Mercedes-Benz

March 8, 2024

0.1 Mercedes-Benz Greener Manufacturing

Objective-You are required to reduce the time that cars spend on the test bench. Others will work with a dataset representing different permutations of features in a Mercedes-Benz car to predict the time it takes to pass testing. Optimal algorithms will contribute to faster testing, resulting in lower carbon dioxide emissions without reducing Mercedes-Benz's standards.

[1]:
```
# Importing the required libraries

import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
# Importing the data

train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

[6]:
```
train.head()
```

[6]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	…	X375	X376	X377	X378	X379	\
0	0	130.81	k	v	at	a	d	u	j	o	…	0	0	1	0	0	
1	6	88.53	k	t	av	e	d	y	l	o	…	1	0	0	0	0	
2	7	76.26	az	w	n	c	d	x	j	x	…	0	0	0	0	0	
3	9	80.62	az	t	n	f	d	x	l	e	…	0	0	0	0	0	
4	13	78.02	az	v	n	f	d	h	d	n	…	0	0	0	0	0	

	X380	X382	X383	X384	X385
0	0	0	0	0	0
1	0	0	0	0	0
2	0	1	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0

```
[5 rows x 378 columns]
```

[4]:
```
test.head()
```

[4]:
```
   ID  X0 X1  X2 X3 X4 X5 X6 X8  X10  …  X375  X376  X377  X378  X379  X380  \
0   1  az  v   n  f  d  t  a  w    0  …     0     0     0     1     0     0
1   2   t  b  ai  a  d  b  g  y    0  …     0     0     1     0     0     0
2   3  az  v  as  f  d  a  j  j    0  …     0     0     0     1     0     0
3   4  az  l   n  f  d  z  l  n    0  …     0     0     0     1     0     0
4   5   w  s  as  c  d  y  i  m    0  …     1     0     0     0     0     0

   X382  X383  X384  X385
0     0     0     0     0
1     0     0     0     0
2     0     0     0     0
3     0     0     0     0
4     0     0     0     0

[5 rows x 377 columns]
```

[7]:
```
train.columns
```

[7]:
```
Index(['ID', 'y', 'X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8',
       …
       'X375', 'X376', 'X377', 'X378', 'X379', 'X380', 'X382', 'X383', 'X384',
       'X385'],
      dtype='object', length=378)
```

[8]:
```
test.columns
```

[8]:
```
Index(['ID', 'X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8', 'X10',
       …
       'X375', 'X376', 'X377', 'X378', 'X379', 'X380', 'X382', 'X383', 'X384',
       'X385'],
      dtype='object', length=377)
```

[9]:
```
train.shape
```

[9]:
```
(4209, 378)
```

[10]:
```
test.shape
```

[10]:
```
(4209, 377)
```

[12]:
```
# Collect the Y values into an array
y_train = train['y'].values
```

```
y_train
```

[12]:
```
array([130.81,  88.53,  76.26, …, 109.22,  87.48, 110.85])
```

[13]:
```
# Understanding the data types:
cols = [c for c in train.columns if 'X' in c]
print('Number of features: {}'.format(len(cols)))
print('Feature types:')
train[cols].dtypes.value_counts()
```

```
Number of features: 376
Feature types:
```

[13]:
```
int64     368
object      8
dtype: int64
```

[14]:
```
# Count the data in each of the columns

counts = [[], [], []]
for c in cols:
    typ = train[c].dtype
    uniq = len(np.unique(train[c]))
    if uniq ==1:
        counts[0].append(c)
    elif uniq == 2 and typ ==np.int64:
        counts[1].append(c)
    else:
        counts[2].append(c)
print('Constant features: {} Binary feature: {} Categorical features: {}\n'
 .format(*[len(c) for c in counts]))
print('Constant features:',counts[0])
print('Categorical features:', counts[2])
```

```
Constant features: 12 Binary feature: 356 Categorical features: 8

Constant features: ['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289',
'X290', 'X293', 'X297', 'X330', 'X347']
Categorical features: ['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8']
```

[15]:
```
# Splitting the data

usable_columns = list(set(train.columns) - set(['ID','y']))
y_train = train['y'].values
id_test = test['ID'].values
x_train = train[usable_columns]
x_test = test[usable_columns]
```

Checking for null values and unique values for train and test data

[20]:
```
x_train.isnull().any().any()
```

[20]: False

[21]:
```
x_test.isnull().any().any()
```

[21]: False

0.2 Label Encoding the Categorical Values

[22]:
```
for column in usable_columns:
    cardinality = len(np.unique(x_train[column]))
    if cardinality == 1:
        x_train.drop(column, axis=1) # column with only one
        # value is useless so we drop it.
        x_test.drop(column, axis=1)
    if cardinality >2: # Column is categorical
        mapper = lambda x: sum([ord(digit) for digit in x])
        x_train[column] = x_train[column].apply(mapper)
        x_test[column] = x_test[column].apply(mapper)
x_train.head()
```

[22]:

	X315	X111	X156	X124	X207	X218	X17	X164	X74	X31	…	X34	X357	\
0	0	1	1	0	0	0	0	0	1	1	…	0	0	
1	0	1	1	0	0	0	0	0	1	1	…	0	0	
2	0	1	0	0	0	1	1	0	1	1	…	0	0	
3	0	1	0	0	0	1	0	0	1	1	…	0	0	
4	0	1	0	0	0	1	0	0	1	1	…	0	0	

	X383	X220	X350	X77	X329	X270	X150	X69
0	0	1	0	0	1	0	1	0
1	0	0	0	0	1	0	1	0
2	0	1	1	0	0	0	1	0
3	0	1	1	0	0	0	1	0
4	0	1	1	0	0	0	1	0

[5 rows x 376 columns]

[23]:
```
# Make sure the data is changed into numerical values

print('feature dtypes:')
x_train[cols].dtypes.value_counts()
```

feature dtypes:

[23]:
```
int64    376
dtype: int64
```

0.3 Perform dimensionality reduction.

[24]:
```
n_comp = 12
pca = PCA(n_components = n_comp,random_state = 420)
pca2_results_train = pca.fit_transform(x_train)
pca2_results_test = pca.transform(x_test)
```

[28]:
```
pca2_results_train
```

[28]:
```
array([[-49.08156207,  -4.90948084, -17.25085325, …,   1.65805072,
          0.93297242,   1.67842477],
       [-48.94680383,  -7.22674339, -13.7631947 , …,  -0.21428893,
          0.10899682,   0.44965265],
       [ 92.62761708,  31.9940341 , -26.17503456, …,  -0.62195786,
          2.92596081,  -0.52732605],
       …,
       [ 89.47970814,  20.44554421,  48.11999819, …,  -1.27196174,
         -0.2873013 ,   2.00806385],
       [ 96.97110845,  31.50977186,  49.20059282, …,   0.14366004,
         -0.9797229 ,   0.99172893],
       [-17.21024322, -14.22166025,  55.38091289, …,  -0.28904432,
         -0.31653098,   0.69155615]])
```

[26]:
```
pca2_results_test
```

[26]:
```
array([[ 9.22615149e+01,  3.29260839e+01, -3.01130736e+01, …,
        -4.11418166e-01,  3.62103016e+00, -1.20767016e+00],
       [-3.48622379e+01,  6.87132606e+00, -3.74760829e+01, …,
         6.09270156e-01, -6.95837529e-01, -4.24915489e-01],
       [ 4.36560426e+01, -5.05939489e+01, -6.10591086e+01, …,
        -3.20457644e-01,  2.60144467e+00, -1.53760386e+00],
       …,
       [-2.52437784e+01, -2.63794193e+01,  5.40742341e+01, …,
         6.03526031e-01,  2.61277676e-02,  3.67039655e-02],
       [ 4.53823778e+01, -6.38062446e+01,  3.58666036e+01, …,
        -9.15188266e-01, -6.72303829e-01,  5.15228832e-01],
       [-4.23807477e+01, -2.52862351e+01,  6.10815522e+01, …,
        -2.98851963e-01, -9.77085229e-01,  5.35179833e-02]])
```

0.4 Predict your test_df values using XGBoost.

[29]:
```
# Training Using XGBoost

import xgboost as xgb
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
```

[30]:
```
x_train,x_val,y_train,y_val = train_test_split(pca2_results_train, y_train,␣
↪test_size=0.2, random_state=4242)
```

[31]:
```
d_train = xgb.DMatrix(x_train,label = y_train)
d_val = xgb.DMatrix(x_val,label = y_val)

# dtest = xgb.DMatrix(x_test)

d_test = xgb.DMatrix(pca2_results_test)
```

[32]:
```
params = {}
params['Objective'] = 'reg:linear'
params['eta'] = 0.02
params['max_depth'] = 4

def xgb_r2_score(preds, dtrain):
    labels = dtrain.get_label()
    return 'r2', r2_score(labels, preds)
watchlist = [(d_train, 'train'),(d_val,'valid')]
clf = xgb.train(params, d_train, 1000, watchlist, early_stopping_rounds=50,
                feval=xgb_r2_score, maximize=True, verbose_eval=10)
```

```
[10:00:42] WARNING: ../src/learner.cc:627:
Parameters: { "Objective" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


[0]     train-rmse:99.14834     train-r2:-58.35295      valid-rmse:98.26297
valid-r2:-67.63754
[10]    train-rmse:81.27653     train-r2:-38.88428      valid-rmse:80.36433
valid-r2:-44.91014
[20]    train-rmse:66.71610     train-r2:-25.87403      valid-rmse:65.77334
valid-r2:-29.75260
```

```
[30]    train-rmse:54.86912     train-r2:-17.17722      valid-rmse:53.89136
valid-r2:-19.64525
[40]    train-rmse:45.24709     train-r2:-11.36097      valid-rmse:44.22323
valid-r2:-12.90218
[50]    train-rmse:37.44854     train-r2:-7.46723       valid-rmse:36.37628
valid-r2:-8.40630
[60]    train-rmse:31.14584     train-r2:-4.85695       valid-rmse:30.02266
valid-r2:-5.40738
[70]    train-rmse:26.08417     train-r2:-3.10795       valid-rmse:24.91510
valid-r2:-3.41273
[80]    train-rmse:22.04312     train-r2:-1.93371       valid-rmse:20.83068
valid-r2:-2.08453
[90]    train-rmse:18.84671     train-r2:-1.14458       valid-rmse:17.59609
valid-r2:-1.20097
[100]   train-rmse:16.33297     train-r2:-0.61065       valid-rmse:15.07907
valid-r2:-0.61633
[110]   train-rmse:14.39787     train-r2:-0.25161       valid-rmse:13.14761
valid-r2:-0.22878
[120]   train-rmse:12.92938     train-r2:-0.00932       valid-rmse:11.69322
valid-r2:0.02804
[130]   train-rmse:11.81501     train-r2:0.15717        valid-rmse:10.61718
valid-r2:0.19869
[140]   train-rmse:10.98634     train-r2:0.27125        valid-rmse:9.84977
valid-r2:0.31034
[150]   train-rmse:10.37862     train-r2:0.34964        valid-rmse:9.31622
valid-r2:0.38303
[160]   train-rmse:9.92636      train-r2:0.40509        valid-rmse:8.95744
valid-r2:0.42964
[170]   train-rmse:9.59382      train-r2:0.44428        valid-rmse:8.71413
valid-r2:0.46020
[180]   train-rmse:9.34595      train-r2:0.47263        valid-rmse:8.55244
valid-r2:0.48005
[190]   train-rmse:9.15988      train-r2:0.49342        valid-rmse:8.44786
valid-r2:0.49269
[200]   train-rmse:9.01715      train-r2:0.50908        valid-rmse:8.38564
valid-r2:0.50013
[210]   train-rmse:8.91491      train-r2:0.52015        valid-rmse:8.34641
valid-r2:0.50480
[220]   train-rmse:8.82930      train-r2:0.52932        valid-rmse:8.32277
valid-r2:0.50760
[230]   train-rmse:8.76269      train-r2:0.53640        valid-rmse:8.30887
valid-r2:0.50924
[240]   train-rmse:8.71004      train-r2:0.54195        valid-rmse:8.30193
valid-r2:0.51006
[250]   train-rmse:8.66384      train-r2:0.54680        valid-rmse:8.29649
valid-r2:0.51070
[260]   train-rmse:8.62535      train-r2:0.55082        valid-rmse:8.29381
valid-r2:0.51102
```

```
[270]   train-rmse:8.59332      train-r2:0.55414        valid-rmse:8.29034
valid-r2:0.51143
[280]   train-rmse:8.56503      train-r2:0.55708        valid-rmse:8.28796
valid-r2:0.51171
[290]   train-rmse:8.53976      train-r2:0.55969        valid-rmse:8.28659
valid-r2:0.51187
[300]   train-rmse:8.51381      train-r2:0.56236        valid-rmse:8.28935
valid-r2:0.51155
[310]   train-rmse:8.48420      train-r2:0.56540        valid-rmse:8.28632
valid-r2:0.51190
[320]   train-rmse:8.46249      train-r2:0.56762        valid-rmse:8.28691
valid-r2:0.51183
[330]   train-rmse:8.43999      train-r2:0.56991        valid-rmse:8.28423
valid-r2:0.51215
[340]   train-rmse:8.41568      train-r2:0.57239        valid-rmse:8.28313
valid-r2:0.51228
[350]   train-rmse:8.38980      train-r2:0.57501        valid-rmse:8.28000
valid-r2:0.51265
[360]   train-rmse:8.36332      train-r2:0.57769        valid-rmse:8.27881
valid-r2:0.51279
[370]   train-rmse:8.33579      train-r2:0.58047        valid-rmse:8.27188
valid-r2:0.51360
[380]   train-rmse:8.31223      train-r2:0.58284        valid-rmse:8.27152
valid-r2:0.51365
[390]   train-rmse:8.29179      train-r2:0.58488        valid-rmse:8.27227
valid-r2:0.51356
[400]   train-rmse:8.26806      train-r2:0.58726        valid-rmse:8.26870
valid-r2:0.51398
[410]   train-rmse:8.23972      train-r2:0.59008        valid-rmse:8.26509
valid-r2:0.51440
[420]   train-rmse:8.21456      train-r2:0.59258        valid-rmse:8.26054
valid-r2:0.51494
[430]   train-rmse:8.18769      train-r2:0.59524        valid-rmse:8.26058
valid-r2:0.51493
[440]   train-rmse:8.16357      train-r2:0.59762        valid-rmse:8.26004
valid-r2:0.51499
[450]   train-rmse:8.13851      train-r2:0.60009        valid-rmse:8.26038
valid-r2:0.51496
[460]   train-rmse:8.11633      train-r2:0.60227        valid-rmse:8.26123
valid-r2:0.51486
[470]   train-rmse:8.09531      train-r2:0.60433        valid-rmse:8.25834
valid-r2:0.51519
[480]   train-rmse:8.07090      train-r2:0.60671        valid-rmse:8.25674
valid-r2:0.51538
[490]   train-rmse:8.05218      train-r2:0.60853        valid-rmse:8.25828
valid-r2:0.51520
[500]   train-rmse:8.02514      train-r2:0.61116        valid-rmse:8.25752
valid-r2:0.51529
```

```
[510]   train-rmse:7.99972      train-r2:0.61361        valid-rmse:8.25632
valid-r2:0.51543
[520]   train-rmse:7.97983      train-r2:0.61553        valid-rmse:8.25644
valid-r2:0.51542
[530]   train-rmse:7.95794      train-r2:0.61764        valid-rmse:8.25607
valid-r2:0.51546
[532]   train-rmse:7.95286      train-r2:0.61813        valid-rmse:8.25679
valid-r2:0.51538
```

[33]:
```
p_test = clf.predict(d_test)
```

[35]:
```
sub = pd.DataFrame()
sub['ID'] = id_test
sub['y'] = p_test
sub.to_csv('test_df.csv', index = False)
sub.head(10)
```

[35]:

	ID	y
0	1	83.153229
1	2	96.980148
2	3	82.968224
3	4	76.981483
4	5	112.925842
5	8	91.509911
6	10	100.502510
7	11	93.808548
8	12	117.162224
9	14	95.924377

[ ]:
```

```