

Model Development Phase Template

Date	10 July 2024
Team ID	740064
Project Title	Trip-Based Modelling of Fuel Consumption in Modern Fleet Vehicles 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:

Paste the screenshot of the model training code

```

✓ [18] from sklearn.model_selection import train_test_split
0s      from sklearn.linear_model import LinearRegression

#Splitting Data Into Train And Test

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=42)

```


Linear regression

```

✓ [25] #Linear Regression
0s

linReg = LinearRegression()
linReg.fit(x_train,y_train)


```



▾ LinearRegression
 LinearRegression()

Lasso Regression

✓ [39] #Lasso Regression model


✓ 0s  `lassoReg = linear_model.Lasso(alpha = 0.1)`
`lassoReg.fit(x,y)`



▼ Lasso
Lasso(alpha=0.1)

Decision Tree

✓ [47] #Decision Tree Model:

✓ 0s  `dt = DecisionTreeRegressor(random_state = 0)`
`dt.fit(x,y)`



▼ DecisionTreeRegressor
DecisionTreeRegressor(random_state=0)

Random Forest

✓ [51] #Random Forest Model:

✓ 0s `rf = RandomForestRegressor(n_estimators = 100 , random_state = 0)`
`rf.fit(x,y)`



▼ RandomForestRegressor
RandomForestRegressor(random_state=0)

Model Validation and Evaluation Report:

Model	Training Report	Accuracy	Metrix
Linear Regression	<pre>[15] LinearRegression LinearRegression() [48] x_train.shape (771, 2) y_pred = linreg.predict(x_test) print(y_pred) [4.74529819 5.28891213 5.11845572 5.18647084 4.56157089 5.94253804 5.67791211 5.1873266 5.9187236 4.89998451 4.11401841 4.81810568 6.55844017 4.54298256 5.12075484 5.27925582 5.56127759 5.1741241 5.9806296 5.18924689 4.15432091 5.27454886 4.88292477 5.18919997 4.91854932 4.81581281 4.81518828 4.23426984 5.19745057 5.90768814 4.93773837 5.1268321 4.6972807 4.8628162 5.50925144 4.93937775 4.88951514 4.89178814 5.11518485 4.89718977 4.41081181 5.24454536 5.9578805 4.4757374 4.65456117 4.10882762 5.0415661 5.21180179 4.98028461 4.9576388 4.82470462 5.40886892 5.40516957 5.22776227 4.60757862 4.56119185 4.78635171 5.8877556 4.6714596 4.8111791 5.65291181 4.87535816 5.09023912 4.75794574 4.7078117 4.71136158 5.5895258 4.37456262 4.84902389 4.87652148 4.21228735 4.79226084 5.25855747 4.19811818 5.13881151 4.85766627 5.15876861 5.80861615 5.2171176 5.11778705 5.3851894 5.48841181 4.5127386 5.38705851 5.28727892 4.46588844 5.25282825 5.38283086 5.11834712 4.41888829 4.77059454 4.5419151 5.13884959 4.81741587 4.54151162 5.45766629 5.89313983 4.88851457 5.11845122 5.10015732 5.02870111 4.81907371 4.52598833 4.44424088 5.25788948 4.52896624 4.37401409 4.57485328 4.51893819 4.81681875 4.73093957 5.17896887 4.15471976 4.87538886 4.77326158 5.41212489 4.88509781]</pre>	0.11	<pre>[58] print("Prediction Evaluation using Linear Regression") print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred)) print("Mean Squared Error:", mean_squared_error(y_test, y_pred)) print("Root Mean Squared Error:", np.sqrt(mean_squared_error(y_test, y_pred))) print("R-squared:", r2_score(y_test, y_pred)) Prediction Evaluation using Linear Regression Mean Absolute Error: 0.6635761182069623 Mean Squared Error: 0.742453260904708 Root Mean Squared Error: 0.8616572757808562 R-squared: 0.1134733714697449</pre>
Lasso Regression	<pre>[40] LassoModel(Lasso(alpha = 0.1)) LassoModel(0.1) Lasso Lasso(alpha=0.1) y_pred = lasso.predict(x_test) print(y_pred) [4.70462482 5.27086104 5.10821120 5.2101708 4.61894498 5.94256981 4.8187446 5.15543386 5.51295937 4.84848957 4.52866758 4.81921317 4.69552481 4.64847131 4.90986323 4.33824279 5.24889992 5.62683312 5.24448138 5.18628998 4.89894584 4.78181351 5.49246754 5.18820816 5.98118874 4.81388888 4.8548118 4.54896176 5.07664727 5.11171511 5.07115875 5.21888895 4.68823881 4.78814681 4.72886683 5.15171375 4.78881388 4.8051081 4.61835881 4.81845847 4.91188751 5.1881614 5.21179593 5.10810802 4.82385937 4.51889584 5.12149554 4.67819496 4.68888881 5.27778889 4.42888889 4.42888889 5.46278889 5.18514118 4.85137501 5.85817501 5.17811374 5.177751 4.7815751 4.8047501 5.6309112 4.51888901 4.42888889 4.4214772 4.78788889 4.82388889 5.38891988 4.42888889 4.42888889 4.4024772 4.72724 4.80488889 5.18759583 4.88825051 5.1218713 4.56774334 5.15188575 5.1186887 5.4277244 5.10828889 5.10828889 4.56888889 4.56888889 5.4052072 5.4484848 4.7924242 5.2221228 5.4052072 5.2912992 4.6918136 5.56988889 4.6984848 5.17775138 4.37948889 4.5284848 5.48827751 5.1644898 4.6922222 5.1082122 5.2095050 5.1248479 4.9315042 4.88878889 4.5189195 5.27487788 4.57538889 4.7889527 4.6715333 4.4272888 5.084222 4.7282481 5.2887027 5.0924842 4.8488889 4.8818138 5.40181014 4.78888889]</pre>	0.14	<pre>y_pred = lasso.predict(x_test) print("Prediction Evaluation using Lasso Regression") print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred)) print("Mean Squared Error:", mean_squared_error(y_test, y_pred)) print("Root Mean Squared Error:", np.sqrt(mean_squared_error(y_test, y_pred))) print("R-squared:", r2_score(y_test, y_pred)) Prediction Evaluation using Lasso Regression Mean Absolute Error: 0.6296444264267669 Mean Squared Error: 0.7155358198781405 Root Mean Squared Error: 0.8458935938633958 R-squared: 0.1456141532515728</pre>
SVM	<pre>[43] #SVM MODEL [44] svr = SVR().fit(x,y) [45] y_pred = svr.predict(x_test) [46] accuracy = svr.score(x_test,y_test) print(accuracy) 0.4176454053391483</pre>	0.41	<pre>y_pred = svr.predict(x_test) print("Prediction Evaluation using svr Regression") print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred)) print("Mean Squared Error:", mean_squared_error(y_test, y_pred)) print("Root Mean Squared Error:", np.sqrt(mean_squared_error(y_test, y_pred))) print("R-squared:", r2_score(y_test, y_pred)) Prediction Evaluation using svr Regression Mean Absolute Error: 0.49633196595528446 Mean Squared Error: 0.48771357193244815 Root Mean Squared Error: 0.698364926828722 R-squared: 0.4176454053391483</pre>
Decision Tree	<pre>[44] dt = DecisionTreeRegressor(random_state = 0) dt.fit(x,y) DecisionTreeRegressor DecisionTreeRegressor(random_state=0) [48] y_pred = dt.predict(x_test) print(y_pred) [[5.8 5.1 5.9 4.8 4.1 5.1 5. 4.8 5.3 4.7 4.1 4.4 5.4 4.3 4.8 4.7 5.1 4.1 4.9 5.8 5.1 5.3 4.1 4.1 4.1 4.7 5.9 5.3 4.9 5.9 4.1 4.1 5.3 4.7 4.7 4.9 4.9 4.7 4.7 4.8 5.1 4.9 4.9 4.9 5.1 4.1 5.1 4.8 5.2 5. 4.6 4.5 4.1 5.1 4.1 4.6 4.8 4. 4.6 4.4 4.4 4.5 4.7 4.2 4.9 5.1 4.7 4.7 4.8 4.2 4. 5.0 5.2 4. 4.9 4.2 6.1 3.7 5.1 5.1 4.1 4.1]] [49] accuracy = dt.score(x_test,y_test) print(accuracy) 0.986452128267205</pre>	0.98	<pre>y_pred = dt.predict(x_test) print("Prediction Evaluation using decisiontree Regression") print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred)) print("Mean Squared Error:", mean_squared_error(y_test, y_pred)) print("Root Mean Squared Error:", np.sqrt(mean_squared_error(y_test, y_pred))) print("R-squared:", r2_score(y_test, y_pred)) Prediction Evaluation using decisiontree Regression Mean Absolute Error: 0.011346151846153877 Mean Squared Error: 0.06666666666666666 Root Mean Squared Error: 0.10651832633943249 R-squared: 0.986452128267205</pre>

Random forest

```
[2]: rf = RandomForestRegressor(n_estimators = 100, random_state = 0)
      rf.fit(X,y)
```

```
RandomForestRegressor
(RandomForestRegressor(random_state=0))
```

```
y_pred = rf.predict(x_test)
print(y_pred)
```

```
[ 0.095  0.116  0.0280  4.046  4.211  0.468
 4.0825  4.788  0.01  4.008  4.185  4.288
 0.821  0.082  1.042  4.0  0.181  4.872
 4.004  4.01  4.021  4.024  0.186  4.016
 4.872  4.972  4.797  4.006  4.02  4.274
 4.037  0.185  0.089  4.025  0.012  4.067
 4.0191667 4.015 0.009 0.191 0.119 4.786
 4.0 0.009 0.186 4.000 0.001 4.776
 0.001 0.185 0.08 0.015 0.021 4.088
 4.095 0.008 0.712 0.087 0.26 4.175
 0.005 4.099 4.014 4.004 0.281 0.746
 0.197 4.095 4.095 0.177 4.07 4.09016
 0.118 4.082 4.078 4.076 4.05 4.037
 0.02 4.04 4.059 0.084 0.087 4.054
 0.003 0.179 0.074 4.087 4.084 4.074
 4.065 4.05 0.087 4.0 4.03 0.334
 4.092 4.025 0.0205 4.047 4.029 4.095
 0.771 0.019 4.032 4.034 4.025 0.316
 0.089 4.067 0.022 0.0729 0.011 0.792
 4.0875 4.085 4.095 ]
```

0.93

```
y_pred = rf.predict(x_test)
print("Prediction Evaluation using Random Regression")
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error:", np.sqrt(mean_squared_error(y_test, y_pred)))
print("R-squared:", r2_score(y_test, y_pred))
```

Prediction Evaluation using Random Regression
Mean Absolute Error: 0.1625574874874877
Mean Squared Error: 0.05404362903371328
Root Mean Squared Error: 0.23247285655257321
R-squared: 0.9354691820163654