Project 1: Mercedes-Benz Greener Manufacturing

by SHAKTI NATH SAINI

In [1]:

```
####################################################################
''' Step 1: Import the required libraries'''
####################################################################

import numpy as np
import pandas as pd

# for dimensionality reduction
from sklearn.decomposition import PCA
```

In [2]:

```
import warnings
warnings.filterwarnings("ignore")
```

In [3]:

```
####################################################################
''' Step 2: Read the data from train.csv'''
####################################################################

df_train = pd.read_csv(r'C:\Users\Shakti\Documents\Python Scripts\Projects\Mercedes-Benz Greener Manufacturing\Dataset\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
```

Out[3]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	0	130.81	k	v	at	a	d	u	j	o	...	0	0	1	0	0	0	0	0	0	0
1	6	88.53	k	t	av	e	d	y	l	o	...	1	0	0	0	0	0	0	0	0	0
2	7	76.26	az	w	n	c	d	x	j	x	...	0	0	0	0	0	0	1	0	0	0
3	9	80.62	az	t	n	f	d	x	l	e	...	0	0	0	0	0	0	0	0	0	0
4	13	78.02	az	v	n	f	d	h	d	n	...	0	0	0	0	0	0	0	0	0	0

5 rows × 378 columns

In [4]:

```
####################################################################
''' Step 3: 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
```

In [5]:

```
####################################################################
''' Step 4: 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:
```

Out[5]:

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

In [6]:

```
####################################################################
''' Step 5: 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']
```

In [7]:

```
####################################################################
''' Step 6: Read the test.csv data'''
####################################################################


df_test = pd.read_csv(r'C:\Users\Shakti\Documents\Python Scripts\Projects\Mercedes-Benz Greener Manufacturing\Dataset\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]
```

In [8]:

```
####################################################################
'''  Step 7: 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()
```

Out[8]:

	X370	X177	X151	X346	X28	X101	X368	X205	X227	X361	...	X30	X165	X283	X138	X231	X191	X213	X114	X258	X226
0	0	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	1	0	1	0	1	...	0	1	0	0	0	0	0	0	0	0
2	0	0	0	0	1	1	0	1	0	1	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	0	1	0	1	...	0	0	0	0	0	0	0	1	0	0
4	0	0	0	0	1	1	0	1	0	1	...	0	0	0	0	0	0	0	1	0	0

5 rows × 376 columns

In [9]:

```
####################################################################
''' Step 7: 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
```

In [10]:

```
####################################################################
''' Step 9: Make sure the data is now changed into numericals'''
####################################################################


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

```
Feature types:
```

Out[10]:

```
int64    376
dtype: int64
```

In [11]:

```
####################################################################
''' 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)
```

In [12]:

```
####################################################################
''' Step 11: 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)
```

```
[21:43:48] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/objective/regression_obj.cu:171: reg:linear is now deprecated in favor of reg:squared
error.
[0]     train-rmse:99.14835     train-r2:-58.35295      valid-rmse:98.26297     valid-r2:-67.63754
[10]    train-rmse:81.27651     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:54.86911     train-r2:-17.17722      valid-rmse:53.89136     valid-r2:-19.64525
[40]    train-rmse:45.24709     train-r2:-11.36097      valid-rmse:44.22322     valid-r2:-12.90218
[50]    train-rmse:37.44853     train-r2:-7.46723       valid-rmse:36.37614     valid-r2:-8.40622
[60]    train-rmse:31.14585     train-r2:-4.85695       valid-rmse:30.02252     valid-r2:-5.40732
[70]    train-rmse:26.08417     train-r2:-3.10795       valid-rmse:24.91497     valid-r2:-3.41268
[80]    train-rmse:22.04313     train-r2:-1.93371       valid-rmse:20.83055     valid-r2:-2.08449
[90]    train-rmse:18.84671     train-r2:-1.14458       valid-rmse:17.59679     valid-r2:-1.20115
[100]   train-rmse:16.33278     train-r2:-0.61062       valid-rmse:15.07888     valid-r2:-0.61629
[110]   train-rmse:14.39756     train-r2:-0.25155       valid-rmse:13.14822     valid-r2:-0.22890
[120]   train-rmse:12.92783     train-r2:-0.00907       valid-rmse:11.69290     valid-r2:0.02809
[130]   train-rmse:11.80680     train-r2:0.15834        valid-rmse:10.61692     valid-r2:0.19873
[140]   train-rmse:10.98472     train-r2:0.27147        valid-rmse:9.85685      valid-r2:0.30935
[150]   train-rmse:10.37312     train-r2:0.35033        valid-rmse:9.32088      valid-r2:0.38242
[160]   train-rmse:9.91865      train-r2:0.40601        valid-rmse:8.95928      valid-r2:0.42940
[170]   train-rmse:9.58084      train-r2:0.44578        valid-rmse:8.71328      valid-r2:0.46031
[180]   train-rmse:9.33902      train-r2:0.47341        valid-rmse:8.55257      valid-r2:0.48003
[190]   train-rmse:9.15109      train-r2:0.49439        valid-rmse:8.44777      valid-r2:0.49270
[200]   train-rmse:9.00817      train-r2:0.51006        valid-rmse:8.38791      valid-r2:0.49986
[210]   train-rmse:8.90467      train-r2:0.52125        valid-rmse:8.34975      valid-r2:0.50440
[220]   train-rmse:8.82894      train-r2:0.52936        valid-rmse:8.31976      valid-r2:0.50796
[230]   train-rmse:8.76702      train-r2:0.53594        valid-rmse:8.30397      valid-r2:0.50982
[240]   train-rmse:8.72161      train-r2:0.54073        valid-rmse:8.29916      valid-r2:0.51039
[250]   train-rmse:8.67967      train-r2:0.54514        valid-rmse:8.29209      valid-r2:0.51122
[260]   train-rmse:8.64575      train-r2:0.54869        valid-rmse:8.28976      valid-r2:0.51150
[270]   train-rmse:8.61264      train-r2:0.55214        valid-rmse:8.28928      valid-r2:0.51155
[280]   train-rmse:8.58157      train-r2:0.55536        valid-rmse:8.28961      valid-r2:0.51152
[290]   train-rmse:8.55609      train-r2:0.55800        valid-rmse:8.28593      valid-r2:0.51195
[300]   train-rmse:8.53319      train-r2:0.56036        valid-rmse:8.28463      valid-r2:0.51210
[310]   train-rmse:8.50275      train-r2:0.56349        valid-rmse:8.28408      valid-r2:0.51217
[320]   train-rmse:8.48186      train-r2:0.56564        valid-rmse:8.28211      valid-r2:0.51240
[330]   train-rmse:8.45378      train-r2:0.56851        valid-rmse:8.27860      valid-r2:0.51281
[340]   train-rmse:8.42514      train-r2:0.57143        valid-rmse:8.27825      valid-r2:0.51285
[350]   train-rmse:8.40047      train-r2:0.57393        valid-rmse:8.28035      valid-r2:0.51261
[360]   train-rmse:8.37167      train-r2:0.57685        valid-rmse:8.27932      valid-r2:0.51273
[370]   train-rmse:8.35221      train-r2:0.57881        valid-rmse:8.27776      valid-r2:0.51291
[380]   train-rmse:8.33159      train-r2:0.58089        valid-rmse:8.27285      valid-r2:0.51349
[390]   train-rmse:8.30856      train-r2:0.58320        valid-rmse:8.27193      valid-r2:0.51360
[400]   train-rmse:8.28113      train-r2:0.58595        valid-rmse:8.26742      valid-r2:0.51413
[410]   train-rmse:8.25393      train-r2:0.58867        valid-rmse:8.26621      valid-r2:0.51427
[420]   train-rmse:8.22845      train-r2:0.59120        valid-rmse:8.26609      valid-r2:0.51428
[430]   train-rmse:8.20038      train-r2:0.59399        valid-rmse:8.26503      valid-r2:0.51441
[440]   train-rmse:8.17690      train-r2:0.59631        valid-rmse:8.26380      valid-r2:0.51455
[450]   train-rmse:8.15205      train-r2:0.59876        valid-rmse:8.26049      valid-r2:0.51494
[460]   train-rmse:8.12528      train-r2:0.60139        valid-rmse:8.26295      valid-r2:0.51465
[470]   train-rmse:8.10488      train-r2:0.60339        valid-rmse:8.26308      valid-r2:0.51464
[480]   train-rmse:8.08610      train-r2:0.60523        valid-rmse:8.26411      valid-r2:0.51452
[490]   train-rmse:8.06069      train-r2:0.60770        valid-rmse:8.26696      valid-r2:0.51418
[499]   train-rmse:8.04085      train-r2:0.60963        valid-rmse:8.26537      valid-r2:0.51437
```

In [13]:

```
####################################################################
''' Step 12: 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()
```

Out[13]:

	ID	y
0	1	82.949509
1	2	97.652306
2	3	83.115051
3	4	76.980057
4	5	112.474335

In [14]:

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

Out[14]:

```
'                          End                        '
```