Mercedes-Benz Greener Manufacturing

[2]:

```
######################################################################
'''    Project No. 1: Mercedes-Benz Greener Manufacturing       '''
######################################################################

# Step1: Import the required libraries

# linear algebra
import numpy as np
# data processing, CSV file I/O (e.g. pd.read_csv)
import pandas as pd
# for dimensionality reduction
from sklearn.decomposition import PCA
```

[3]:

```
# Step2: Read the data from train.csv

df_train = pd.read_csv('train.csv')
# let us understand the data
print('Size of training set: {} rows and {} columns'
      .format(*df_train.shape))
# print few rows and see how the data looks like
df_train.head()
```

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

[3]:

```
   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]:

```
# Step3: Collect the Y values into an array

# seperate the y from the data as we will use this to learn as
# the prediction output
y_train = df_train['y'].values
```

[5]:

```
# Step4: Understand the data types we have

# iterate through all the columns which has X in the name of the column
cols = [c for c in df_train.columns if 'X' in c]
print('Number of features: {}'.format(len(cols)))

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

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

[5]:

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

[6]:

```
# Step5: Count the data in each of the columns

counts = [[], [], []]
for c in cols:
    typ = df_train[c].dtype
    uniq = len(np.unique(df_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']
```

[7]:

```python
# Step6: Read the test.csv data

df_test = pd.read_csv('test.csv')

# remove columns ID and Y from the data as they are not used for learning
usable_columns = list(set(df_train.columns) - set(['ID', 'y']))
y_train = df_train['y'].values
id_test = df_test['ID'].values

x_train = df_train[usable_columns]
x_test = df_test[usable_columns]
```

[8]:

```python
# Step7: Check for null and unique values for test and train sets

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")
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
```

[9]:

```python
# Step8: If for any column(s), the variance is equal to zero,
# then you need to remove those variable(s).
# Apply label encoder

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:13:
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
```

```
  del sys.path[0]
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:14:
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
```

[9]:

```
   X29  X37  X81  X229  X339  X95  X110  X138  X321  X26  …  X241  X325  \
0    0    1    0     0     0    0     0     0     0    0  …     0     0
1    0    1    0     1     0    0     0     0     0    0  …     0     0
2    1    1    0     0     0    0     0     0     0    0  …     1     0
3    1    1    0     1     0    0     0     0     0    0  …     1     0
4    1    1    0     1     0    0     0     0     0    0  …     0     0

   X108  X286  X263  X35  X365  X210  X297  X83
0     0     0     1    1     0     0     0    0
1     0     0     1    1     0     0     0    0
2     0     1     0    1     0     0     0    0
3     1     1     0    1     0     0     0    0
4     1     1     0    1     0     0     0    0

[5 rows x 376 columns]
```

[10]:

```python
# Step9: Make sure the data is now changed into numericals

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

```
Feature types:
```

[10]:

```
int64    376
dtype: int64
```

[11]:

```python
# Step10: Perform dimensionality reduction
# Linear dimensionality reduction using Singular Value Decomposition of
# the data to project it to a lower dimensional space.
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)
```

[12]:

```python
# Step11: Training using xgboost

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

x_train, x_valid, y_train, y_valid = train_test_split(
        pca2_results_train,
        y_train, test_size=0.2,
        random_state=4242)

d_train = xgb.DMatrix(x_train, label=y_train)
d_valid = xgb.DMatrix(x_valid, label=y_valid)
#d_test = xgb.DMatrix(x_test)
d_test = xgb.DMatrix(pca2_results_test)

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_valid, 'valid')]

clf = xgb.train(params, d_train,
                1000, watchlist, early_stopping_rounds=50,
                feval=xgb_r2_score, maximize=True, verbose_eval=10)
```

```
[01:36:15] 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:53.89103     train-r2:-17.17724
valid-r2:-19.64500
[40]    train-rmse:45.24563     valid-rmse:44.22213     train-r2:-11.36018
valid-r2:-12.90149
[50]    train-rmse:37.44740     valid-rmse:36.37753     train-r2:-7.46671
valid-r2:-8.40694
[60]    train-rmse:31.15104     valid-rmse:30.01760     train-r2:-4.85891
valid-r2:-5.40522
```

```
[70]    train-rmse:26.08689     valid-rmse:24.90709     train-r2:-3.10881
valid-r2:-3.40989
[80]    train-rmse:22.04886     valid-rmse:20.82341     train-r2:-1.93524
valid-r2:-2.08238
[90]    train-rmse:18.85231     valid-rmse:17.59764     train-r2:-1.14586
valid-r2:-1.20136
[100]   train-rmse:16.34305     valid-rmse:15.08330     train-r2:-0.61264
valid-r2:-0.61724
[110]   train-rmse:14.40352     valid-rmse:13.15132     train-r2:-0.25259
valid-r2:-0.22948
[120]   train-rmse:12.92834     valid-rmse:11.69463     train-r2:-0.00915
valid-r2:0.02780
[130]   train-rmse:11.81810     valid-rmse:10.62550     train-r2:0.15673
valid-r2:0.19743
[140]   train-rmse:10.98451     valid-rmse:9.86766      train-r2:0.27149
valid-r2:0.30783
[150]   train-rmse:10.37958     valid-rmse:9.32896      train-r2:0.34952
valid-r2:0.38135
[160]   train-rmse:9.93296      valid-rmse:8.96920      train-r2:0.40430
valid-r2:0.42814
[170]   train-rmse:9.60073      valid-rmse:8.72622      train-r2:0.44348
valid-r2:0.45870
[180]   train-rmse:9.35625      valid-rmse:8.56540      train-r2:0.47146
valid-r2:0.47847
[190]   train-rmse:9.16724      valid-rmse:8.46754      train-r2:0.49260
valid-r2:0.49032
[200]   train-rmse:9.02452      valid-rmse:8.40277      train-r2:0.50828
valid-r2:0.49809
[210]   train-rmse:8.92179      valid-rmse:8.36758      train-r2:0.51941
valid-r2:0.50228
[220]   train-rmse:8.84667      valid-rmse:8.34451      train-r2:0.52747
valid-r2:0.50502
[230]   train-rmse:8.78950      valid-rmse:8.33266      train-r2:0.53356
valid-r2:0.50643
[240]   train-rmse:8.73954      valid-rmse:8.32621      train-r2:0.53884
valid-r2:0.50719
[250]   train-rmse:8.69151      valid-rmse:8.32158      train-r2:0.54390
valid-r2:0.50774
[260]   train-rmse:8.65760      valid-rmse:8.32211      train-r2:0.54745
valid-r2:0.50768
[270]   train-rmse:8.62205      valid-rmse:8.31773      train-r2:0.55116
valid-r2:0.50820
[280]   train-rmse:8.59481      valid-rmse:8.31729      train-r2:0.55399
valid-r2:0.50825
[290]   train-rmse:8.56563      valid-rmse:8.31759      train-r2:0.55701
valid-r2:0.50821
[300]   train-rmse:8.54286      valid-rmse:8.31700      train-r2:0.55937
valid-r2:0.50828
```

```
[310]   train-rmse:8.51938      valid-rmse:8.31458      train-r2:0.56178
valid-r2:0.50857
[320]   train-rmse:8.49042      valid-rmse:8.31094      train-r2:0.56476
valid-r2:0.50900
[330]   train-rmse:8.45625      valid-rmse:8.30934      train-r2:0.56825
valid-r2:0.50919
[340]   train-rmse:8.42987      valid-rmse:8.31028      train-r2:0.57094
valid-r2:0.50908
[350]   train-rmse:8.40154      valid-rmse:8.30833      train-r2:0.57382
valid-r2:0.50931
[360]   train-rmse:8.37956      valid-rmse:8.30414      train-r2:0.57605
valid-r2:0.50980
[370]   train-rmse:8.34886      valid-rmse:8.30262      train-r2:0.57915
valid-r2:0.50998
[380]   train-rmse:8.32188      valid-rmse:8.30416      train-r2:0.58187
valid-r2:0.50980
[390]   train-rmse:8.28929      valid-rmse:8.30400      train-r2:0.58514
valid-r2:0.50982
[400]   train-rmse:8.26299      valid-rmse:8.30126      train-r2:0.58776
valid-r2:0.51014
[410]   train-rmse:8.23729      valid-rmse:8.30153      train-r2:0.59032
valid-r2:0.51011
[420]   train-rmse:8.21368      valid-rmse:8.30054      train-r2:0.59267
valid-r2:0.51023
[430]   train-rmse:8.18400      valid-rmse:8.30051      train-r2:0.59561
valid-r2:0.51023
[440]   train-rmse:8.16538      valid-rmse:8.30541      train-r2:0.59745
valid-r2:0.50965
[450]   train-rmse:8.13921      valid-rmse:8.30286      train-r2:0.60002
valid-r2:0.50995
[460]   train-rmse:8.11800      valid-rmse:8.30342      train-r2:0.60211
valid-r2:0.50989
Stopping. Best iteration:
[415]   train-rmse:8.22393      valid-rmse:8.29841      train-r2:0.59165
valid-r2:0.51048
```

[14]:

```python
# Step12: Predict your test_df values using xgboost

p_test = clf.predict(d_test)

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

sub.head()
```

```
[14]:    ID           y
      0   1   83.118958
      1   2   97.624886
      2   3   83.798798
      3   4   77.174225
      4   5  112.502441
```

```
[15]: sub.head()

      #######################################################################
      '''                          End                         '''
      #######################################################################
```

```
[15]: '                          End                         '
```

```
[ ]:
```