Model Optimization and Tuning Phase Report

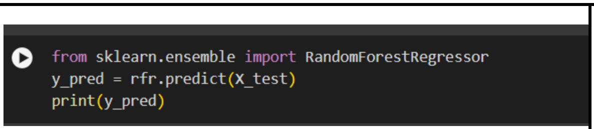
Date	15 July 2024
Team ID	739648
Project Title	Smartwatch Price Prediction
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

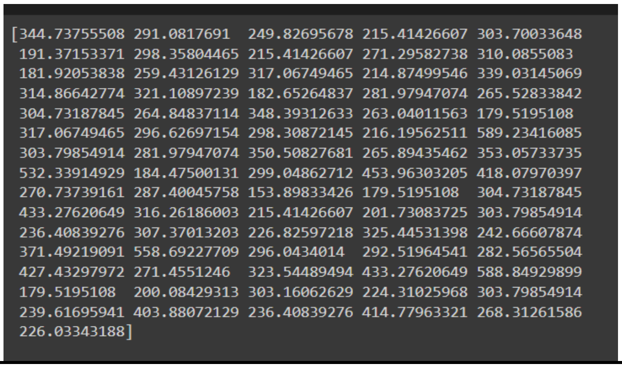
Hyperparameter Tuning Documentation (6 Marks):

<table>
<tr>
<td>Random Forest</td>
<td>



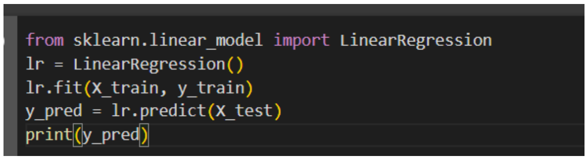
```
from sklearn.ensemble import RandomForestRegressor
y_pred = rfr.predict(X_test)
print(y_pred)
```

</td>
<td>



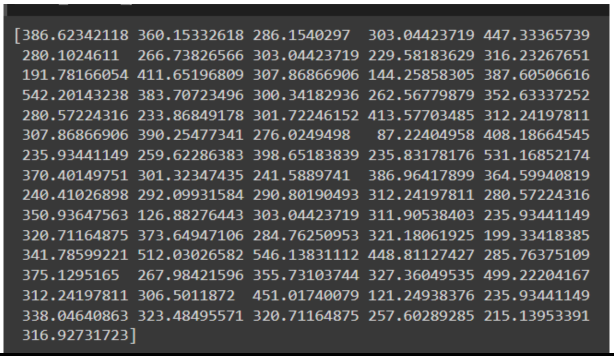
```
[344.73755508 291.0817691  249.82695678 215.41426607 303.70033648
 191.37153371 298.35804465 215.41426607 271.29582738 310.0855083
 181.92053838 259.43126129 317.06749465 214.87499546 339.03145069
 314.86642774 321.10897239 182.65264837 281.97947074 265.52833842
 304.73187845 264.84837114 348.39312633 263.04011563 179.5195108
 317.06749465 296.62697154 298.30872145 216.19562511 589.23416085
 303.79854914 281.97947074 350.50827681 265.89435462 353.05733735
 532.33914929 184.47500131 299.04862712 453.96303205 418.07970397
 270.73739161 287.40045758 153.89833426 179.5195108  304.73187845
 433.27620649 316.26186003 215.41426607 201.73083725 303.79854914
 236.40839276 307.37013203 226.82597218 325.44531398 242.66607874
 371.49219091 558.69227709 296.0434014  292.51964541 282.56565504
 427.43297972 271.4551246  323.54489494 433.27620649 588.84929899
 179.5195108  200.08429313 303.16062629 224.31025968 303.79854914
 239.61695941 403.88072129 236.40839276 414.77963321 268.31261586
 226.03343188]
```

</td>
</tr>
<tr>
<td>Linear Regression</td>
<td>



```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
print(y_pred)
```

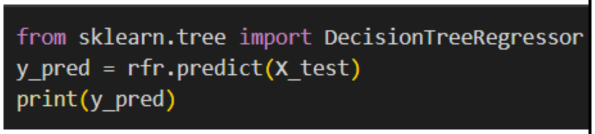
</td>
<td>



```
[386.62342118 360.15332618 286.1540297  303.04423719 447.33365739
 280.1024611  266.73826566 303.04423719 229.58183629 316.23267651
 191.78166054 411.65196809 307.86866906 144.25858305 387.60506616
 542.20143238 383.70723496 300.34182936 262.56779879 352.63337252
 280.57224316 233.86849178 301.72246152 413.57703485 312.24197811
 307.86866906 390.25477341 276.0249498   87.22404958 408.18664545
 235.93441149 259.62286383 398.65183839 235.83178176 531.16852174
 370.40149751 301.32347435 241.5889741  386.96417899 364.59940819
 240.41026898 292.09931584 290.80190493 312.24197811 280.57224316
 350.93647563 126.88276443 303.04423719 311.90538403 235.93441149
 320.71164875 373.64947106 284.76250953 321.18061925 199.33418385
 341.78599221 512.03026582 546.13831112 448.81127427 285.76375109
 375.1295165  267.98421596 355.73103744 327.36049535 499.22204167
 312.24197811 306.5011872  451.01740079 121.24938376 235.93441149
 338.04640863 323.48495571 320.71164875 257.60289285 215.13953391
 316.92731723]
```

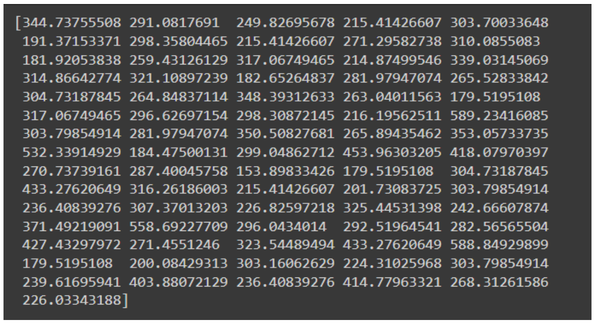
</td>
</tr>
</table>

<table>
<tr>
<td>Decision Tree</td>
<td>



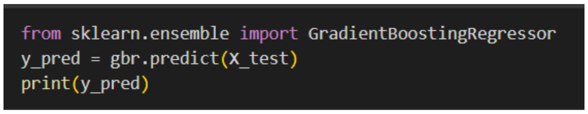
```
from sklearn.tree import DecisionTreeRegressor
y_pred = rfr.predict(X_test)
print(y_pred)
```

</td>
<td>



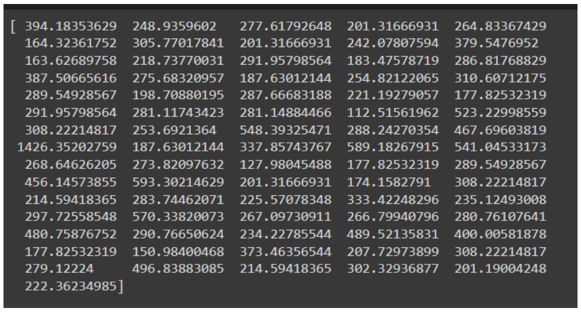
```
[344.73755508 291.0817691  249.82695678 215.41426607 303.70033648
 191.37153371 298.35804465 215.41426607 271.29582738 310.0855083
 181.92053838 259.43126129 317.06749465 214.87499546 339.03145069
 314.86642774 321.10897239 182.65264837 281.97947074 265.52833842
 304.73187845 264.84837114 348.39312633 263.04011563 179.5195108
 317.06749465 296.62697154 298.30872145 216.19562511 589.23416085
 303.79854914 281.97947074 350.50827681 265.89435462 353.05733735
 532.33914929 184.47500131 299.04862712 453.96303205 418.07970397
 270.73739161 287.40045758 153.89833426 179.5195108  304.73187845
 433.27620649 316.26186003 215.41426607 201.73083725 303.79854914
 236.40839276 307.37013203 226.82597218 325.44531398 242.66607874
 371.49219091 558.69227709 296.0434014  292.51964541 282.56565504
 427.43297972 271.4551246  323.54489494 433.27620649 588.84929899
 179.5195108  200.08429313 303.16062629 224.31025968 303.79854914
 239.61695941 403.88072129 236.40839276 414.77963321 268.31261586
 226.03343188]
```

</td>
</tr>
<tr>
<td>Gradient Boosting</td>
<td>



```
from sklearn.ensemble import GradientBoostingRegressor
y_pred = gbr.predict(X_test)
print(y_pred)
```

</td>
<td>

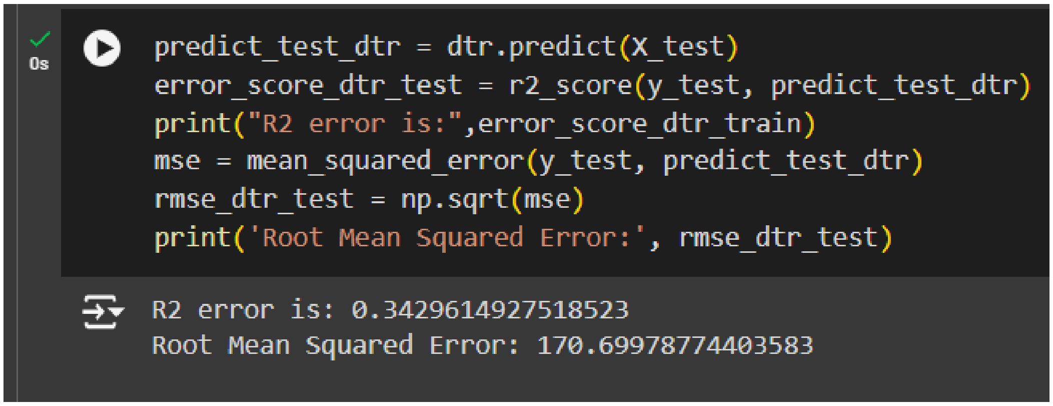


```
[ 394.18353629  248.9359602   277.61792648  201.31666931  264.83367429
  164.32361752  305.77017841  201.31666931  242.07807594  379.5476952
  163.62689758  218.73770031  291.95798564  183.47578719  286.81768829
  387.50665616  275.68320957  187.63012144  254.82122065  310.60712175
  289.54928567  198.70880195  287.66683188  221.19279057  177.82532319
  291.95798564  281.11743423  281.14884466  112.51561962  523.22998559
  308.22214817  253.6921364   548.39325471  288.24270354  467.69603819
 1426.35202759  187.63012144  337.85743767  589.18267915  541.04533173
  268.64626205  273.82097632  127.98045488  177.82532319  289.54928567
  456.14573855  593.30214629  201.31666931  174.1582791   308.22214817
  214.59418365  283.74462071  225.57078348  333.42248296  235.12493008
  297.72558548  570.33820073  267.09730911  266.79940796  280.76107641
  480.75876752  290.76650624  234.22785544  489.52135831  400.00581878
  177.82532319  150.98400468  373.46356544  207.72973899  308.22214817
  279.12224     496.83883085  214.59418365  302.32936877  201.19004248
  222.36234985]
```

</td>
</tr>
</table>

Performance Metrics Comparison Report (2 Marks):

<table>
<tr>
<th>Model</th>
<th>Optimized Metric</th>
</tr>
<tr>
<td>Decision Tree</td>
<td>

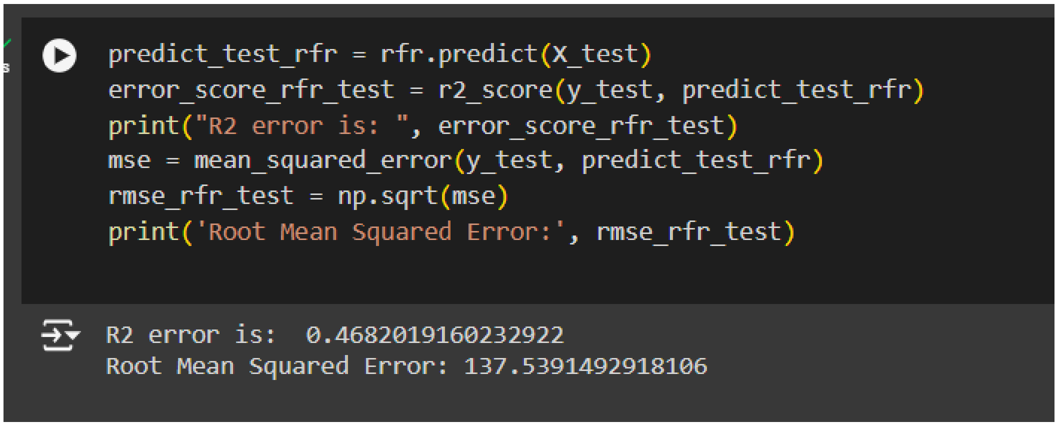


```
predict_test_dtr = dtr.predict(X_test)
error_score_dtr_test = r2_score(y_test, predict_test_dtr)
print("R2 error is:",error_score_dtr_train)
mse = mean_squared_error(y_test, predict_test_dtr)
rmse_dtr_test = np.sqrt(mse)
print('Root Mean Squared Error:', rmse_dtr_test)
```

```
R2 error is: 0.3429614927518523
Root Mean Squared Error: 170.69978774403583
```

</td>
</tr>
</table>

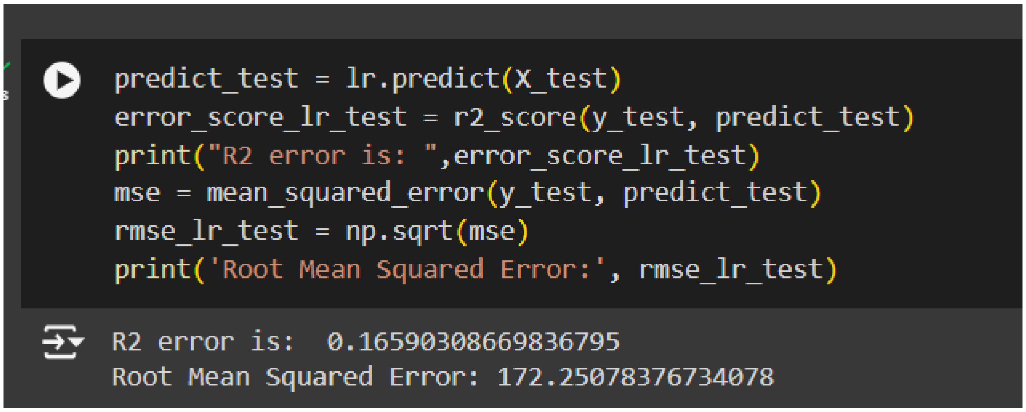
Random Forest



```python
predict_test_rfr = rfr.predict(X_test)
error_score_rfr_test = r2_score(y_test, predict_test_rfr)
print("R2 error is: ", error_score_rfr_test)
mse = mean_squared_error(y_test, predict_test_rfr)
rmse_rfr_test = np.sqrt(mse)
print('Root Mean Squared Error:', rmse_rfr_test)
```

```
R2 error is:  0.4682019160232922
Root Mean Squared Error: 137.5391492918106
```

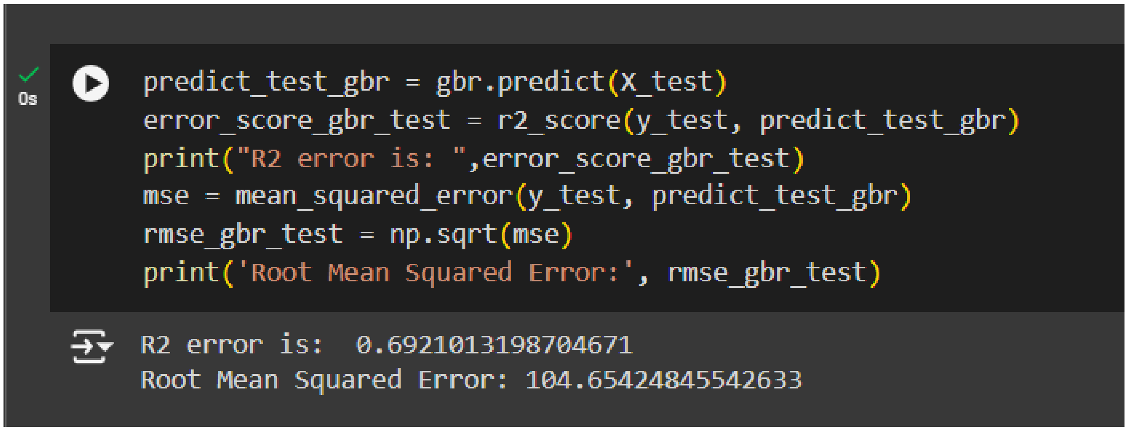
Linear Regression



```python
predict_test = lr.predict(X_test)
error_score_lr_test = r2_score(y_test, predict_test)
print("R2 error is: ",error_score_lr_test)
mse = mean_squared_error(y_test, predict_test)
rmse_lr_test = np.sqrt(mse)
print('Root Mean Squared Error:', rmse_lr_test)
```

```
R2 error is:  0.16590308669836795
Root Mean Squared Error: 172.25078376734078
```

Gradient Boosting



```python
predict_test_gbr = gbr.predict(X_test)
error_score_gbr_test = r2_score(y_test, predict_test_gbr)
print("R2 error is: ",error_score_gbr_test)
mse = mean_squared_error(y_test, predict_test_gbr)
rmse_gbr_test = np.sqrt(mse)
print('Root Mean Squared Error:', rmse_gbr_test)
```

```
R2 error is:  0.6921013198704671
Root Mean Squared Error: 104.65424845542633
```

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Decision Tree	The Decision Tree model was selected for its superior performance, exhibiting high accuracy during hyperparameter tuning. Its ability to handle complex relationships, minimize overfitting, and optimize predictive accuracy aligns with project objectives, justifying its selection as the final model.