Machine Learning Project - MecedesBenz Manufacturing

1. Import Libaray

In [1]:

```
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
%matplotlib inline
```

2. Import Training & Test Dataset

In [2]:

```
df_train = pd.read_csv('train.csv', header=0)
df_train.head()
```

Out[2]:

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

5 rows × 378 columns

In [3]:

```
df_train.shape
```

Out[3]:

```
(4209, 378)
```

In [40]:

```
df_train['y'].describe()
```

Out[40]:

```
count    4209.000000
mean      100.669318
std        12.679381
min        72.110000
25%        90.820000
50%        99.150000
75%       109.010000
max       265.320000
Name: y, dtype: float64
```

In [4]:

```
df_test = pd.read_csv('test.csv', header=0)
df_test.head()
```

Out[4]:

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X3
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	

5 rows × 377 columns

In [5]:

```
df_test.shape
```

Out[5]:

```
(4209, 377)
```

3. Data Cleaning

3.1. Remove Non-Relevent Features like Y & ID

In [6]:

```
X_train= df_train.drop(['y','ID'], axis=1)
X_test = df_test.drop(['ID'], axis = 1)
y_label = df_train['y']
print('Training Set: {}'.format(X_train.shape))
print('Training Label: {}'.format(y_label.shape))
print('Test Set: {}'.format(X_test.shape))
```

```
Training Set: (4209, 376)
Training Label: (4209,)
Test Set: (4209, 376)
```

3.2. Checking Missing Value Or Not

In [7]:

```
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")
```

In [8]:

```
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
```

3.3. Features With Unique Values Will Be Removed from Training Set & Test Set

In [9]:

```
cateCols = []
oneUniqueCols = []
for c in X_train.columns:
    cardinality = len(np.unique(X_train[c]))
    if cardinality == 1:
        oneUniqueCols.append(c)
    if X_train[c].dtype == 'object':
        cateCols.append(c)
```

In [10]:

```
oneUniqueCols
```

Out[10]:

```
['X11',
 'X93',
 'X107',
 'X233',
 'X235',
 'X268',
 'X289',
 'X290',
 'X293',
 'X297',
 'X330',
 'X347']
```

In [11]:

```
X_train_clean = X_train.drop(oneUniqueCols, axis = 1)
X_train_clean.head()
```

Out[11]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X12	...	X375	X376	X377	X378	X379	X380
0	k	v	at	a	d	u	j	o	0	0	...	0	0	1	0	0	0
1	k	t	av	e	d	y	l	o	0	0	...	1	0	0	0	0	0
2	az	w	n	c	d	x	j	x	0	0	...	0	0	0	0	0	0
3	az	t	n	f	d	x	l	e	0	0	...	0	0	0	0	0	0
4	az	v	n	f	d	h	d	n	0	0	...	0	0	0	0	0	0

5 rows × 364 columns

In [12]:

```
X_test_clean = X_test.drop(oneUniqueCols, axis = 1)
X_test_clean.head()
```

Out[12]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X12	...	X375	X376	X377	X378	X379	X380
0	az	v	n	f	d	t	a	w	0	0	...	0	0	0	1	0	0
1	t	b	ai	a	d	b	g	y	0	0	...	0	0	1	0	0	0
2	az	v	as	f	d	a	j	j	0	0	...	0	0	0	1	0	0
3	az	l	n	f	d	z	l	n	0	0	...	0	0	0	1	0	0
4	w	s	as	c	d	y	i	m	0	0	...	1	0	0	0	0	0

5 rows × 364 columns

4. PCA Feature Selection

4.1. Data Type Analyst

In [13]:

```
X_train_clean.dtypes.value_counts()
```

Out[13]:

```
int64     356
object      8
dtype: int64
```

4.1.1 Features With Categorical Data

In [14]:

```
cateCols
```

Out[14]:

```
['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8']
```

4.1.2. Features With Binary Data

In [15]:

```
X_train_clean.select_dtypes(include=[np.number]).head()
```

Out[15]:

	X10	X12	X13	X14	X15	X16	X17	X18	X19	X20	...	X375	X376	X377	X378	X379
0	0	0	1	0	0	0	0	1	0	0	...	0	0	1	0	0
1	0	0	0	0	0	0	0	1	0	0	...	1	0	0	0	0
2	0	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0

5 rows × 356 columns

In [16]:

```
np.unique(X_train_clean.select_dtypes(include=[np.number]))
```

Out[16]:

```
array([0, 1])
```

4.2. Encode Categorical Data

In order to implement PCA, Categorical data need to be encode

In [17]:

```
X_train_clean_encode = X_train_clean
X_test_clean_encode = X_test_clean
```

In [18]:

```
from sklearn.preprocessing import LabelEncoder
trainLabEncoder = LabelEncoder()
testLabEncoder = LabelEncoder()
```

In [19]:

```
for c in cateCols:
    X_train_clean_encode[c] = trainLabEncoder.fit_transform(X_train_clean_encode[c])
    X_test_clean_encode[c] = testLabEncoder.fit_transform(X_test_clean_encode[c])
```

In [20]:

```
X_train_clean_encode.dtypes.value_counts()
```

Out[20]:

```
int64    364
dtype: int64
```

In [21]:

```
X_train_clean_encode.head()
```

Out[21]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X12	...	X375	X376	X377	X378	X379	X380
0	32	23	17	0	3	24	9	14	0	0	...	0	0	1	0	0	0
1	32	21	19	4	3	28	11	14	0	0	...	1	0	0	0	0	0
2	20	24	34	2	3	27	9	23	0	0	...	0	0	0	0	0	0
3	20	21	34	5	3	27	11	4	0	0	...	0	0	0	0	0	0
4	20	23	34	5	3	12	3	13	0	0	...	0	0	0	0	0	0

5 rows × 364 columns

In [22]:

```
X_test_clean_encode.dtypes.value_counts()
```

Out[22]:

```
int64    364
dtype: int64
```

In [23]:

```
X_test_clean_encode.head()
```

Out[23]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X12	...	X375	X376	X377	X378	X379	X380
0	21	23	34	5	3	26	0	22	0	0	...	0	0	0	1	0	0
1	42	3	8	0	3	9	6	24	0	0	...	0	0	1	0	0	0
2	21	23	17	5	3	0	9	9	0	0	...	0	0	0	1	0	0
3	21	13	34	5	3	31	11	13	0	0	...	0	0	0	1	0	0
4	45	20	17	2	3	30	8	12	0	0	...	1	0	0	0	0	0

5 rows × 364 columns

4.3. PCA on Training & Test Data Set

In [24]:

```
n_comp = 12
pca = PCA(n_components=n_comp, random_state=420)
```

In [25]:

```
trainPCA = pca.fit_transform(X_train_clean_encode)
trainPCA.shape
```

Out[25]:

```
(4209, 12)
```

In [26]:

```
testPCA = pca.transform(X_test_clean_encode)
testPCA.shape
```

Out[26]:

```
(4209, 12)
```

5. Train XGB model

5.1. Import XGB Lib

In [27]:

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

5.2. Split Training Data Set

In [28]:

```
X_train_split, X_valid_split, y_train_split,y_valid_split = train_test_split(trainPCA,
y_label, test_size = 0.2, random_state=4242)
```

5.3. Store data in DMatrix

In [29]:

```
xgb_X_train_split = xgb.DMatrix(X_train_split, label=y_train_split)
xgb_X_valid_split = xgb.DMatrix(X_valid_split, label=y_valid_split)
```

```
/opt/anaconda3/lib/python3.7/site-packages/xgboost/core.py:587: FutureWarn
ing: Series.base is deprecated and will be removed in a future version
  if getattr(data, 'base', None) is not None and \
```

5.4. Specify XGB Parameters

In [30]:

```
params = {}
params['objective'] = 'reg:linear'
params['eta'] = 0.005
params['max_depth'] = 6
params['n_trees'] = 500
params['subsample'] = 0.5
params['eval_metric'] = 'rmse'
params['base_score'] = np.mean(y_label)
params['silent'] = 1
```

In [31]:

```
evallist = [(xgb_X_train_split, 'train'), (xgb_X_valid_split, 'valid')]
```

In [32]:

```
def xgb_r2_score(preds, dtrain):
    labels = dtrain.get_label()
    return 'r2', r2_score(labels, preds)
```

5.5. Train XGB Model

In [33]:

```
model = xgb.train(params,xgb_X_train_split,1000,evallist,feval=xgb_r2_score,maximize=True,verbose_eval=10)
```

```
[0]	train-rmse:12.845	valid-rmse:11.8462	train-r2:0.003816	valid-r2:0.002443
[10]	train-rmse:12.5915	valid-rmse:11.5659	train-r2:0.042754	valid-r2:0.049089
[20]	train-rmse:12.3529	valid-rmse:11.3078	train-r2:0.078681	valid-r2:0.091054
[30]	train-rmse:12.1355	valid-rmse:11.0665	train-r2:0.110832	valid-r2:0.129436
[40]	train-rmse:11.9269	valid-rmse:10.8405	train-r2:0.141132	valid-r2:0.164621
[50]	train-rmse:11.7325	valid-rmse:10.637	train-r2:0.168903	valid-r2:0.195699
[60]	train-rmse:11.553	valid-rmse:10.4452	train-r2:0.194138	valid-r2:0.224435
[70]	train-rmse:11.3856	valid-rmse:10.2716	train-r2:0.217324	valid-r2:0.250012
[80]	train-rmse:11.2277	valid-rmse:10.0996	train-r2:0.238875	valid-r2:0.274919
[90]	train-rmse:11.0785	valid-rmse:9.94822	train-r2:0.258975	valid-r2:0.296486
[100]	train-rmse:10.938	valid-rmse:9.80937	train-r2:0.277647	valid-r2:0.315987
[110]	train-rmse:10.8016	valid-rmse:9.67603	train-r2:0.295555	valid-r2:0.334456
[120]	train-rmse:10.6775	valid-rmse:9.55711	train-r2:0.311642	valid-r2:0.350716
[130]	train-rmse:10.5606	valid-rmse:9.44737	train-r2:0.326638	valid-r2:0.365541
[140]	train-rmse:10.4494	valid-rmse:9.3395	train-r2:0.340749	valid-r2:0.379946
[150]	train-rmse:10.3431	valid-rmse:9.23764	train-r2:0.354091	valid-r2:0.393398
[160]	train-rmse:10.2469	valid-rmse:9.15172	train-r2:0.366046	valid-r2:0.404629
[170]	train-rmse:10.1509	valid-rmse:9.06705	train-r2:0.377865	valid-r2:0.415595
[180]	train-rmse:10.0576	valid-rmse:8.98601	train-r2:0.389259	valid-r2:0.425995
[190]	train-rmse:9.97109	valid-rmse:8.91254	train-r2:0.399717	valid-r2:0.435342
[200]	train-rmse:9.88706	valid-rmse:8.84202	train-r2:0.409791	valid-r2:0.444242
[210]	train-rmse:9.81209	valid-rmse:8.77838	train-r2:0.418708	valid-r2:0.452214
[220]	train-rmse:9.73503	valid-rmse:8.71936	train-r2:0.427802	valid-r2:0.459556
[230]	train-rmse:9.66744	valid-rmse:8.67007	train-r2:0.435721	valid-r2:0.465648
[240]	train-rmse:9.6043	valid-rmse:8.62749	train-r2:0.443067	valid-r2:0.470885
[250]	train-rmse:9.54289	valid-rmse:8.58739	train-r2:0.450166	valid-r2:0.475791
[260]	train-rmse:9.47675	valid-rmse:8.54475	train-r2:0.457762	valid-r2:0.480985
[270]	train-rmse:9.41585	valid-rmse:8.50679	train-r2:0.464708	valid-r2:0.485585
[280]	train-rmse:9.35931	valid-rmse:8.4717	train-r2:0.471118	valid-r2:0.489821
[290]	train-rmse:9.30068	valid-rmse:8.43769	train-r2:0.477724	valid-r2:0.493908
[300]	train-rmse:9.24547	valid-rmse:8.41128	train-r2:0.483905
```

```
valid-r2:0.497072
[310]   train-rmse:9.18894      valid-rmse:8.3845       train-r2:0.490197
valid-r2:0.500269
[320]   train-rmse:9.13911      valid-rmse:8.35269      train-r2:0.495712
valid-r2:0.504054
[330]   train-rmse:9.0899       valid-rmse:8.32949      train-r2:0.501127
valid-r2:0.506805
[340]   train-rmse:9.04112      valid-rmse:8.30771      train-r2:0.506468
valid-r2:0.50938
[350]   train-rmse:8.99067      valid-rmse:8.2891       train-r2:0.51196
valid-r2:0.511576
[360]   train-rmse:8.94112      valid-rmse:8.27254      train-r2:0.517324
valid-r2:0.513526
[370]   train-rmse:8.89839      valid-rmse:8.25555      train-r2:0.521927
valid-r2:0.515522
[380]   train-rmse:8.84918      valid-rmse:8.23919      train-r2:0.5272 va
lid-r2:0.517441
[390]   train-rmse:8.80791      valid-rmse:8.22365      train-r2:0.531599
valid-r2:0.519259
[400]   train-rmse:8.76909      valid-rmse:8.20863      train-r2:0.535719
valid-r2:0.521013
[410]   train-rmse:8.7329       valid-rmse:8.19518      train-r2:0.539544
valid-r2:0.522582
[420]   train-rmse:8.68955      valid-rmse:8.18239      train-r2:0.544104
valid-r2:0.524071
[430]   train-rmse:8.6503       valid-rmse:8.16941      train-r2:0.548213
valid-r2:0.52558
[440]   train-rmse:8.6143       valid-rmse:8.16254      train-r2:0.551966
valid-r2:0.526377
[450]   train-rmse:8.57552      valid-rmse:8.15608      train-r2:0.55599
valid-r2:0.527127
[460]   train-rmse:8.53868      valid-rmse:8.14953      train-r2:0.559797
valid-r2:0.527885
[470]   train-rmse:8.50647      valid-rmse:8.14124      train-r2:0.563112
valid-r2:0.528846
[480]   train-rmse:8.47029      valid-rmse:8.13251      train-r2:0.566821
valid-r2:0.529855
[490]   train-rmse:8.43403      valid-rmse:8.12301      train-r2:0.570521
valid-r2:0.530953
[500]   train-rmse:8.40197      valid-rmse:8.11828      train-r2:0.57378
valid-r2:0.5315
[510]   train-rmse:8.36928      valid-rmse:8.11139      train-r2:0.57709
valid-r2:0.532295
[520]   train-rmse:8.33616      valid-rmse:8.10313      train-r2:0.58043
valid-r2:0.533246
[530]   train-rmse:8.30549      valid-rmse:8.09816      train-r2:0.583513
valid-r2:0.533818
[540]   train-rmse:8.27495      valid-rmse:8.09369      train-r2:0.58657
valid-r2:0.534333
[550]   train-rmse:8.24329      valid-rmse:8.09017      train-r2:0.589727
valid-r2:0.534738
[560]   train-rmse:8.21783      valid-rmse:8.08563      train-r2:0.592258
valid-r2:0.53526
[570]   train-rmse:8.18836      valid-rmse:8.0811       train-r2:0.595177
valid-r2:0.535781
[580]   train-rmse:8.15661      valid-rmse:8.07827      train-r2:0.59831
valid-r2:0.536107
[590]   train-rmse:8.12677      valid-rmse:8.07254      train-r2:0.601244
valid-r2:0.536764
[600]   train-rmse:8.09919      valid-rmse:8.06654      train-r2:0.603946
valid-r2:0.537452
```

```
[610]   train-rmse:8.071        valid-rmse:8.06057      train-r2:0.606698
valid-r2:0.538137
[620]   train-rmse:8.04403      valid-rmse:8.05754      train-r2:0.609321
valid-r2:0.538484
[630]   train-rmse:8.01314      valid-rmse:8.05334      train-r2:0.612317
valid-r2:0.538965
[640]   train-rmse:7.99194      valid-rmse:8.05136      train-r2:0.614366
valid-r2:0.539191
[650]   train-rmse:7.96791      valid-rmse:8.0486       train-r2:0.616681
valid-r2:0.539507
[660]   train-rmse:7.94309      valid-rmse:8.04497      train-r2:0.619066
valid-r2:0.539923
[670]   train-rmse:7.92131      valid-rmse:8.03984      train-r2:0.621152
valid-r2:0.540509
[680]   train-rmse:7.90025      valid-rmse:8.04207      train-r2:0.623163
valid-r2:0.540254
[690]   train-rmse:7.87589      valid-rmse:8.04172      train-r2:0.625484
valid-r2:0.540294
[700]   train-rmse:7.84781      valid-rmse:8.03999      train-r2:0.628149
valid-r2:0.540492
[710]   train-rmse:7.81966      valid-rmse:8.04033      train-r2:0.630812
valid-r2:0.540453
[720]   train-rmse:7.79603      valid-rmse:8.03713      train-r2:0.63304
valid-r2:0.540819
[730]   train-rmse:7.77108      valid-rmse:8.0372       train-r2:0.635385
valid-r2:0.540811
[740]   train-rmse:7.74711      valid-rmse:8.03678      train-r2:0.637632
valid-r2:0.540859
[750]   train-rmse:7.72211      valid-rmse:8.03744      train-r2:0.639966
valid-r2:0.540784
[760]   train-rmse:7.70172      valid-rmse:8.03427      train-r2:0.641865
valid-r2:0.541146
[770]   train-rmse:7.68357      valid-rmse:8.03279      train-r2:0.643551
valid-r2:0.541315
[780]   train-rmse:7.66153      valid-rmse:8.03377      train-r2:0.645593
valid-r2:0.541203
[790]   train-rmse:7.64141      valid-rmse:8.03096      train-r2:0.647452
valid-r2:0.541524
[800]   train-rmse:7.61673      valid-rmse:8.03 train-r2:0.649725       va
lid-r2:0.541633
[810]   train-rmse:7.59877      valid-rmse:8.03091      train-r2:0.651375
valid-r2:0.541529
[820]   train-rmse:7.57794      valid-rmse:8.02898      train-r2:0.653284
valid-r2:0.541749
[830]   train-rmse:7.55482      valid-rmse:8.02459      train-r2:0.655397
valid-r2:0.542251
[840]   train-rmse:7.53085      valid-rmse:8.02394      train-r2:0.65758
valid-r2:0.542324
[850]   train-rmse:7.51177      valid-rmse:8.02274      train-r2:0.659313
valid-r2:0.542461
[860]   train-rmse:7.49514      valid-rmse:8.02534      train-r2:0.66082
valid-r2:0.542165
[870]   train-rmse:7.47659      valid-rmse:8.02289      train-r2:0.662496
valid-r2:0.542445
[880]   train-rmse:7.45615      valid-rmse:8.02152      train-r2:0.664339
valid-r2:0.542601
[890]   train-rmse:7.43299      valid-rmse:8.02123      train-r2:0.666421
valid-r2:0.542634
[900]   train-rmse:7.41652      valid-rmse:8.02087      train-r2:0.667898
valid-r2:0.542675
[910]   train-rmse:7.3992       valid-rmse:8.01885      train-r2:0.669447
```

```
valid-r2:0.542905
[920]	train-rmse:7.38058	valid-rmse:8.01834	train-r2:0.671109
valid-r2:0.542964
[930]	train-rmse:7.3623	valid-rmse:8.01483	train-r2:0.672736
valid-r2:0.543364
[940]	train-rmse:7.34098	valid-rmse:8.01376	train-r2:0.674629
valid-r2:0.543485
[950]	train-rmse:7.32291	valid-rmse:8.01126	train-r2:0.676229
valid-r2:0.54377
[960]	train-rmse:7.30382	valid-rmse:8.00967	train-r2:0.677914
valid-r2:0.543951
[970]	train-rmse:7.28536	valid-rmse:8.00703	train-r2:0.67954
valid-r2:0.544252
[980]	train-rmse:7.26904	valid-rmse:8.00731	train-r2:0.680974
valid-r2:0.544219
[990]	train-rmse:7.25269	valid-rmse:8.00748	train-r2:0.682408
valid-r2:0.544201
[999]	train-rmse:7.23772	valid-rmse:8.00635	train-r2:0.683718
valid-r2:0.544329
```

5.6. Predict with Test set

In [34]:

```
xgb_X_test = xgb.DMatrix(testPCA)
```

In [35]:

```
prediectY = model.predict(xgb_X_test)
prediectY
```

Out[35]:

```
array([ 79.10548,  94.82104,  81.91955, ..., 100.73042, 109.27812,
        95.47911], dtype=float32)
```

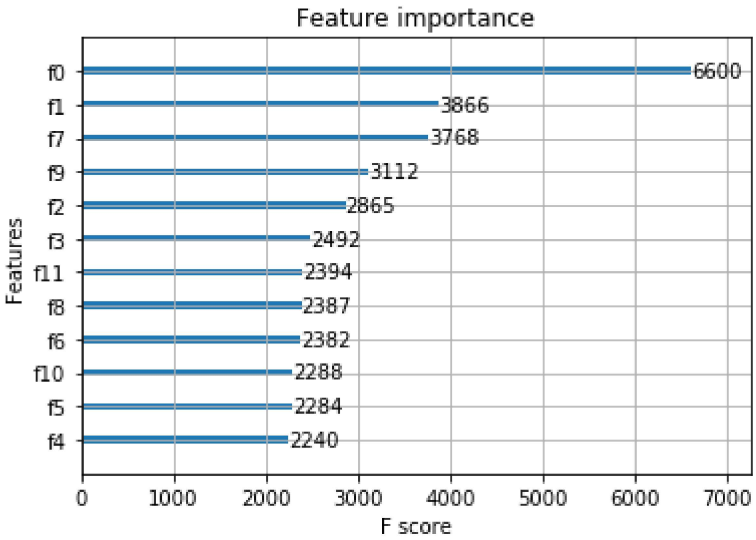
5.7. Plot Importance Features

In [36]:

```
xgb.plot_importance(model)
```

Out[36]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb97578fdd8>
```



In [37]:

```
r2_score(y_valid_split,model.predict(xgb.DMatrix(X_valid_split)))
```

Out[37]:

```
0.5443294860369894
```

In [ ]:

```

```