Model Development Phase Template

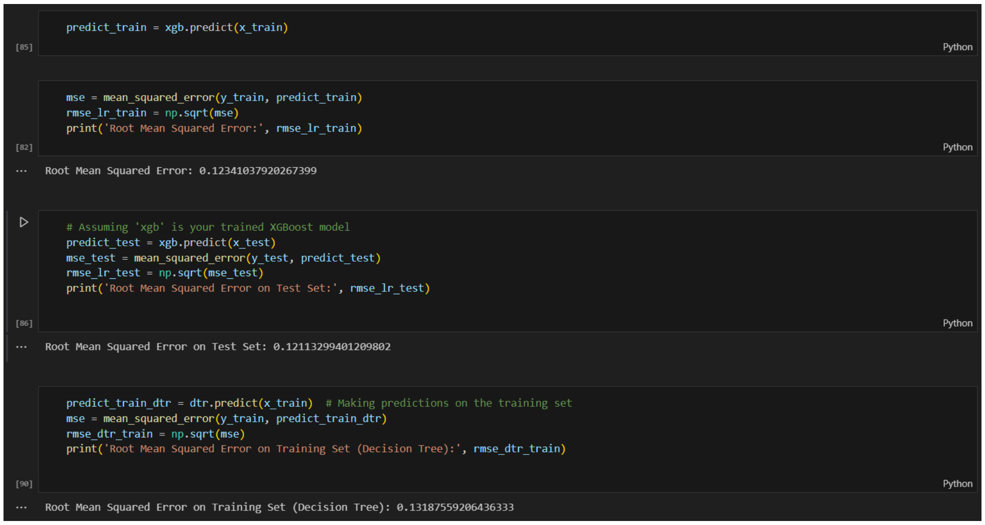
Date	5th July 2024
Team ID	SWTID1720673861
Project Title	Garment Worker Efficiency Calculator
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

1)Bagging Regressor Model

This code creates a machine learning model using the XGBoost algorithm. The model is designed to predict a numeric value (the target variable) based on a set of input variables.
To improve the accuracy of the model, the Bagging technique is used. This involves creating multiple versions of the model, each with slightly different training data, and combining their predictions to create a final prediction.
Finally, the model is trained using a set of input data (X_train) and the corresponding target values (y_train). This involves adjusting the weights of the various components of the model until it can accurately predict the target values based on the input data.

2)Boosting Regressor Model
This code creates a machine learning model to predict a numeric value based on a set of input variables using the XGBoost algorithm as the base model.
To improve the accuracy of the model, the AdaBoost technique is used. AdaBoost stands for Adaptive Boosting and works by creating multiple versions of the model, each with slightly different training data, and weighting the predictions of each model based on its accuracy.
Finally, the model is trained using a set of input data (X_train) and the corresponding target values (y_train). This involves adjusting the weights of the various components of the model until it can accurately predict the target values based on the input data.



```
predict_train = xgb.predict(x_train)
```

```
mse = mean_squared_error(y_train, predict_train)
rmse_lr_train = np.sqrt(mse)
print('Root Mean Squared Error:', rmse_lr_train)
```

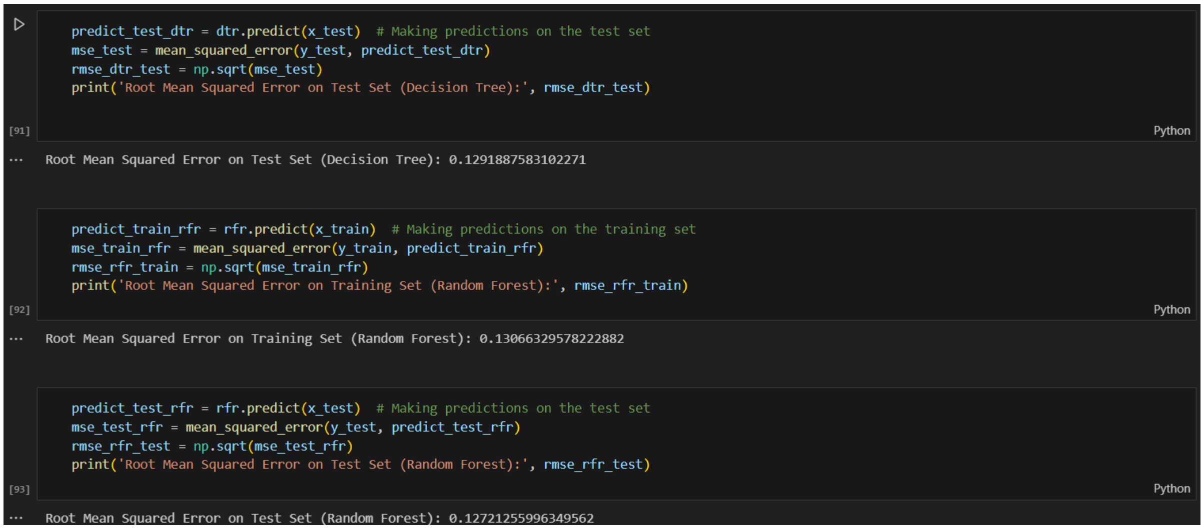
```
Root Mean Squared Error: 0.12341037920267399
```

```
# Assuming 'xgb' is your trained XGBoost model
predict_test = xgb.predict(x_test)
mse_test = mean_squared_error(y_test, predict_test)
rmse_lr_test = np.sqrt(mse_test)
print('Root Mean Squared Error on Test Set:', rmse_lr_test)
```

```
Root Mean Squared Error on Test Set: 0.12113299401209802
```

```
predict_train_dtr = dtr.predict(x_train)  # Making predictions on the training set
mse = mean_squared_error(y_train, predict_train_dtr)
rmse_dtr_train = np.sqrt(mse)
print('Root Mean Squared Error on Training Set (Decision Tree):', rmse_dtr_train)
```

```
Root Mean Squared Error on Training Set (Decision Tree): 0.13187559206436333
```



```
predict_test_dtr = dtr.predict(x_test)  # Making predictions on the test set
mse_test = mean_squared_error(y_test, predict_test_dtr)
rmse_dtr_test = np.sqrt(mse_test)
print('Root Mean Squared Error on Test Set (Decision Tree):', rmse_dtr_test)
```

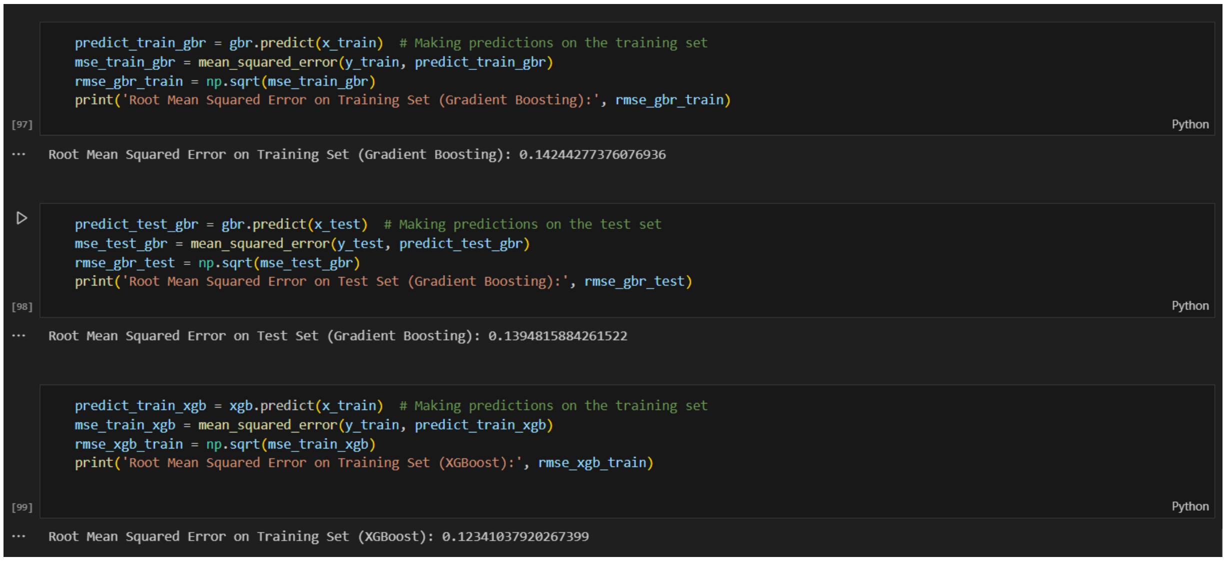
```
Root Mean Squared Error on Test Set (Decision Tree): 0.1291887583102271
```

```
predict_train_rfr = rfr.predict(x_train)  # Making predictions on the training set
mse_train_rfr = mean_squared_error(y_train, predict_train_rfr)
rmse_rfr_train = np.sqrt(mse_train_rfr)
print('Root Mean Squared Error on Training Set (Random Forest):', rmse_rfr_train)
```

```
Root Mean Squared Error on Training Set (Random Forest): 0.13066329578222882
```

```
predict_test_rfr = rfr.predict(x_test)  # Making predictions on the test set
mse_test_rfr = mean_squared_error(y_test, predict_test_rfr)
rmse_rfr_test = np.sqrt(mse_test_rfr)
print('Root Mean Squared Error on Test Set (Random Forest):', rmse_rfr_test)
```

```
Root Mean Squared Error on Test Set (Random Forest): 0.12721255996349562
```



```
predict_train_gbr = gbr.predict(x_train)  # Making predictions on the training set
mse_train_gbr = mean_squared_error(y_train, predict_train_gbr)
rmse_gbr_train = np.sqrt(mse_train_gbr)
print('Root Mean Squared Error on Training Set (Gradient Boosting):', rmse_gbr_train)
```

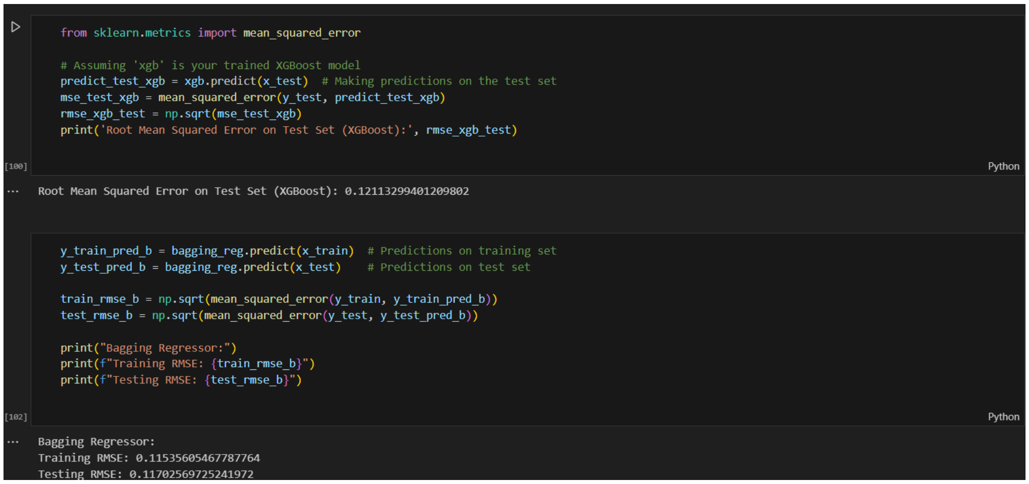
```
Root Mean Squared Error on Training Set (Gradient Boosting): 0.14244277376076936
```

```
predict_test_gbr = gbr.predict(x_test)  # Making predictions on the test set
mse_test_gbr = mean_squared_error(y_test, predict_test_gbr)
rmse_gbr_test = np.sqrt(mse_test_gbr)
print('Root Mean Squared Error on Test Set (Gradient Boosting):', rmse_gbr_test)
```

```
Root Mean Squared Error on Test Set (Gradient Boosting): 0.1394815884261522
```

```
predict_train_xgb = xgb.predict(x_train)  # Making predictions on the training set
mse_train_xgb = mean_squared_error(y_train, predict_train_xgb)
rmse_xgb_train = np.sqrt(mse_train_xgb)
print('Root Mean Squared Error on Training Set (XGBoost):', rmse_xgb_train)
```

```
Root Mean Squared Error on Training Set (XGBoost): 0.12341037920267399
```



```python
from sklearn.metrics import mean_squared_error

# Assuming 'xgb' is your trained XGBoost model
predict_test_xgb = xgb.predict(x_test)  # Making predictions on the test set
mse_test_xgb = mean_squared_error(y_test, predict_test_xgb)
rmse_xgb_test = np.sqrt(mse_test_xgb)
print('Root Mean Squared Error on Test Set (XGBoost):', rmse_xgb_test)
```

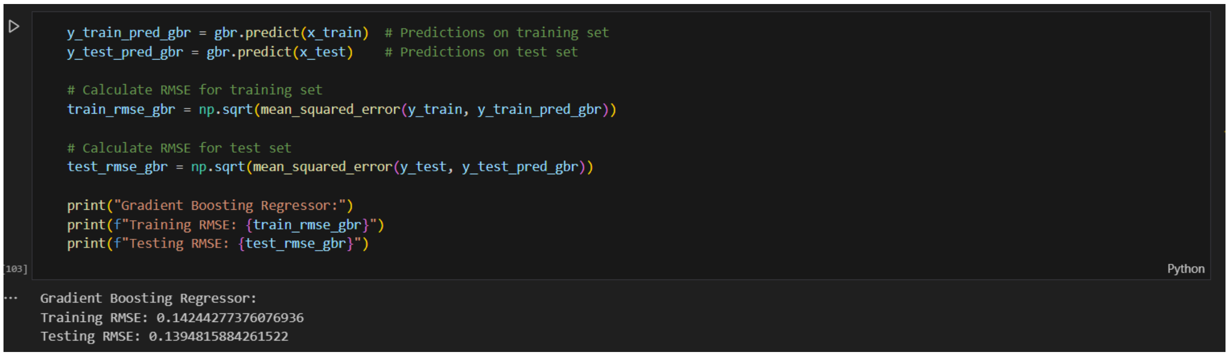
```
Root Mean Squared Error on Test Set (XGBoost): 0.12113299401209802
```

```python
y_train_pred_b = bagging_reg.predict(x_train)  # Predictions on training set
y_test_pred_b = bagging_reg.predict(x_test)    # Predictions on test set

train_rmse_b = np.sqrt(mean_squared_error(y_train, y_train_pred_b))
test_rmse_b = np.sqrt(mean_squared_error(y_test, y_test_pred_b))

print("Bagging Regressor:")
print(f"Training RMSE: {train_rmse_b}")
print(f"Testing RMSE: {test_rmse_b}")
```

```
Bagging Regressor:
Training RMSE: 0.11535605467787764
Testing RMSE: 0.11702569725241972
```



```python
y_train_pred_gbr = gbr.predict(x_train)  # Predictions on training set
y_test_pred_gbr = gbr.predict(x_test)    # Predictions on test set

# Calculate RMSE for training set
train_rmse_gbr = np.sqrt(mean_squared_error(y_train, y_train_pred_gbr))

# Calculate RMSE for test set
test_rmse_gbr = np.sqrt(mean_squared_error(y_test, y_test_pred_gbr))

print("Gradient Boosting Regressor:")
print(f"Training RMSE: {train_rmse_gbr}")
print(f"Testing RMSE: {test_rmse_gbr}")
```

```
Gradient Boosting Regressor:
Training RMSE: 0.14244277376076936
Testing RMSE: 0.1394815884261522
```