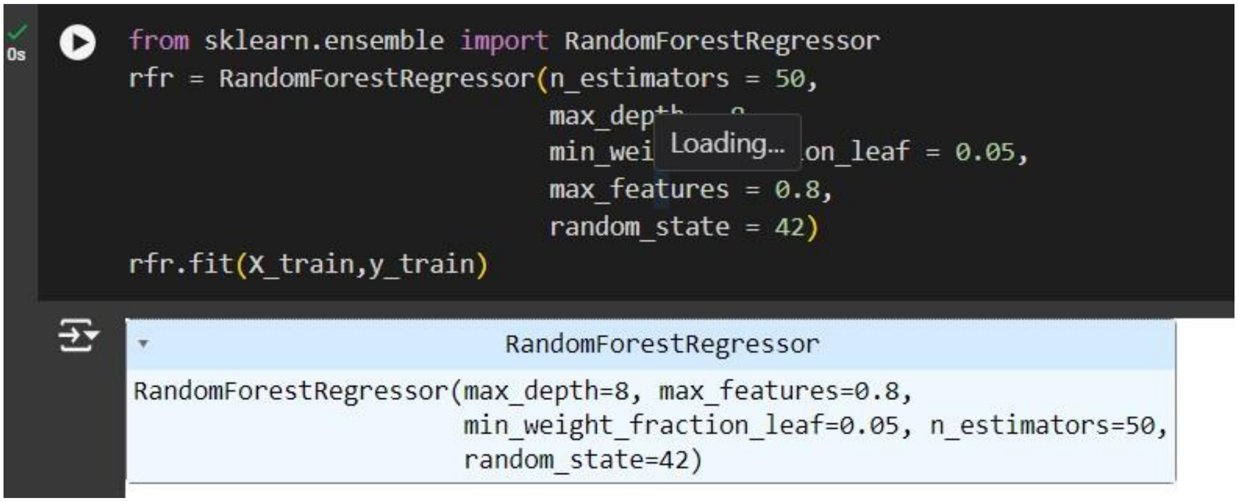
# Model Development Phase Template

| | |
|---|---|
| Date | 15 July 2024 |
| Team ID | 739921 |
| Project Title | Smartwatch Price Prediction |
| Maximum Marks | 4 Marks |

## Initial Model Training Code, Model Validation and Evaluation Report

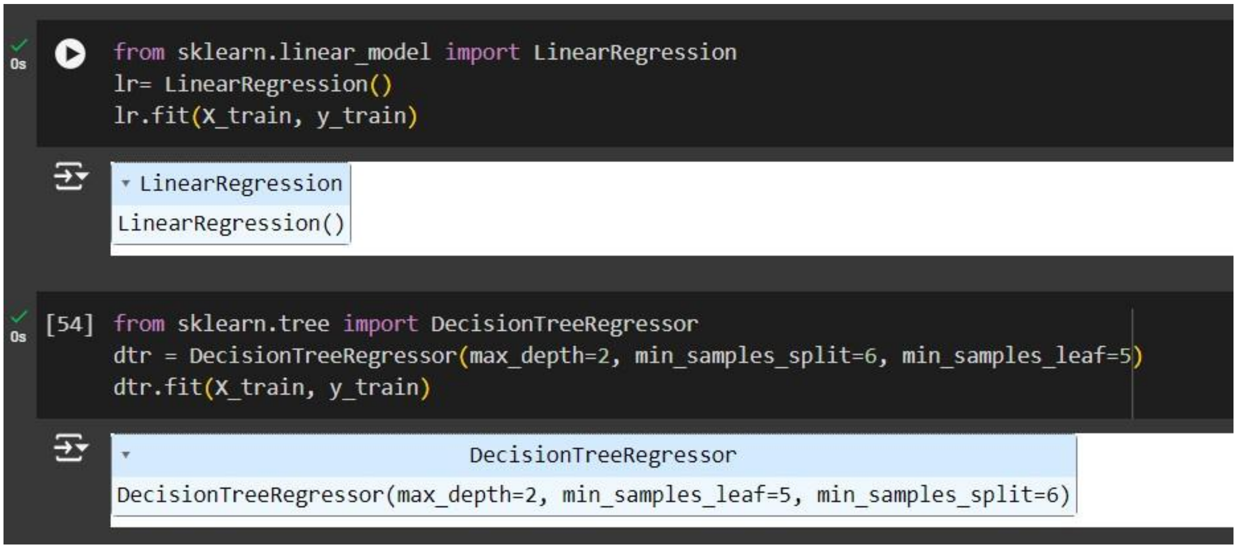
The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

## Initial Model Training Code:



```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor(n_estimators = 50,
                            max_dep
                            min_wei Loading... on_leaf = 0.05,
                            max_features = 0.8,
                            random_state = 42)
rfr.fit(X_train,y_train)
```

```
▾                    RandomForestRegressor
RandomForestRegressor(max_depth=8, max_features=0.8,
                      min_weight_fraction_leaf=0.05, n_estimators=50,
                      random_state=42)
```

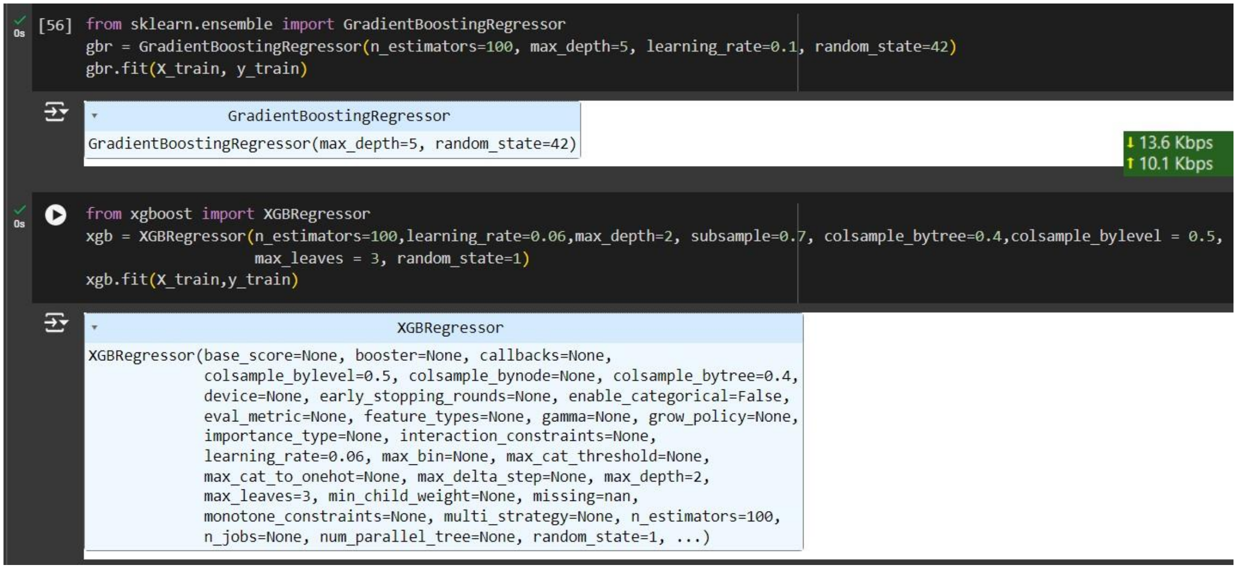


```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(X_train, y_train)
```

```
LinearRegression
LinearRegression()
```

```
[54] from sklearn.tree import DecisionTreeRegressor
dtr = DecisionTreeRegressor(max_depth=2, min_samples_split=6, min_samples_leaf=5)
dtr.fit(X_train, y_train)
```

```
DecisionTreeRegressor
DecisionTreeRegressor(max_depth=2, min_samples_leaf=5, min_samples_split=6)
```



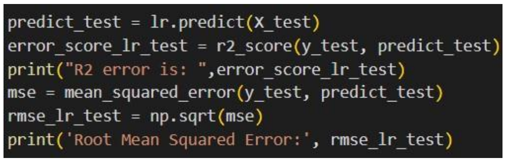
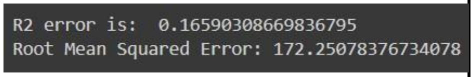
```
[56] from sklearn.ensemble import GradientBoostingRegressor
gbr = GradientBoostingRegressor(n_estimators=100, max_depth=5, learning_rate=0.1, random_state=42)
gbr.fit(X_train, y_train)
```

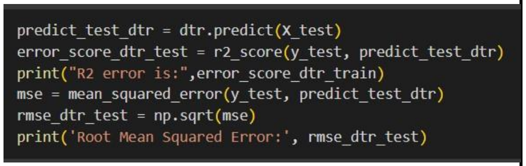
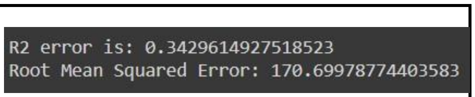
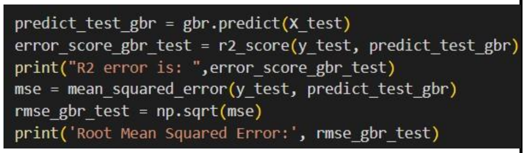
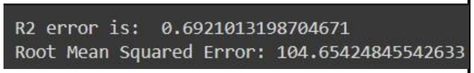
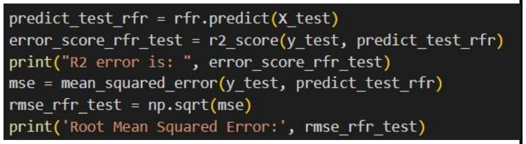
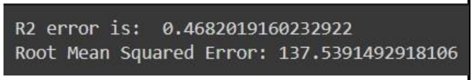
```
GradientBoostingRegressor
GradientBoostingRegressor(max_depth=5, random_state=42)
```

```
from xgboost import XGBRegressor
xgb = XGBRegressor(n_estimators=100,learning_rate=0.06,max_depth=2, subsample=0.7, colsample_bytree=0.4,colsample_bylevel = 0.5,
                   max_leaves = 3, random_state=1)
xgb.fit(X_train,y_train)
```

```
XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=0.5, colsample_bynode=None, colsample_bytree=0.4,
             device=None, early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, feature_types=None, gamma=None, grow_policy=None,
             importance_type=None, interaction_constraints=None,
             learning_rate=0.06, max_bin=None, max_cat_threshold=None,
             max_cat_to_onehot=None, max_delta_step=None, max_depth=2,
             max_leaves=3, min_child_weight=None, missing=nan,
             monotone_constraints=None, multi_strategy=None, n_estimators=100,
             n_jobs=None, num_parallel_tree=None, random_state=1, ...)
```

# Model Validation and Evaluation Report:

| Model | Classification Report | Score | RMS Values |
|---|---|---|---|
| Linear Regression | <br>`predict_test = lr.predict(X_test)`<br>`error_score_lr_test = r2_score(y_test, predict_test)`<br>`print("R2 error is: ",error_score_lr_test)`<br>`mse = mean_squared_error(y_test, predict_test)`<br>`rmse_lr_test = np.sqrt(mse)`<br>`print('Root Mean Squared Error:', rmse_lr_test)` | 0.85 | <br>`R2 error is:  0.16590308669836795`<br>`Root Mean Squared Error: 172.25078376734078` |

| | | | |
|---|---|---|---|
| **Decision Tree** | <br>predict_test_dtr = dtr.predict(X_test)<br>error_score_dtr_test = r2_score(y_test, predict_test_dtr)<br>print("R2 error is:",error_score_dtr_train)<br>mse = mean_squared_error(y_test, predict_test_dtr)<br>rmse_dtr_test = np.sqrt(mse)<br>print('Root Mean Squared Error:', rmse_dtr_test) | 0.90 | <br>R2 error is: 0.3429614927518523<br>Root Mean Squared Error: 170.69978774403583 |
| **Gradient Boosting** | <br>predict_test_gbr = gbr.predict(X_test)<br>error_score_gbr_test = r2_score(y_test, predict_test_gbr)<br>print("R2 error is: ",error_score_gbr_test)<br>mse = mean_squared_error(y_test, predict_test_gbr)<br>rmse_gbr_test = np.sqrt(mse)<br>print('Root Mean Squared Error:', rmse_gbr_test) | 0.89 | <br>R2 error is:  0.6921013198704671<br>Root Mean Squared Error: 104.65424845542633 |
| Random Forest Tree | <br>predict_test_rfr = rfr.predict(X_test)<br>error_score_rfr_test = r2_score(y_test, predict_test_rfr)<br>print("R2 error is: ", error_score_rfr_test)<br>mse = mean_squared_error(y_test, predict_test_rfr)<br>rmse_rfr_test = np.sqrt(mse)<br>print('Root Mean Squared Error:', rmse_rfr_test) | 0.92 | <br>R2 error is:  0.4682019160232922<br>Root Mean Squared Error: 137.5391492918106 |