Merceded_Benz

July 13, 2023

[1]:

```
# Importing the required libraries
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
```

[3]:

```
# Importing the data

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

[4]:

```
train.head()
```

[4]:

```
   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]
```

[5]:

```
test.head()
```

[5]:

```
   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]
```

[6]:

```
print('Size of training set: {} rows and {} columns'.format(*train.shape))
print('Size of testing set: {} rows and {} columns'.format(*test.shape))
```

```
Size of training set: 4209 rows and 378 columns
Size of testing set: 4209 rows and 377 columns
```

[7]:

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

[8]:

```
y_train
```

[8]:

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

[9]:

```
# Understand 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:
```

[9]:

```
int64     368
object      8
dtype: int64
```

[10]:

```
# 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 features: {} Categorical features: {}\n'
 .format(*[len(c) for c in counts]))
print('Constant features:', counts[0])
print('Categorical features:', counts[2])
```

```
Constant features: 12 Binary features: 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']
```

[11]:

```
# 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]
```

Check for null values and unique values for train & test data

[12]:

```
def check_missing_values(df):
    if df.isnull().any().any():
        print('There are missing values in the dataframe')
    else:
        print('There are no missing values in the dataframe')
```

[13]:

```
check_missing_values(x_train)
check_missing_values(x_test)
```

```
There are no missing values in the dataframe
There are no missing values in the dataframe
```

Label Encoding the categorical values

[14]:

```
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()
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:9:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  if __name__ == '__main__':
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:10:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  # Remove the CWD from sys.path while we load stuff.
```

```
[14]:    X42  X50  X246  X379  X155  X118  X294  X143  X101  X22  …  X124  X233  \
      0    0    0     0     0     0     1     0     0     0    0  …     0     0
      1    0    0     0     0     0     1     0     0     1    0  …     0     0
      2    0    0     1     0     0     0     0     0     1    0  …     0     0
      3    0    0     1     0     0     0     0     0     1    0  …     0     0
      4    0    0     1     0     0     0     0     0     1    0  …     0     0

         X225  X270  X323  X277  X275  X205  X80  X67
      0     0     0     0     0     1     0    0    0
      1     0     0     0     0     1     1    1    0
      2     0     0     0     0     0     1    1    0
      3     0     0     0     0     0     1    1    0
      4     0     0     0     0     0     1    1    0

      [5 rows x 376 columns]
```

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

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

```
Feature types:
```

```
[15]: int64    376
      dtype: int64
```

Perform Dimensionality reduction

```
[16]: 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)
```

Training using XGBoost

[17]:

```
# Training using XGBoost

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

[18]:

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

[19]:

```
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)
```

[20]:

```
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)
```

```
[12:48:41] WARNING: /workspace/src/objective/regression_obj.cu:167: reg:linear
is now deprecated in favor of reg:squarederror.
[0]     train-rmse:99.14835     valid-rmse:98.26297     train-r2:-58.35295
valid-r2:-67.63754
Multiple eval metrics have been passed: 'valid-r2' will be used for early
stopping.

Will train until valid-r2 hasn't improved in 50 rounds.
[10]    train-rmse:81.27653     valid-rmse:80.36433     train-r2:-38.88428
valid-r2:-44.91014
[20]    train-rmse:66.71610     valid-rmse:65.77334     train-r2:-25.87403
valid-r2:-29.75260
[30]    train-rmse:54.86915     valid-rmse:53.89120     train-r2:-17.17724
valid-r2:-19.64513
[40]    train-rmse:45.24564     valid-rmse:44.22231     train-r2:-11.36018
valid-r2:-12.90160
[50]    train-rmse:37.44742     valid-rmse:36.37758     train-r2:-7.46672
valid-r2:-8.40697
[60]    train-rmse:31.15106     valid-rmse:30.01765     train-r2:-4.85891
```

```
valid-r2:-5.40524
[70]    train-rmse:26.08771     valid-rmse:24.90846     train-r2:-3.10907
valid-r2:-3.41038
[80]    train-rmse:22.04900     valid-rmse:20.82502     train-r2:-1.93528
valid-r2:-2.08285
[90]    train-rmse:18.84765     valid-rmse:17.60010     train-r2:-1.14480
valid-r2:-1.20198
[100]   train-rmse:16.33699     valid-rmse:15.08526     train-r2:-0.61145
valid-r2:-0.61766
[110]   train-rmse:14.39788     valid-rmse:13.15610     train-r2:-0.25161
valid-r2:-0.23037
[120]   train-rmse:12.93041     valid-rmse:11.69388     train-r2:-0.00948
valid-r2:0.02793
[130]   train-rmse:11.81665     valid-rmse:10.62117     train-r2:0.15694
valid-r2:0.19809
[140]   train-rmse:10.98570     valid-rmse:9.85576      train-r2:0.27134
valid-r2:0.30950
[150]   train-rmse:10.37823     valid-rmse:9.32776      train-r2:0.34969
valid-r2:0.38150
[160]   train-rmse:9.92529      valid-rmse:8.96124      train-r2:0.40522
valid-r2:0.42916
[170]   train-rmse:9.59273      valid-rmse:8.71470      train-r2:0.44441
valid-r2:0.46013
[180]   train-rmse:9.34136      valid-rmse:8.55182      train-r2:0.47314
valid-r2:0.48013
[190]   train-rmse:9.16018      valid-rmse:8.44863      train-r2:0.49338
valid-r2:0.49260
[200]   train-rmse:9.01539      valid-rmse:8.38342      train-r2:0.50927
valid-r2:0.50040
[210]   train-rmse:8.91053      valid-rmse:8.34375      train-r2:0.52062
valid-r2:0.50511
[220]   train-rmse:8.83439      valid-rmse:8.31880      train-r2:0.52878
valid-r2:0.50807
[230]   train-rmse:8.76692      valid-rmse:8.30589      train-r2:0.53595
valid-r2:0.50960
[240]   train-rmse:8.71890      valid-rmse:8.30186      train-r2:0.54102
valid-r2:0.51007
[250]   train-rmse:8.67835      valid-rmse:8.29627      train-r2:0.54528
valid-r2:0.51073
[260]   train-rmse:8.63351      valid-rmse:8.29345      train-r2:0.54997
valid-r2:0.51106
[270]   train-rmse:8.59840      valid-rmse:8.28935      train-r2:0.55362
valid-r2:0.51155
[280]   train-rmse:8.57183      valid-rmse:8.28958      train-r2:0.55637
valid-r2:0.51152
[290]   train-rmse:8.54671      valid-rmse:8.28931      train-r2:0.55897
valid-r2:0.51155
[300]   train-rmse:8.51790      valid-rmse:8.28741      train-r2:0.56194
```

```
valid-r2:0.51177
[310]   train-rmse:8.48728      valid-rmse:8.28676      train-r2:0.56508
valid-r2:0.51185
[320]   train-rmse:8.45935      valid-rmse:8.28654      train-r2:0.56794
valid-r2:0.51188
[330]   train-rmse:8.43960      valid-rmse:8.28397      train-r2:0.56995
valid-r2:0.51218
[340]   train-rmse:8.41649      valid-rmse:8.28167      train-r2:0.57231
valid-r2:0.51245
[350]   train-rmse:8.39454      valid-rmse:8.27978      train-r2:0.57453
valid-r2:0.51267
[360]   train-rmse:8.37122      valid-rmse:8.28102      train-r2:0.57689
valid-r2:0.51253
[370]   train-rmse:8.34339      valid-rmse:8.27881      train-r2:0.57970
valid-r2:0.51279
[380]   train-rmse:8.31823      valid-rmse:8.27762      train-r2:0.58223
valid-r2:0.51293
[390]   train-rmse:8.29572      valid-rmse:8.27565      train-r2:0.58449
valid-r2:0.51316
[400]   train-rmse:8.27208      valid-rmse:8.27303      train-r2:0.58686
valid-r2:0.51347
[410]   train-rmse:8.24628      valid-rmse:8.26927      train-r2:0.58943
valid-r2:0.51391
[420]   train-rmse:8.22016      valid-rmse:8.26749      train-r2:0.59203
valid-r2:0.51412
[430]   train-rmse:8.18886      valid-rmse:8.26454      train-r2:0.59513
valid-r2:0.51447
[440]   train-rmse:8.16603      valid-rmse:8.26397      train-r2:0.59738
valid-r2:0.51453
[450]   train-rmse:8.13853      valid-rmse:8.26562      train-r2:0.60009
valid-r2:0.51434
[460]   train-rmse:8.11520      valid-rmse:8.26616      train-r2:0.60238
valid-r2:0.51428
[470]   train-rmse:8.09094      valid-rmse:8.26538      train-r2:0.60475
valid-r2:0.51437
[480]   train-rmse:8.06924      valid-rmse:8.26371      train-r2:0.60687
valid-r2:0.51456
[490]   train-rmse:8.04953      valid-rmse:8.26259      train-r2:0.60879
valid-r2:0.51470
[500]   train-rmse:8.02556      valid-rmse:8.26429      train-r2:0.61111
valid-r2:0.51450
[510]   train-rmse:8.00515      valid-rmse:8.26562      train-r2:0.61309
valid-r2:0.51434
[520]   train-rmse:7.98177      valid-rmse:8.26275      train-r2:0.61535
valid-r2:0.51468
[530]   train-rmse:7.96485      valid-rmse:8.26534      train-r2:0.61698
valid-r2:0.51437
Stopping. Best iteration:
```

```
[489]   train-rmse:8.05238      valid-rmse:8.26128      train-r2:0.60851
valid-r2:0.51485
```

Predict test_df using XGBoost

[21]:

```
p_test = clf.predict(d_test)
```

[22]:

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

[22]:

```
   ID           y
0   1   82.865776
1   2   97.628395
2   3   83.197395
3   4   77.039124
4   5  112.527901
```

[ ]: