)
    print("***DecisionTreeClassifier***")
    print("confusion matrix")
    print(confusion_matrix(y_test,y_pred))
    print("classification report")
    print(classification_report(y_test,y_pred))
```

Decision Tree – Confusion Matrix:

```
***DecisionTreeClassifier***
confusion matrix
[[9206  775]
 [ 561 9514]]
```

Random Forest – Classification Report:

```
classification report
              precision    recall  f1-score   support

           0       0.95      0.94      0.95      9981
           1       0.95      0.95      0.95     10075
...
   macro avg       0.92      0.92      0.92     20056
weighted avg       0.92      0.92      0.92     20056
```

Random Forest – Accuracy:

```
def RandomForest(x_train,x_test,y_train,y_test):
    rf=RandomForestClassifier()
    rf.fit(x_train,y_train)
    y_pred=rf.predict(x_test)
    print("***RandomForestClassifier***")
    print("confusion matrix")
    print(confusion_matrix(y_test,y_pred))
    print("classification report")
    print(classification_report(y_test,y_pred))
```

Random Forest – Confusion Matrix:

```
***RandomForestClassifier***
confusion matrix
[[9430  551]
 [ 472 9603]]
```

KNN – Classification Report:

```
classification report
              precision    recall  f1-score   support

           0       0.92      0.86      0.89      9981
           1       0.87      0.93      0.90     10075

    accuracy                           0.90     20056
   macro avg       0.90      0.89      0.89     20056
weighted avg       0.90      0.90      0.89     20056
```

KNN – Accuracy:

```
def KNN(x_train,x_test,y_train,y_test):
    knn=KNeighborsClassifier()
    knn.fit(x_train,y_train)
    y_pred=knn.predict(x_test)
    print("***KNeighborsClassifier***")
    print("confusion matrix")
    print(confusion_matrix(y_test,y_pred))
    print("classification report")
    print(classification_report(y_test,y_pred))
```

KNN – Confusion Matrix:

```
***KNeighborsClassifier***
confusion matrix
[[8582 1399]
 [ 706 9369]]
```

XG Boost – Classification Report:

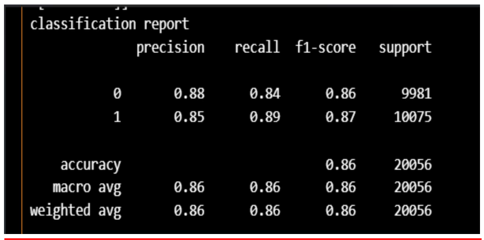
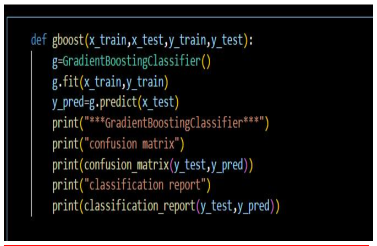
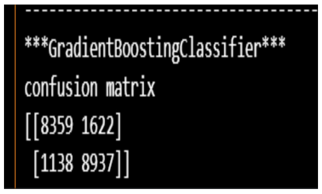
```
classification report
              precision    recall  f1-score   support

           0       0.90      0.94      0.92      9981
           1       0.94      0.89      0.91     10075

    accuracy                           0.92     20056
   macro avg       0.92      0.92      0.92     20056
weighted avg       0.92      0.92      0.92     20056
```

XG Boost – Accuracy:

```
import xgboost as xgb
def xgboost(x_train,x_test,y_train,y_test):
    y_train = y_train.astype(int)
    y_test = y_test.astype(int)
    xx=xgb.XGBClassifier()
    xx.fit(x_train,y_train)
    y_pred=xx.predict(x_test)
    print("***XgBoostingClassifier***")
    print("confusion matrix")
    print(confusion_matrix(y_test,y_pred))
    print("classification report")
    print(classification_report(y_test,y_pred))
```

XG Boost – Confusion Matrix:

```
***XgBoostingClassifier***
confusion matrix
[[9370  611]
 [1085 8990]]
```

Gradient Boosting			

```
classification report
              precision    recall  f1-score   support

           0       0.88      0.84      0.86      9981
           1       0.85      0.89      0.87     10075

    accuracy                           0.86     20056
   macro avg       0.86      0.86      0.86     20056
weighted avg       0.86      0.86      0.86     20056
```

```
def gboost(x_train,x_test,y_train,y_test):
    g=GradientBoostingClassifier()
    g.fit(x_train,y_train)
    y_pred=g.predict(x_test)
    print("***GradientBoostingClassifier***")
    print("confusion matrix")
    print(confusion_matrix(y_test,y_pred))
    print("classification report")
    print(classification_report(y_test,y_pred))
```

```
-----------------------------------------
***GradientBoostingClassifier***
confusion matrix
[[8359 1622]
 [1138 8937]]
```