ML Mercedes Benz AI submission

December 5, 2019

[33]:
```
import numpy as np
import pandas as pd
import sklearn as sk
from sklearn import preprocessing

import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

import xgboost
color = sns.color_palette()
```

[34]:
```
#Load train dataset
trainData = pd.read_csv('train.csv')

#Load test dataset
testData = pd.read_csv('test.csv')
```

[35]:
```
#Metadata info
trainData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4209 entries, 0 to 4208
Columns: 378 entries, ID to X385
dtypes: float64(1), int64(369), object(8)
memory usage: 12.1+ MB
```

[36]:
```
trainData.head()
```

[36]:
```
   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]
```

[37]:
```
#statistical summary of the train dataset
trainData.describe()
```

[37]:
```
                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]
```

[38]:
```python
#0 variance - remove columns

trainData.var()[trainData.var()==0].index.values
```

[38]:
```
array(['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290',
       'X293', 'X297', 'X330', 'X347'], dtype=object)
```

[39]:
```python
#drop 0 variance columns- as per instructions
trainDataFinal=trainData.drop(trainData.var()[trainData.var()==0].index.values,
 ↪axis=1)
trainDataFinal.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4209 entries, 0 to 4208
Columns: 366 entries, ID to X385
dtypes: float64(1), int64(357), object(8)
memory usage: 11.8+ MB
```

[40]:
```python
#select features and labels
features=trainDataFinal.iloc[:,2:]
label=trainDataFinal.iloc[:,1].values

features.head()
```

[40]:
```
   X0 X1  X2 X3 X4 X5 X6 X8  X10  X12  ...  X375  X376  X377  X378  X379  \
0   k  v  at  a  d  u  j  o    0    0  ...     0     0     1     0     0
1   k  t  av  e  d  y  l  o    0    0  ...     1     0     0     0     0
2  az  w   n  c  d  x  j  x    0    0  ...     0     0     0     0     0
3  az  t   n  f  d  x  l  e    0    0  ...     0     0     0     0     0
4  az  v   n  f  d  h  d  n    0    0  ...     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 364 columns]
```

[41]:
```python
#Get object columns from feature column for LE and OHE

features.describe(include=['object'])
```

[41]:
```
          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
```

[42]:
```python
#Get only object column names

objcols=features.describe(include=['object']).columns.values
objcols
```

[42]:
```
array(['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8'], dtype=object)
```

[43]:
```python
#Label Encoder on the object column

from sklearn.preprocessing import LabelEncoder
#from sklearn.preprocessing import OHE

le=LabelEncoder()

for i in objcols:
  features[i]= le.fit_transform(features[i])

#ohe=OHE(categorical_features=[0,1,2,3,4,5,6,8])

Features=features.values
#featureohe=Features #ohe.fit_transform(fea).toarray()

#featureOHE
FeaturesCpy = Features

#import warnings
#warnings.filterwarnings('ignore')

#stateOHE = OneHotEncoder(categorical_features=[0,1,2,3,4,5,6,7])
#stateOHE.fit(FeaturesCpy)
#FeaturesCpy = stateOHE.transform(FeaturesCpy).toarray()
```

[44]:
```python
#PCA

#Feature scaling
from sklearn.preprocessing import StandardScaler
Stdsclr = StandardScaler()
ScaledFeatures = Stdsclr.fit_transform(FeaturesCpy)
ScaledFeatures.shape
```

[44]:
```
(4209, 364)
```

[45]:
```python
#PCA (dimension reduction) - All-in 364 compnents
from sklearn.decomposition import PCA

pca = PCA(n_components=364, svd_solver='full')
```

```
pca.fit(ScaledFeatures,label)
```

[45]:
```
PCA(copy=True, iterated_power='auto', n_components=364, random_state=None,
    svd_solver='full', tol=0.0, whiten=False)
```

[46]:
```
pca.explained_variance_ratio_
```

[46]:
```
array([6.89266892e-02, 5.68841213e-02, 4.53745695e-02, 3.42677135e-02,
       3.26430877e-02, 3.16266067e-02, 2.86252442e-02, 2.12375123e-02,
       1.97041337e-02, 1.78319550e-02, 1.64006646e-02, 1.56428622e-02,
       1.46274226e-02, 1.44833491e-02, 1.34828580e-02, 1.29516175e-02,
       1.24383219e-02, 1.17310333e-02, 1.12105051e-02, 1.07727793e-02,
       9.92512898e-03, 9.69449157e-03, 9.42523076e-03, 9.09867975e-03,
       8.74223908e-03, 8.43069482e-03, 7.90204943e-03, 7.63217614e-03,
       7.33541876e-03, 7.14905572e-03, 6.92957269e-03, 6.76677298e-03,
       6.52534438e-03, 6.41494089e-03, 6.22692040e-03, 5.99196153e-03,
       5.88086185e-03, 5.74693562e-03, 5.63702738e-03, 5.53409616e-03,
       5.50641149e-03, 5.40082599e-03, 5.33910600e-03, 5.24611598e-03,
       5.10194247e-03, 5.03232420e-03, 4.95885463e-03, 4.72691565e-03,
       4.64475228e-03, 4.56521484e-03, 4.39624472e-03, 4.32947533e-03,
       4.30322432e-03, 4.23762131e-03, 4.20211958e-03, 4.15473420e-03,
       4.06699537e-03, 4.03462750e-03, 3.91873995e-03, 3.88747084e-03,
       3.81759321e-03, 3.75560105e-03, 3.72440051e-03, 3.65910562e-03,
       3.59566547e-03, 3.55229656e-03, 3.49700953e-03, 3.46182278e-03,
       3.40471717e-03, 3.34157815e-03, 3.30984908e-03, 3.25521932e-03,
       3.24084510e-03, 3.21045090e-03, 3.16671200e-03, 3.16169105e-03,
       3.09963730e-03, 3.07579140e-03, 3.05317642e-03, 3.03955238e-03,
       3.00255847e-03, 2.98903148e-03, 2.95834824e-03, 2.92906873e-03,
       2.90789282e-03, 2.89246840e-03, 2.87088871e-03, 2.84861071e-03,
       2.82893540e-03, 2.80661865e-03, 2.79489424e-03, 2.76964748e-03,
       2.74977677e-03, 2.72538185e-03, 2.72396743e-03, 2.66915947e-03,
       2.64849773e-03, 2.63035478e-03, 2.60995440e-03, 2.60665425e-03,
       2.57383600e-03, 2.54968974e-03, 2.53621249e-03, 2.51953928e-03,
       2.50398175e-03, 2.48457976e-03, 2.44053200e-03, 2.42183601e-03,
       2.40353580e-03, 2.38432335e-03, 2.34517965e-03, 2.31166118e-03,
       2.30049772e-03, 2.27223313e-03, 2.25613123e-03, 2.23532575e-03,
       2.21465563e-03, 2.20028258e-03, 2.15055308e-03, 2.13667795e-03,
       2.10335610e-03, 2.09410362e-03, 2.05842058e-03, 2.04122098e-03,
       2.02468809e-03, 1.98912003e-03, 1.93876262e-03, 1.91967571e-03,
       1.91801301e-03, 1.87935329e-03, 1.85171337e-03, 1.81419321e-03,
       1.79357485e-03, 1.78621289e-03, 1.75392321e-03, 1.71752943e-03,
       1.70719881e-03, 1.67983148e-03, 1.65808186e-03, 1.61860975e-03,
       1.61139410e-03, 1.57706719e-03, 1.54261681e-03, 1.53538284e-03,
       1.49678823e-03, 1.49384890e-03, 1.47852266e-03, 1.42828100e-03,
       1.41108081e-03, 1.37376476e-03, 1.33814992e-03, 1.28803283e-03,
       1.26972355e-03, 1.26213163e-03, 1.20721187e-03, 1.16191771e-03,
       1.14336841e-03, 1.12752230e-03, 1.09498612e-03, 1.05869705e-03,
       9.99910624e-04, 9.89110199e-04, 9.86361463e-04, 9.54546927e-04,
```

```
9.45056062e-04, 9.11514885e-04, 8.99355342e-04, 8.80886749e-04,
8.63653466e-04, 8.59439761e-04, 8.29410991e-04, 8.24059997e-04,
7.98201807e-04, 7.91126650e-04, 7.78713087e-04, 7.44845247e-04,
7.24330743e-04, 7.08041140e-04, 6.85829934e-04, 6.72145117e-04,
6.55718352e-04, 6.40733883e-04, 6.12000447e-04, 5.86075445e-04,
5.58289468e-04, 5.45679895e-04, 5.41944228e-04, 5.30507026e-04,
5.23125193e-04, 5.09290043e-04, 4.93967305e-04, 4.75601629e-04,
4.59394696e-04, 4.52812516e-04, 4.39558529e-04, 4.26949429e-04,
4.14907621e-04, 4.07139378e-04, 3.99902280e-04, 3.96288548e-04,
3.86933450e-04, 3.72623799e-04, 3.62443593e-04, 3.54579912e-04,
3.38415435e-04, 3.29616574e-04, 3.16880863e-04, 3.07011359e-04,
2.91609975e-04, 2.84735221e-04, 2.74172380e-04, 2.65105542e-04,
2.57082857e-04, 2.44896369e-04, 2.37787938e-04, 2.29725881e-04,
2.28237225e-04, 2.26852369e-04, 2.18016571e-04, 2.09597981e-04,
1.98544338e-04, 1.74896477e-04, 1.63275485e-04, 1.56435191e-04,
1.48275745e-04, 1.45817565e-04, 1.39375105e-04, 1.25010833e-04,
1.16516633e-04, 1.08763512e-04, 1.00668042e-04, 9.32800134e-05,
8.37733596e-05, 7.81804284e-05, 6.89060898e-05, 6.38124700e-05,
6.20529750e-05, 5.69132679e-05, 4.99048590e-05, 4.85560153e-05,
4.49553468e-05, 4.31149301e-05, 4.23039470e-05, 3.76702109e-05,
3.67759864e-05, 3.16512054e-05, 3.01120917e-05, 2.60667672e-05,
2.04349916e-05, 1.43314337e-05, 1.24786542e-05, 1.17912074e-05,
8.13427984e-06, 7.14835152e-06, 4.73619419e-06, 4.54880789e-06,
3.00516288e-06, 2.48700880e-06, 1.42103632e-06, 1.35933926e-32,
2.03775596e-33, 2.02531407e-33, 1.60817378e-33, 1.41392516e-33,
1.33726553e-33, 1.22379355e-33, 1.15927483e-33, 1.02240091e-33,
8.55156440e-34, 7.95844170e-34, 7.22863434e-34, 5.69909583e-34,
4.18902155e-34, 3.55687842e-34, 3.16603046e-34, 3.14824492e-34,
3.12692316e-34, 3.12692316e-34, 3.12692316e-34, 3.12692316e-34,
3.12692316e-34, 3.12692316e-34, 3.12692316e-34, 3.12692316e-34,
3.12692316e-34, 3.12692316e-34, 3.12692316e-34, 3.12692316e-34,
3.12692316e-34, 3.12692316e-34, 3.12692316e-34, 3.12692316e-34,
3.12692316e-34, 3.12692316e-34, 3.12692316e-34, 3.12692316e-34,
3.12692316e-34, 3.12692316e-34, 3.12692316e-34, 3.12692316e-34,
3.12692316e-34, 3.12692316e-34, 3.12692316e-34, 3.12692316e-34,
3.12692316e-34, 3.12692316e-34, 3.12692316e-34, 3.12692316e-34,
3.12692316e-34, 3.12692316e-34, 3.12692316e-34, 3.12692316e-34,
3.12692316e-34, 3.12692316e-34, 3.12692316e-34, 3.12692316e-34,
3.12692316e-34, 3.12692316e-34, 3.12692316e-34, 3.12692316e-34,
3.12692316e-34, 3.12692316e-34, 3.12692316e-34, 3.12692316e-34,
3.12692316e-34, 3.12692316e-34, 3.12692316e-34, 3.12692316e-34,
3.12692316e-34, 3.12692316e-34, 3.12692316e-34, 3.12692316e-34,
3.12692316e-34, 3.12692316e-34, 3.12692316e-34, 3.12692316e-34,
3.12692316e-34, 3.12692316e-34, 3.12692316e-34, 3.12692316e-34,
3.12692316e-34, 3.12692316e-34, 3.12692316e-34, 3.12692316e-34,
3.12692316e-34, 3.12692316e-34, 3.12692316e-34, 3.12692316e-34,
```

```
       3.12692316e-34, 3.12692316e-34, 3.12692316e-34, 3.12692316e-34,
       3.12692316e-34, 3.12692316e-34, 2.81099293e-34, 1.85208053e-34,
       1.67229422e-34, 8.96352520e-35, 2.99729856e-35, 3.46465135e-36])
```

[47]:
```
#Mean variance - for threshold
np.mean(pca.explained_variance_ratio_)
```

[47]:
```
0.002747252747252748
```

[48]:
```
#number of components >threhold variance
pca.explained_variance_ratio_ > 0.002747252747252748
```

[48]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
```

```
        False, False, False, False, False, False, False, False, False,
        False, False, False, False, False, False, False, False, False,
        False, False, False, False, False, False, False, False, False,
        False, False, False, False])
```

[49]:

```
#PCA (dimension reduction) - 93 components
from sklearn.decomposition import PCA

pca = PCA(n_components=93, svd_solver='full')
pca.fit(ScaledFeatures,label)

#Transform the Features
finalFeatures = pca.transform(ScaledFeatures)
finalFeatures.shape
```

[49]:

```
(4209, 93)
```

[50]:

```
# Considering the n_components= 93 only as per above findings.
# Recreating the PCA Object with n_components = 93
pca = PCA(n_components=93)
pca.fit(ScaledFeatures,label)
```

[50]:

```
PCA(copy=True, iterated_power='auto', n_components=93, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

[62]:

```
#Using KFold method for gradient boosting

#Initialise the algo
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()

#Initiallise KFold Method
from sklearn.model_selection import KFold
from xgboost import XGBRegressor

kfold=KFold(n_splits=25,
           random_state=1,
           shuffle=True)
#kfold.split(ScaledFeatures)

#Initialise for Loop
i=0
for train,test in kfold.split(ScaledFeatures):
    i = i+1
    X_train,X_test = ScaledFeatures[train],ScaledFeatures[test]
    y_train,y_test = label[train],label[test]
    model = XGBRegressor(objective='reg:squarederror', learning_rate=0.1)

    model.fit(X_train,y_train)
```

```
    if model.score(X_test,y_test) > 0.10: #model.score(X_train, y_train)
        print("Test Score: {}, train score: {}, for Sample Split: {}".
↪format(model.score(X_test,y_test),model.score(X_train,y_train),i))
```

```
Test Score: 0.48812156371958315, train score: 0.6131953344929971, for Sample
Split: 1
Test Score: 0.564488402858333, train score: 0.6098104362924426, for Sample
Split: 2
Test Score: 0.6037606920349106, train score: 0.6090151691828285, for Sample
Split: 3
Test Score: 0.5648590045659287, train score: 0.6103941336900728, for Sample
Split: 4
Test Score: 0.7483431141129421, train score: 0.6041973992150029, for Sample
Split: 5
Test Score: 0.6817623681188989, train score: 0.6070890723682825, for Sample
Split: 6
Test Score: 0.7309118390733638, train score: 0.6057534591617242, for Sample
Split: 7
Test Score: 0.6007772988961381, train score: 0.6086492179415797, for Sample
Split: 8
Test Score: 0.577071324552104, train score: 0.608395674552963, for Sample Split:
9
Test Score: 0.22003356043711442, train score: 0.6390559120942797, for Sample
Split: 10
Test Score: 0.42882686957253974, train score: 0.615517256274491, for Sample
Split: 11
Test Score: 0.6427831543534956, train score: 0.6073399102797304, for Sample
Split: 12
Test Score: 0.5926522347218254, train score: 0.6090972555835208, for Sample
Split: 13
Test Score: 0.5902656777272614, train score: 0.6085134982687097, for Sample
Split: 14
Test Score: 0.5222916652718459, train score: 0.6115777254611762, for Sample
Split: 15
Test Score: 0.509359191100992, train score: 0.6119883355456182, for Sample
Split: 16
Test Score: 0.5129599864112163, train score: 0.6127247537876668, for Sample
Split: 17
Test Score: 0.599266123931889, train score: 0.6094343025934467, for Sample
Split: 18
Test Score: 0.6645951139827526, train score: 0.6053542047417008, for Sample
Split: 19
Test Score: 0.5474651839887594, train score: 0.6095127492152392, for Sample
Split: 20
Test Score: 0.7190861639379977, train score: 0.6053068943652814, for Sample
Split: 21
```

```
Test Score: 0.6290824984505579, train score: 0.6057355932542555, for Sample
Split: 22
Test Score: 0.6367173564176096, train score: 0.6063298839872899, for Sample
Split: 23
Test Score: 0.4879831374027417, train score: 0.609191491455471, for Sample
Split: 24
Test Score: 0.5825672705231493, train score: 0.6091322287630705, for Sample
Split: 25
```

[75]:
```
#Test Score: 0.7483431141129421, train score: 0.6041973992150029, for Sample␣
 ↪Split: 5
#Extract sample 5 with split 25

kfold = KFold(n_splits=25,
              random_state=1,
              shuffle=True)
i=0
for train,test in kfold.split(finalFeatures):
    i = i+1
    if i == 5:
        X_train,X_test,y_train,y_test =␣
 ↪finalFeatures[train],finalFeatures[test],label[train],label[test]
```

[78]:
```
#fit into model- XGBoost regression
model = XGBRegressor(objective='reg:squarederror', learning_rate=0.1)

model.fit(X_train,y_train)

#Check the quality of model
print("Training Accuracy ",model.score(X_train,y_train))
print("Testing Accuracy ",model.score(X_test,y_test))
```

```
Training Accuracy  0.6562389078326774
Testing Accuracy  0.6756990881451739
```

[79]:
```
#Check model accuracy and error metrics
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error

#MAE - check this error metrics
mean_absolute_error(y_test,model.predict(X_test))
```

[79]: 5.200063899113583

[80]:
```
#Check model accuracy based on R2
r2_score(y_test,model.predict(X_test), multioutput='variance_weighted')
```

[80]: 0.6756990881451739

[81]:
```
#Check MSE
mean_squared_error(y_test,model.predict(X_test))
```

[81]: 43.56893272513377

[82]:
```
#Process TEST dataset to predict Y
```

[83]:
```
#TEST data
testData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4209 entries, 0 to 4208
Columns: 377 entries, ID to X385
dtypes: int64(369), object(8)
memory usage: 12.1+ MB
```

[84]:
```
testData.head()
```

[84]:
```
   ID  X0 X1  X2 X3 X4 X5 X6 X8  X10  ...  X375  X376  X377  X378  X379  X380  \
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

   X382  X383  X384  X385
0     0     0     0     0
1     0     0     0     0
2     0     0     0     0
3     0     0     0     0
4     0     0     0     0

[5 rows x 377 columns]
```

[85]:
```
#statistical summary of the test data set
testData.describe()
```

[85]:
```
                ID          X10          X11          X12          X13  \
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000
mean   4211.039202     0.019007     0.000238     0.074364     0.061060
std    2423.078926     0.136565     0.015414     0.262394     0.239468
min       1.000000     0.000000     0.000000     0.000000     0.000000
25%    2115.000000     0.000000     0.000000     0.000000     0.000000
50%    4202.000000     0.000000     0.000000     0.000000     0.000000
75%    6310.000000     0.000000     0.000000     0.000000     0.000000
max    8416.000000     1.000000     1.000000     1.000000     1.000000

               X14          X15          X16          X17          X18  ...  \
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000  ...
mean      0.427893     0.000713     0.002613     0.008791     0.010216  ...
```

```
std        0.494832     0.026691     0.051061     0.093357     0.100570  ...
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     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      0.325968     0.049656     0.311951     0.019244     0.011879
std       0.468791     0.217258     0.463345     0.137399     0.108356
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     0.008791     0.000475     0.000713     0.001663
std       0.089524     0.093357     0.021796     0.026691     0.040752
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 369 columns]
```

[92]:
```
#Remove some columns like for Train
#'X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290', 'X293', 'X297',␣
↪'X330', 'X347'

datafinal=testData.drop(['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289',␣
↪'X290','X293', 'X297', 'X330', 'X347'], axis=1)
datafinal.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4209 entries, 0 to 4208
Columns: 365 entries, ID to X385
dtypes: int64(357), object(8)
memory usage: 11.7+ MB
```

[93]:
```
#Get feature columns

testFeatures=datafinal.iloc[:,1:]

#get object columns
```

```
objcolstest=testFeatures.describe(include=['object']).columns.values
objcolstest
```

[93]:
```
array(['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8'], dtype=object)
```

[94]:
```
#Apply Label Encoder for object columns.

from sklearn.preprocessing import LabelEncoder
#from sklearn.preprocessing import OneHotEncoder

le=LabelEncoder()

for i in objcolstest:
  testFeatures[i]= le.fit_transform(testFeatures[i])

f1=testFeatures.values



f1.ndim
```

[94]:
```
2
```

[95]:
```
#Transform test data - PCA

transformFromPCA = pca.transform(f1)

labelpred=model.predict(transformFromPCA)
labelpred
```

[95]:
```
array([ 97.06038 ,  94.02681 ,  94.24919 , ...,  89.247284, 101.25636 ,
        91.10257 ], dtype=float32)
```

[102]:
```
#Save to file

testData.to_csv('TestPredictiveValue.csv')
```

[101]:
```
#Append Y predicted value in test data set

testdataforfile=pd.concat([testData.iloc[:,0],pd.concat([pd.
 ↪DataFrame(data=labelpred, columns=['y']),testData.iloc[:,1:]],␣
 ↪axis=1)],axis=1)

testdataforfile.head()
```

[101]:
```
   ID          y  X0 X1  X2 X3 X4 X5 X6 X8  ...  X375  X376  X377  X378  X379  \
0   1  97.060379  az  v   n  f  d  t  a  w  ...     0     0     0     1     0
1   2  94.026810   t  b  ai  a  d  b  g  y  ...     0     0     1     0     0
2   3  94.249191  az  v  as  f  d  a  j  j  ...     0     0     0     1     0
3   4  95.560425  az  l   n  f  d  z  l  n  ...     0     0     0     1     0
4   5  94.971527   w  s  as  c  d  y  i  m  ...     1     0     0     0     0
```

```
   X380  X382  X383  X384  X385
0     0     0     0     0     0
1     0     0     0     0     0
2     0     0     0     0     0
3     0     0     0     0     0
4     0     0     0     0     0

[5 rows x 378 columns]
```

[ ]: