Classification of Fuel type of the car (Petrol or Diesel) using EXTREME LEARNING MACHINE

In [1151]:

```
import pandas as pd
import numpy as np
import math
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import accuracy_score
import time
from sklearn.metrics import mean_squared_error
import statistics
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

In [ ]:

```
```

New Section

In [1152]:

```
data = pd.read_csv('ToyotaCorolla.csv')
```

In [1153]:

```
print(data)
```

```
      Price  Age     KM FuelType    CC  Weight
0     13500   23  46986   Diesel  2000    1165
1     13750   23  72937   Diesel  2000    1165
2     13950   24  41711   Diesel  2000    1165
3     14950   26  48000   Diesel  2000    1165
4     13750   30  38500   Diesel  2000    1170
...     ...  ...    ...      ...   ...     ...
1431   7500   69  20544   Petrol  1300    1025
1432  10845   72  19000   Petrol  1300    1015
1433   8500   71  17016   Petrol  1300    1015
1434   7250   70  16916   Petrol  1300    1015
1435   6950   76      1   Petrol  1600    1114

[1436 rows x 6 columns]
```

New Section

In [1154]:

```
labelencoder = LabelEncoder()#----------------------------USing label encoder here
data['FuelType'] = labelencoder.fit_transform(data['FuelType'])
```

In [1155]:

```
data
```

Out[1155]:

	Price	Age	KM	FuelType	CC	Weight
0	13500	23	46986	1	2000	1165
1	13750	23	72937	1	2000	1165
2	13950	24	41711	1	2000	1165
3	14950	26	48000	1	2000	1165
4	13750	30	38500	1	2000	1170
...	...	...	...	...	...	...
1431	7500	69	20544	2	1300	1025
1432	10845	72	19000	2	1300	1015
1433	8500	71	17016	2	1300	1015
1434	7250	70	16916	2	1300	1015
1435	6950	76	1	2	1600	1114

1436 rows × 6 columns

In [1156]:

```
y = data.FuelType
X = data
```

In [1157]:

```
X= X.drop(columns="FuelType")
```

In [1158]:

```
X#---------viewing dataset
```

Out[1158]:

	Price	Age	KM	CC	Weight
0	13500	23	46986	2000	1165
1	13750	23	72937	2000	1165
2	13950	24	41711	2000	1165
3	14950	26	48000	2000	1165
4	13750	30	38500	2000	1170
...	...	...	...	...	...
1431	7500	69	20544	1300	1025
1432	10845	72	19000	1300	1015
1433	8500	71	17016	1300	1015
1434	7250	70	16916	1300	1015
1435	6950	76	1	1600	1114

1436 rows × 5 columns

In [1159]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.7)#----------splitting
print(y_train.shape)
print(y_test.shape)
```

```
(430,)
(1006,)
```

In [ ]:

```

```

In [1160]:

```
X_train=pd.DataFrame(X_train).to_numpy()#--------------pandas to numpy datfrme

w = np.random.rand(5,1000)*100#-----------------------------creating random weight matrix=
weights = pd.DataFrame(w)

#weights =to_numpy.DataFrame(w)
weights=pd.DataFrame(weights).to_numpy()

weights_transpose = np.transpose(weights)
print(weights.shape)
print(X_train.shape)
print(type(weights))
print(type(X_train))
```

```
(5, 1000)
(430, 5)
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

In [1161]:

```
X_train
```

Out[1161]:

```
array([[ 13500,     42,  38932,   1600,   1040],
       [  7950,     69,  83133,   1300,   1015],
       [ 10450,     64, 120400,   1600,   1090],
       ...,
       [  8950,     76,  89520,   1600,   1050],
       [  8450,     70, 124743,   1600,   1050],
       [ 18450,     10,  13747,   1400,   1110]], dtype=int64)
```

In [1162]:

```
X_train=pd.DataFrame(X_train).to_numpy()

w = np.random.rand(1000,5)*100#-----------------------------creating random weight matrix=
weights = pd.DataFrame(w)

#weights =to_numpy.DataFrame(w)
weights=pd.DataFrame(weights).to_numpy()

weights_t= np.transpose(weights)
print(weights.shape)
print(X_train.shape)
print(type(weights))
print(type(X_train))
print(weights_t.shape)
```

```
(1000, 5)
(430, 5)
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
(5, 1000)
```

In [1163]:

```
h_new = np.dot(X_train,weights_t)
print(h_new.shape)
h_t = np.transpose(h_new)
h_tt = np.dot(h_t,h_new)
```

```
(430, 1000)
```

In [1164]:

```
h_inv = np.linalg.pinv(h_tt)#-----------------finding h invere
h_p = np.dot(h_inv,h_t)
print(h_p.shape)
```

```
(1000, 430)
```

In [1165]:

```
beta = np.dot(h_p, y_train)#----------------------finding beta using dot product
```

In [1166]:

```
h_new[1:10]
print(h_new.shape)
print(beta.shape)
```

```
(430, 1000)
(1000,)
```

In [1167]:

```
print(beta[1:10])
```

```
[-3.36868410e-07 -3.33085757e-07 -8.63992023e-08 -1.15208070e-08
 -4.31866381e-07  3.69409184e-08  7.46856695e-08  4.36240202e-08
  3.41427306e-07]
```

In [1168]:

```
predicted_output= np.dot(h_new, beta) #-------------------------------------------------
print(predicted_output.shape)
```

```
(430,)
```

In [1169]:

```
predicted_output[1:10]
```

Out[1169]:

```
array([2.04740131, 1.8023068 , 1.76529775, 1.76568129, 1.39798615,
       1.84947038, 1.61482888, 1.40362265, 1.98979382])
```

In [1170]:

```
rounded_labels=np.round(predicted_output)
rounded=pd.DataFrame(rounded_labels).to_numpy()
```

In [1171]:

```
rounded_labels[1:10]
```

Out[1171]:

```
array([2., 2., 2., 2., 1., 2., 2., 1., 2.])
```

Final Training Accuracy

In [1172]:

```
acc = accuracy_score(rounded_labels,y_train)#----------------------------FInding Training
print("Training Accuracy is", + acc*100, "%")
```

```
Training Accuracy is 95.81395348837209 %
```

In [1173]:

```
#--------TESTING accuracy starts here
h_new_test = np.dot(X_test,weights_t)
predicted_output_test= np.dot(h_new_test, beta)#--------------this beta was calcualated dur
```

In [1174]:

```
rounded_labels_test=np.round(predicted_output_test)
rounded_test=pd.DataFrame(rounded_labels_test).to_numpy()
```

Final Testing Accuracy

In [1175]:

```
acc_test = accuracy_score(rounded_labels_test,y_test)#----------------------------FInding
print("Testing Accuracy is", + acc_test*100, "%")
```

```
Testing Accuracy is 93.93638170974154 %
```

In [ ]:

```

```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: