Mercedes Benz Project

February 6, 2023

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

[3]:
```
df_test = pd.read_csv('G:\\Machine Learning\\test.csv')
df_train = pd.read_csv('G:\\Machine Learning\\train.csv')
```

[4]:
```
print(df_test.shape)
print(df_train.shape)
```

```
(4209, 377)
(4209, 378)
```

[5]:
```
df_train.head()
```

[5]:

	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]

[6]:
```
df_train.dtypes
```

[6]:
```
ID        int64
y       float64
X0       object
```

```
X1       object
X2       object
         …
X380      int64
X382      int64
X383      int64
X384      int64
X385      int64
Length: 378, dtype: object
```

[7]:
```
df_train.describe()
```

[7]:

	ID	y	X10	X11	X12	\
count	4209.000000	4209.000000	4209.000000	4209.0	4209.000000	
mean	4205.960798	100.669318	0.013305	0.0	0.075077	
std	2437.608688	12.679381	0.114590	0.0	0.263547	
min	0.000000	72.110000	0.000000	0.0	0.000000	
25%	2095.000000	90.820000	0.000000	0.0	0.000000	
50%	4220.000000	99.150000	0.000000	0.0	0.000000	
75%	6314.000000	109.010000	0.000000	0.0	0.000000	
max	8417.000000	265.320000	1.000000	0.0	1.000000	

	X13	X14	X15	X16	X17	…	\
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	…	
mean	0.057971	0.428130	0.000475	0.002613	0.007603	…	
std	0.233716	0.494867	0.021796	0.051061	0.086872	…	
min	0.000000	0.000000	0.000000	0.000000	0.000000	…	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	…	
50%	0.000000	0.000000	0.000000	0.000000	0.000000	…	
75%	0.000000	1.000000	0.000000	0.000000	0.000000	…	
max	1.000000	1.000000	1.000000	1.000000	1.000000	…	

	X375	X376	X377	X378	X379	\
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	
mean	0.318841	0.057258	0.314802	0.020670	0.009503	
std	0.466082	0.232363	0.464492	0.142294	0.097033	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	0.000000	0.000000	
75%	1.000000	0.000000	1.000000	0.000000	0.000000	
max	1.000000	1.000000	1.000000	1.000000	1.000000	

	X380	X382	X383	X384	X385
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000
mean	0.008078	0.007603	0.001663	0.000475	0.001426
std	0.089524	0.086872	0.040752	0.021796	0.037734
min	0.000000	0.000000	0.000000	0.000000	0.000000

```
25%       0.000000     0.000000     0.000000     0.000000     0.000000
50%       0.000000     0.000000     0.000000     0.000000     0.000000
75%       0.000000     0.000000     0.000000     0.000000     0.000000
max       1.000000     1.000000     1.000000     1.000000     1.000000

[8 rows x 370 columns]
```

[8]:
```
train_data = np.var(df_train,axis = 0)
train_data
```

```
C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:3721:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise
TypeError.  Select only valid columns before calling the reduction.
  return var(axis=axis, dtype=dtype, out=out, ddof=ddof, **kwargs)
```

[8]:
```
ID      5.940524e+06
y       1.607285e+02
X10     1.312780e-02
X11     0.000000e+00
X12     6.944063e-02
            …
X380    8.012675e-03
X382    7.544954e-03
X383    1.660337e-03
X384    4.749465e-04
X385    1.423485e-03
Length: 370, dtype: float64
```

[9]:
```
test_data = np.var(df_test,axis = 0)
test_data
```

```
C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:3721:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise
TypeError.  Select only valid columns before calling the reduction.
  return var(axis=axis, dtype=dtype, out=out, ddof=ddof, **kwargs)
```

[9]:
```
ID      5.869917e+06
X10     1.864563e-02
X11     2.375297e-04
X12     6.883438e-02
X13     5.733136e-02
            …
X380    8.012675e-03
X382    8.713410e-03
X383    4.749465e-04
X384    7.122504e-04
```

```
X385    1.660337e-03
Length: 369, dtype: float64
```

[12]:
```
train_name=[]
for i in train_data.iteritems():
    if(i[1]==0):
        train_name.append(i[0])
        #print(i)
print(train_name)

test_name=[]
for i in test_data.iteritems():
    if(i[1]==0):
        test_name.append(i[0])
        #print(i)
print(test_name)
```

```
['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290', 'X293', 'X297',
'X330', 'X347']
['X257', 'X258', 'X295', 'X296', 'X369']
```

[13]:
```
df_train.drop (train_name,axis = 1 ,inplace = True)
df_train.drop(test_name ,axis = 1 ,inplace = True)
```

[14]:
```
df_test.drop(train_name,axis = 1,inplace = True)
df_test.drop(test_name,axis = 1,inplace = True)
```

[17]:
```
print(df_train.shape)
print(df_test.shape)
```

```
(4209, 361)
(4209, 360)
```

0.0.1 Check for null and unique values for dataset

[19]:
```
for i,j in zip (df_train.columns,df_train.isnull().sum()):
    if (j != 0):
        print (i)
```

[21]:
```
train_desc = df_train.describe(include='O')
df_train.describe(include='O')
```

[21]:

	X0	X1	X2	X3	X4	X5	X6	X8
count	4209	4209	4209	4209	4209	4209	4209	4209
unique	47	27	44	7	4	29	12	25
top	z	aa	as	c	d	w	g	j
freq	360	833	1659	1942	4205	231	1042	277

[23]:
```
train_desc.columns
```

[23]:
```
Index(['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8'], dtype='object')
```

[25]:
```
test_desc = df_test.describe(include = 'O')
df_train.describe(include='O')
```

[25]:

	X0	X1	X2	X3	X4	X5	X6	X8
count	4209	4209	4209	4209	4209	4209	4209	4209
unique	47	27	44	7	4	29	12	25
top	z	aa	as	c	d	w	g	j
freq	360	833	1659	1942	4205	231	1042	277

[27]:
```
test_desc.columns
```

[27]:
```
Index(['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8'], dtype='object')
```

[30]:
```
for i in ['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8']:
    print ('df_train')
    print(i,df_train[i].unique())
    print('df_test')
    print(i,df_test[i].unique())
```

```
df_train
X0 ['k' 'az' 't' 'al' 'o' 'w' 'j' 'h' 's' 'n' 'ay' 'f' 'x' 'y' 'aj' 'ak' 'am'
 'z' 'q' 'at' 'ap' 'v' 'af' 'a' 'e' 'ai' 'd' 'aq' 'c' 'aa' 'ba' 'as' 'i'
 'r' 'b' 'ax' 'bc' 'u' 'ad' 'au' 'm' 'l' 'aw' 'ao' 'ac' 'g' 'ab']
df_test
X0 ['az' 't' 'w' 'y' 'x' 'f' 'ap' 'o' 'ay' 'al' 'h' 'z' 'aj' 'd' 'v' 'ak'
 'ba' 'n' 'j' 's' 'af' 'ax' 'at' 'aq' 'av' 'm' 'k' 'a' 'e' 'ai' 'i' 'ag'
 'b' 'am' 'aw' 'as' 'r' 'ao' 'u' 'l' 'c' 'ad' 'au' 'bc' 'g' 'an' 'ae' 'p'
 'bb']
df_train
X1 ['v' 't' 'w' 'b' 'r' 'l' 's' 'aa' 'c' 'a' 'e' 'h' 'z' 'j' 'o' 'u' 'p' 'n'
 'i' 'y' 'd' 'f' 'm' 'k' 'g' 'q' 'ab']
df_test
X1 ['v' 'b' 'l' 's' 'aa' 'r' 'a' 'i' 'p' 'c' 'o' 'm' 'z' 'e' 'h' 'w' 'g' 'k'
 'y' 't' 'u' 'd' 'j' 'q' 'n' 'f' 'ab']
df_train
X2 ['at' 'av' 'n' 'e' 'as' 'aq' 'r' 'ai' 'ak' 'm' 'a' 'k' 'ae' 's' 'f' 'd'
 'ag' 'ay' 'ac' 'ap' 'g' 'i' 'aw' 'y' 'b' 'ao' 'al' 'h' 'x' 'au' 't' 'an'
 'z' 'ah' 'p' 'am' 'j' 'q' 'af' 'l' 'aa' 'c' 'o' 'ar']
df_test
X2 ['n' 'ai' 'as' 'ae' 's' 'b' 'e' 'ak' 'm' 'a' 'aq' 'ag' 'r' 'k' 'aj' 'ay'
 'ao' 'an' 'ac' 'af' 'ax' 'h' 'i' 'f' 'ap' 'p' 'au' 't' 'z' 'y' 'aw' 'd'
 'at' 'g' 'am' 'j' 'x' 'ab' 'w' 'q' 'ah' 'ad' 'al' 'av' 'u']
df_train
```

```
X3 ['a' 'e' 'c' 'f' 'd' 'b' 'g']
df_test
X3 ['f' 'a' 'c' 'e' 'd' 'g' 'b']
df_train
X4 ['d' 'b' 'c' 'a']
df_test
X4 ['d' 'b' 'a' 'c']
df_train
X5 ['u' 'y' 'x' 'h' 'g' 'f' 'j' 'i' 'd' 'c' 'af' 'ag' 'ab' 'ac' 'ad' 'ae'
 'ah' 'l' 'k' 'n' 'm' 'p' 'q' 's' 'r' 'v' 'w' 'o' 'aa']
df_test
X5 ['t' 'b' 'a' 'z' 'y' 'x' 'h' 'g' 'f' 'j' 'i' 'd' 'c' 'af' 'ag' 'ab' 'ac'
 'ad' 'ae' 'ah' 'l' 'k' 'n' 'm' 'p' 'q' 's' 'r' 'v' 'w' 'o' 'aa']
df_train
X6 ['j' 'l' 'd' 'h' 'i' 'a' 'g' 'c' 'k' 'e' 'f' 'b']
df_test
X6 ['a' 'g' 'j' 'l' 'i' 'd' 'f' 'h' 'c' 'k' 'e' 'b']
df_train
X8 ['o' 'x' 'e' 'n' 's' 'a' 'h' 'p' 'm' 'k' 'd' 'i' 'v' 'j' 'b' 'q' 'w' 'g'
 'y' 'l' 'f' 'u' 'r' 't' 'c']
df_test
X8 ['w' 'y' 'j' 'n' 'm' 's' 'a' 'v' 'r' 'o' 't' 'h' 'c' 'k' 'p' 'u' 'd' 'g'
 'b' 'q' 'e' 'l' 'f' 'i' 'x']
```

0.0.2 Apply label encoder.

[31]:
```
df_train_x = df_train.drop(['ID','y'],axis = 1)
df_train_y = df_train ['y']
```

[32]:
```
print(df_train_x.shape)
print(df_train_y.shape)
```

```
(4209, 359)
(4209,)
```

[34]:
```
df_test_x = df_test.drop(['ID'],axis = 1)
```

[39]:
```
print(df_test_x.shape)
print(df_test_x.shape)
```

```
(4209, 359)
(4209, 359)
```

[40]:
```
from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
for i in train_desc.columns:
    df_train_x[i]=le.fit_transform(df_train_x[i])
for i in test_desc.columns:
```

```
    df_test_x[i]=le.fit_transform(df_test_x[i])
```

[41]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(df_train_x,df_train_y, test_size=0.
 ↪25, random_state=10)
print(xtrain.shape,ytrain.shape)
print(xtest.shape,ytest.shape)
```

```
(3156, 359) (3156,)
(1053, 359) (1053,)
```

0.0.3 Perform dimensionality reduction.

[44]:
```
from sklearn.decomposition import PCA
from xgboost import XGBRegressor
from sklearn.metrics import accuracy_score
```

[43]:
```
pip install xgboost
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting xgboost
  Downloading xgboost-1.7.3-py3-none-win_amd64.whl (89.1 MB)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-
packages (from xgboost) (1.21.5)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-
packages (from xgboost) (1.7.3)
Installing collected packages: xgboost
Successfully installed xgboost-1.7.3
Note: you may need to restart the kernel to use updated packages.
```

[45]:
```
pca =PCA(n_components=0.99, random_state=102)
xtrain_trans=pca.fit_transform(xtrain)
xtest_trans=pca.transform(xtest)
print(xtrain.shape)
print(xtrain_trans.shape)
print(xtest.shape)
print(xtest_trans.shape)
```

```
(3156, 359)
(3156, 27)
(1053, 359)
(1053, 27)
```

[46]:
```
pca =PCA(n_components=0.99, random_state=102)
df_train_trans=pca.fit_transform(df_train_x)
print(df_train_x.shape)
print(df_train_trans.shape)
```

```
(4209, 359)
(4209, 27)
```

0.0.4 Predict your test_df values using XGBoost.

[47]:
```
xgb=XGBRegressor(base_score=0.5, booster='gbtree',␣
↪colsample_bylevel=1,colsample_bynode=1, colsample_bytree=0.5,
                enable_categorical=False, gamma=0, gpu_id=-1,␣
↪importance_type=None, interaction_constraints='',
                learning_rate=0.05, max_delta_step=0, max_depth=8,␣
↪min_child_weight=2, monotone_constraints='()',
                n_estimators=90, n_jobs=-1,num_parallel_tree=1, objective='reg:
↪squarederror',predictor='auto',
                random_state=2341, reg_alpha=1e-06,␣
↪reg_lambda=2,scale_pos_weight=19.3, subsample=1, tree_method='auto',
                validate_parameters=1, verbosity=0, eta=0.003)
```

[49]:
```
xgb.fit(xtrain_trans,ytrain)
```

[49]:
```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=0.5,
             early_stopping_rounds=None, enable_categorical=False, eta=0.003,
             eval_metric=None, feature_types=None, gamma=0, gpu_id=-1,
             grow_policy=None, importance_type=None, interaction_constraints='',
             learning_rate=0.05, max_bin=None, max_cat_threshold=None,
             max_cat_to_onehot=None, max_delta_step=0, max_depth=8,
             max_leaves=None, min_child_weight=2, missing=nan,
             monotone_constraints='()', n_estimators=90, n_jobs=-1,
             num_parallel_tree=1, predictor='auto', …)
```

[50]:
```
ypred=xgb.predict(xtest_trans)
xgb.predict(xtest_trans)
```

[50]:
```
array([ 94.65012 ,  93.778015,  77.745514, …, 109.12848 ,  94.11999 ,
       108.26532 ], dtype=float32)
```

[51]:
```
print(xgb.score(xtrain_trans,ytrain))
print(xgb.score(xtest_trans,ytest))
```

```
0.7837925674595312
0.5259128305736873
```

0.0.5 Prediction on Testing File

[52]:
```
xgb.predict(df_train_trans)
```

[52]:
```
array([103.95934 ,  92.636375,  77.92083 , …, 100.07799 ,  92.68107 ,
        96.01531 ], dtype=float32)
```

[ ]: