Mercedes-Benz Greener Manufacturing

In [3]:
```python
# 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
```

In [4]:
```python
# 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
```

Out[4]:

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

5 rows × 378 columns

In [5]:
```python
# 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
```

In [6]:
```python
# 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:
```

Out[6]:
```
int64     368
object      8
dtype: int64
```

In [7]:
```python
# 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', 'X29
3', 'X297', 'X330', 'X347']
Categorical features: ['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8']
```

In [8]:
```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]
```

In [9]:
```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
```

In [10]:
```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()
```

```
<ipython-input-10-9fdc2c8730c7>: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/use
r_guide/indexing.html#returning-a-view-versus-a-copy
  x_train[column] = x_train[column].apply(mapper)
<ipython-input-10-9fdc2c8730c7>: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/use
r_guide/indexing.html#returning-a-view-versus-a-copy
  x_test[column] = x_test[column].apply(mapper)
```

Out[10]:

	X329	X308	X190	X268	X97	X84	X372	X10	X275	X132	...	X332	X124	X106	X213	X86	X302
0	1	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	1	1	...	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	1	...	0	0	0	0	0	0
3	0	0	0	0	0	1	1	0	0	1	...	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0

5 rows × 376 columns

In [11]:
```python
# Step9: Make sure the data is now changed into numericals

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

```
Feature types:
```

Out[11]:
```
int64    376
dtype: int64
```

In [12]:
```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)
```

In [13]:
```python
#conda install -c anaconda py-xgboost
```

In [14]:
```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)
```

```
[15:34:12] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/o
bjective/regression_obj.cu:170: reg:linear is now deprecated in favor of reg:squareder
ror.
[0]	train-rmse:99.14835	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:54.86956	train-r2:-17.17752	valid-rmse:53.88973	valid-
r2:-19.64401
[40]	train-rmse:45.24491	train-r2:-11.35979	valid-rmse:44.21970	valid-
r2:-12.89996
[50]	train-rmse:37.44729	train-r2:-7.46666	valid-rmse:36.37237	valid-
r2:-8.40428
[60]	train-rmse:31.14748	train-r2:-4.85757	valid-rmse:30.01874	valid-
r2:-5.40570
[70]	train-rmse:26.08660	train-r2:-3.10872	valid-rmse:24.90890	valid-
r2:-3.41053
[80]	train-rmse:22.04638	train-r2:-1.93458	valid-rmse:20.83274	valid-
r2:-2.08514
[90]	train-rmse:18.84403	train-r2:-1.14397	valid-rmse:17.60545	valid-
r2:-1.20331
[100]	train-rmse:16.33631	train-r2:-0.61131	valid-rmse:15.09092	valid-
r2:-0.61888
[110]	train-rmse:14.40308	train-r2:-0.25251	valid-rmse:13.15803	valid-
r2:-0.23073
[120]	train-rmse:12.92889	train-r2:-0.00924	valid-rmse:11.70481	valid-
r2:0.02611
[130]	train-rmse:11.81872	train-r2:0.15664	valid-rmse:10.63299	valid-
r2:0.19630
[140]	train-rmse:10.98615	train-r2:0.27128	valid-rmse:9.86614	valid-
r2:0.30805
[150]	train-rmse:10.37659	train-r2:0.34990	valid-rmse:9.33151	valid-
r2:0.38101
[160]	train-rmse:9.92954	train-r2:0.40471	valid-rmse:8.96882	valid-
r2:0.42819
[170]	train-rmse:9.59170	train-r2:0.44453	valid-rmse:8.72728	valid-
r2:0.45857
[180]	train-rmse:9.34758	train-r2:0.47244	valid-rmse:8.56733	valid-
r2:0.47824
[190]	train-rmse:9.15940	train-r2:0.49347	valid-rmse:8.46174	valid-
r2:0.49102
[200]	train-rmse:9.01791	train-r2:0.50900	valid-rmse:8.39616	valid-
r2:0.49888
[210]	train-rmse:8.91655	train-r2:0.51997	valid-rmse:8.35924	valid-
r2:0.50328
[220]	train-rmse:8.84092	train-r2:0.52808	valid-rmse:8.33544	valid-
r2:0.50610
[230]	train-rmse:8.77850	train-r2:0.53472	valid-rmse:8.32000	valid-
r2:0.50793
[240]	train-rmse:8.73145	train-r2:0.53970	valid-rmse:8.31477	valid-
r2:0.50855
[250]	train-rmse:8.69352	train-r2:0.54369	valid-rmse:8.31109	valid-
r2:0.50898
[260]	train-rmse:8.65809	train-r2:0.54740	valid-rmse:8.31127	valid-
r2:0.50896
[270]	train-rmse:8.62147	train-r2:0.55122	valid-rmse:8.30918	valid-
r2:0.50921
[280]	train-rmse:8.59397	train-r2:0.55408	valid-rmse:8.31045	valid-
r2:0.50906
[290]	train-rmse:8.56569	train-r2:0.55701	valid-rmse:8.31196	valid-
r2:0.50888
[300]	train-rmse:8.54080	train-r2:0.55958	valid-rmse:8.30835	valid-
r2:0.50931
[310]	train-rmse:8.51838	train-r2:0.56189	valid-rmse:8.30819	valid-
r2:0.50932
[320]	train-rmse:8.49085	train-r2:0.56471	valid-rmse:8.30920	valid-
r2:0.50921
[330]	train-rmse:8.46780	train-r2:0.56707	valid-rmse:8.31091	valid-
r2:0.50900
[340]	train-rmse:8.44368	train-r2:0.56954	valid-rmse:8.31012	valid-
r2:0.50910
[350]	train-rmse:8.42110	train-r2:0.57184	valid-rmse:8.30611	valid-
r2:0.50957
[360]	train-rmse:8.39358	train-r2:0.57463	valid-rmse:8.30269	valid-
r2:0.50997
[370]	train-rmse:8.37054	train-r2:0.57696	valid-rmse:8.30065	valid-
r2:0.51021
[380]	train-rmse:8.34589	train-r2:0.57945	valid-rmse:8.29848	valid-
r2:0.51047
[390]	train-rmse:8.32111	train-r2:0.58195	valid-rmse:8.29416	valid-
r2:0.51098
[400]	train-rmse:8.29688	train-r2:0.58437	valid-rmse:8.29347	valid-
r2:0.51106
[410]	train-rmse:8.27725	train-r2:0.58634	valid-rmse:8.29385	valid-
r2:0.51102
[420]	train-rmse:8.25156	train-r2:0.58890	valid-rmse:8.29428	valid-
r2:0.51097
[430]	train-rmse:8.22105	train-r2:0.59194	valid-rmse:8.29203	valid-
r2:0.51123
[440]	train-rmse:8.20270	train-r2:0.59376	valid-rmse:8.29029	valid-
r2:0.51144
[450]	train-rmse:8.17461	train-r2:0.59654	valid-rmse:8.28972	valid-
r2:0.51150
[460]	train-rmse:8.14983	train-r2:0.59898	valid-rmse:8.28852	valid-
r2:0.51164
[470]	train-rmse:8.12920	train-r2:0.60101	valid-rmse:8.28771	valid-
r2:0.51174
[480]	train-rmse:8.11243	train-r2:0.60265	valid-rmse:8.28886	valid-
r2:0.51160
[490]	train-rmse:8.08963	train-r2:0.60488	valid-rmse:8.28657	valid-
r2:0.51187
[500]	train-rmse:8.06128	train-r2:0.60764	valid-rmse:8.28504	valid-
r2:0.51206
[510]	train-rmse:8.03752	train-r2:0.60995	valid-rmse:8.28459	valid-
r2:0.51211
[520]	train-rmse:8.00866	train-r2:0.61275	valid-rmse:8.28543	valid-
r2:0.51201
[530]	train-rmse:7.98790	train-r2:0.61476	valid-rmse:8.28685	valid-
r2:0.51184
[540]	train-rmse:7.96921	train-r2:0.61656	valid-rmse:8.28644	valid-
r2:0.51189
[550]	train-rmse:7.94946	train-r2:0.61845	valid-rmse:8.28512	valid-
r2:0.51204
[553]	train-rmse:7.94079	train-r2:0.61929	valid-rmse:8.28567	valid-
r2:0.51198
```

In [16]:
```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()
```

Out[16]:

	ID	y
0	1	82.504364
1	2	97.501617
2	3	82.758621
3	4	76.961739
4	5	112.794571

In [ ]: