MLproject

November 25, 2022

[1]:
```
#Mercedes-Benz Greener Manufacturing project
```

[137]:
```
import pandas as pd
import numpy as np
import xgboost as xgb
import types
from xgboost import XGBRegressor
```

[3]:
```
from sklearn.decomposition import PCA
```

[4]:
```
df_train = pd.read_csv(r"C:\Users\Pinki\Downloads\machine learning␣
↪simplilearn\train\train.csv")
```

```
      ␣
↪---------------------------------------------------------------------------

       FileNotFoundError                         Traceback (most recent call␣
↪last)

       <ipython-input-4-6f3a1afaa2a6> in <module>
   ----> 1 df_train = pd.read_csv(r"C:\Users\Pinki\Downloads\machine learning␣
↪simplilearn\train\train.csv")


       /usr/local/lib/python3.7/site-packages/pandas/io/parsers.py in␣
↪read_csv(filepath_or_buffer, sep, delimiter, header, names, index_col,␣
↪usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine, converters,␣
↪true_values, false_values, skipinitialspace, skiprows, skipfooter, nrows,␣
↪na_values, keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates,␣
↪infer_datetime_format, keep_date_col, date_parser, dayfirst, cache_dates,␣
↪iterator, chunksize, compression, thousands, decimal, lineterminator,␣
↪quotechar, quoting, doublequote, escapechar, comment, encoding, dialect,␣
↪error_bad_lines, warn_bad_lines, delim_whitespace, low_memory, memory_map,␣
↪float_precision)
       686     )
       687
```

```
   --> 688     return _read(filepath_or_buffer, kwds)
       689
       690


       /usr/local/lib/python3.7/site-packages/pandas/io/parsers.py in␣
↪_read(filepath_or_buffer, kwds)
       452
       453     # Create the parser.
   --> 454     parser = TextFileReader(fp_or_buf, **kwds)
       455
       456     if chunksize or iterator:


       /usr/local/lib/python3.7/site-packages/pandas/io/parsers.py in␣
↪__init__(self, f, engine, **kwds)
       946             self.options["has_index_names"] = kwds["has_index_names"]
       947
   --> 948         self._make_engine(self.engine)
       949
       950     def close(self):


       /usr/local/lib/python3.7/site-packages/pandas/io/parsers.py in␣
↪_make_engine(self, engine)
      1178     def _make_engine(self, engine="c"):
      1179         if engine == "c":
   -> 1180             self._engine = CParserWrapper(self.f, **self.options)
      1181         else:
      1182             if engine == "python":


       /usr/local/lib/python3.7/site-packages/pandas/io/parsers.py in␣
↪__init__(self, src, **kwds)
      2008         kwds["usecols"] = self.usecols
      2009
   -> 2010         self._reader = parsers.TextReader(src, **kwds)
      2011         self.unnamed_cols = self._reader.unnamed_cols
      2012


       pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.__cinit__()


       pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.
↪_setup_parser_source()
```

```
FileNotFoundError: [Errno 2] No such file or directory: 'C:
↪\\Users\\Pinki\\Downloads\\machine learning simplilearn\\train\\train.csv'
```

[7]:
```
df_train = pd.read_csv('C:\Users\Pinki\Downloads\machine learning␣
↪simplilearn\train\train.csv')
```

```
  File "<ipython-input-7-1543ad9eaae8>", line 1
    df_train = pd.read_csv('C:\Users\Pinki\Downloads\machine learning␣
↪simplilearn\train\train.csv')
                                ^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in␣
↪position 2-3: truncated \UXXXXXXXX escape
```

[8]:
```
df_train = pd.read_csv("train.csv")
```

[9]:
```
df_train.head()
```

[9]:

	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]

[10]:
```
df_train.shape
```

[10]:
```
(4209, 378)
```

[11]:
```
df_train.dtypes
```

[11]:
```
ID        int64
y       float64
X0       object
```

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

[12]:
```
df_train.isnull().sum()
```

[12]:
```
ID      0
y       0
X0      0
X1      0
X2      0
       ..
X380    0
X382    0
X383    0
X384    0
X385    0
Length: 378, dtype: int64
```

[13]:
```
df_train.describe()
```

[13]:

	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]
```

[14]:
```
df_train = df_train.drop('ID', axis=1)
```

[15]:
```
df_train.head()
```

[15]:
```
        y  X0 X1  X2 X3 X4 X5 X6 X8  X10  …  X375  X376  X377  X378  X379  \
0  130.81   k  v  at  a  d  u  j  o    0  …     0     0     1     0     0
1   88.53   k  t  av  e  d  y  l  o    0  …     1     0     0     0     0
2   76.26  az  w   n  c  d  x  j  x    0  …     0     0     0     0     0
3   80.62  az  t   n  f  d  x  l  e    0  …     0     0     0     0     0
4   78.02  az  v   n  f  d  h  d  n    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 377 columns]
```

[16]:
```
df_train.shape
```

[16]:
```
(4209, 377)
```

[17]:
```
# Seperating the numerical and categorical columns for train data
df_cat = df_train.select_dtypes(include = np.object)
df_num = df_train.select_dtypes(exclude=np.object)
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:2:
DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`.
To silence this warning, use `object` by itself. Doing this will not modify any
behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations

/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:3:
DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`.
To silence this warning, use `object` by itself. Doing this will not modify any
behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  This is separate from the ipykernel package so we can avoid doing imports
until
```

[18]:
```
# categorical data for train dataset
df_cat.head()
```

[18]:

	X0	X1	X2	X3	X4	X5	X6	X8
0	k	v	at	a	d	u	j	o
1	k	t	av	e	d	y	l	o
2	az	w	n	c	d	x	j	x
3	az	t	n	f	d	x	l	e
4	az	v	n	f	d	h	d	n

[19]:
```
# Numerical data for train dataset
df_num.head()
```

[19]:

	y	X10	X11	X12	X13	X14	X15	X16	X17	X18	…	X375	X376	X377	\
0	130.81	0	0	0	1	0	0	0	0	1	…	0	0	1	
1	88.53	0	0	0	0	0	0	0	0	1	…	1	0	0	
2	76.26	0	0	0	0	0	0	0	1	0	…	0	0	0	
3	80.62	0	0	0	0	0	0	0	0	0	…	0	0	0	
4	78.02	0	0	0	0	0	0	0	0	0	…	0	0	0	

	X378	X379	X380	X382	X383	X384	X385
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0

```
[5 rows x 369 columns]
```

[20]:
```
# drop dependent variable from numerical data of train set
df_num = df_num.drop("y", axis = 1)
df_num.head()
```

[20]:

	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	…	X375	X376	X377	\
0	0	0	0	1	0	0	0	0	1	0	…	0	0	1	
1	0	0	0	0	0	0	0	0	1	0	…	1	0	0	
2	0	0	0	0	0	0	0	1	0	0	…	0	0	0	
3	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	

	X378	X379	X380	X382	X383	X384	X385
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0

```
[5 rows x 368 columns]
```

[21]:
```
columns = df_num.columns
```

[22]:
```
df_num.shape
```

[22]:
```
(4209, 368)
```

[24]:
```
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(df_train)
```

```
There are no missing values in the dataframe
```

[25]:
```
# Applying scaling technique for numerical data of train set
from sklearn.preprocessing import MinMaxScaler, StandardScaler
mn = MinMaxScaler()
```

[26]:
```
df_mn = mn.fit_transform(df_num)
```

[27]:
```
df_num_sc = pd.DataFrame(df_mn, index=df_num.index, columns=df_num.columns)
df_num_sc.head()
```

[27]:
```
   X10  X11  X12  X13  X14  X15  X16  X17  X18  X19  …  X375  X376  X377  \
0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0  1.0  0.0  …   0.0   0.0   1.0
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0  …   1.0   0.0   0.0
2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0  0.0  …   0.0   0.0   0.0
3  0.0  0.0  0.0  0.0  0.0  0.0  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  0.0  0.0  0.0  …   0.0   0.0   0.0

   X378  X379  X380  X382  X383  X384  X385
0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1   0.0   0.0   0.0   0.0   0.0   0.0   0.0
2   0.0   0.0   0.0   1.0   0.0   0.0   0.0
3   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

[5 rows x 368 columns]
```

[28]:
```
#If for any column(s), the variance is equal to zero, then we need to remove
 ↪those variable(s).
```

[30]:
```
if 'X4' in df_train.columns.values:
    df_train = df_train.drop('X4',axis=1)
```

[31]:
```
zero_var = df_train.var()[df_train.var()==0].index
df_train = df_train.drop(zero_var,axis=1)
```

[32]:
```
df_train.head()
```

[32]:
```
        y  X0 X1  X2 X3 X5 X6 X8  X10  X12  …  X375  X376  X377  X378  X379  \
0  130.81   k  v  at  a  u  j  o    0    0  …     0     0     1     0     0
1   88.53   k  t  av  e  y  l  o    0    0  …     1     0     0     0     0
2   76.26  az  w   n  c  x  j  x    0    0  …     0     0     0     0     0
3   80.62  az  t   n  f  x  l  e    0    0  …     0     0     0     0     0
4   78.02  az  v   n  f  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]
```

[34]:
```
np.ravel(zero_var)
```

[34]:
```
array(['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290',
       'X293', 'X297', 'X330', 'X347'], dtype=object)
```

[35]:
```
# columns having Zero variance in training data set will be dropped
df_num_variance_with_zero_drop = df_num.drop(['X11', 'X93', 'X107', 'X233',␣
 ↪'X235', 'X268', 'X289', 'X290',
       'X293', 'X297', 'X330', 'X347'], axis = 1)
```

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

[36]:

	X10	X12	X13	X14	X15	X16	X17	X18	X19	X20	…	X375	X376	X377	\
0	0	0	1	0	0	0	0	1	0	0	…	0	0	1	
1	0	0	0	0	0	0	0	1	0	0	…	1	0	0	
2	0	0	0	0	0	0	1	0	0	0	…	0	0	0	
3	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	

	X378	X379	X380	X382	X383	X384	X385
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0

[5 rows x 356 columns]

[37]:
```
df_num_variance_with_zero_drop.describe()
```

[37]:

	X10	X12	X13	X14	X15	\
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	
mean	0.013305	0.075077	0.057971	0.428130	0.000475	
std	0.114590	0.263547	0.233716	0.494867	0.021796	
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	1.000000	0.000000	
max	1.000000	1.000000	1.000000	1.000000	1.000000	

	X16	X17	X18	X19	X20	…	\
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	…	
mean	0.002613	0.007603	0.007840	0.099549	0.142789	…	
std	0.051061	0.086872	0.088208	0.299433	0.349899	…	
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	…	

	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 356 columns]
```

[38]:
```
df_num_variance_with_zero_drop.shape
```

[38]:
```
(4209, 356)
```

[39]:
```
df_train.shape
```

[39]:
```
(4209, 364)
```

[40]:
```
df_cat.shape
```

[40]:
```
(4209, 8)
```

[41]:
```
#Check unique values for train sets.
```

[42]:
```
# to find number of unique values in each feature
df_train.nunique()
```

[42]:
```
y       2545
X0        47
X1        27
X2        44
X3         7
       ...
X380       2
X382       2
X383       2
X384       2
```

```
X385       2
Length: 364, dtype: int64
```

[43]:
```
#Applying label encoder
```

[44]:
```
# df_cat_dum = pd.get_dummies(df_cat)
# apply OHE - One Hot Encoding
from sklearn.preprocessing import OneHotEncoder
```

[45]:
```
ohe = OneHotEncoder(handle_unknown = "ignore")
```

[46]:
```
df_cat_dum = ohe.fit_transform(df_cat).toarray()
col_names = ohe.get_feature_names()
col_names = np.array(col_names).ravel()
df_cat_oh  =pd.DataFrame(df_cat_dum, columns=col_names)
```

```
/usr/local/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87:
FutureWarning: Function get_feature_names is deprecated; get_feature_names is
deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out
instead.
  warnings.warn(msg, category=FutureWarning)
```

[47]:
```
df_cat_oh.head()
```

[47]:

	x0_a	x0_aa	x0_ab	x0_ac	x0_ad	x0_af	x0_ai	x0_aj	x0_ak	x0_al	…	\
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	
3	0.0	0.0	0.0	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	0.0	0.0	0.0	…	

	x7_p	x7_q	x7_r	x7_s	x7_t	x7_u	x7_v	x7_w	x7_x	x7_y
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
3	0.0	0.0	0.0	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	0.0	0.0	0.0

[5 rows x 195 columns]

[48]:
```
df_cat_oh.shape
```

[48]:
```
(4209, 195)
```

[49]:
```
# Concatenate categorical and numerical data into one data frame of training␣
↪data
```

```
df_train_final = pd.concat([df_num_variance_with_zero_drop, df_cat_oh], axis =␣
↪1)
```

[50]:
```
df_train_final.head()
```

[50]:
```
   X10  X12  X13  X14  X15  X16  X17  X18  X19  X20  …  x7_p  x7_q  x7_r  \
0    0    0    1    0    0    0    0    1    0    0  …   0.0   0.0   0.0
1    0    0    0    0    0    0    0    1    0    0  …   0.0   0.0   0.0
2    0    0    0    0    0    0    1    0    0    0  …   0.0   0.0   0.0
3    0    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   0.0

   x7_s  x7_t  x7_u  x7_v  x7_w  x7_x  x7_y
0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1   0.0   0.0   0.0   0.0   0.0   0.0   0.0
2   0.0   0.0   0.0   0.0   0.0   1.0   0.0
3   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

[5 rows x 551 columns]
```

[51]:
```
df_train_final.shape
```

[51]:
```
(4209, 551)
```

[52]:
```
#Performing dimensionality reduction.
```

[53]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=24)
```

[54]:
```
df_train.dtypes
```

[54]:
```
y       float64
X0       object
X1       object
X2       object
X3       object
         …
X380      int64
X382      int64
X383      int64
X384      int64
X385      int64
Length: 364, dtype: object
```

[55]:
```
x_pca = pca.fit_transform(df_train_final)
```

[56]:
```
df_train_final.shape
```

[56]:
```
(4209, 551)
```

[57]:
```
df_pca = pd.DataFrame(x_pca)
```

[58]:
```
df_pca.head()
```

[58]:
```
          0         1         2         3         4         5         6   \
0  0.850248 -1.252515  2.021640  0.865223  1.592171 -0.056846  0.563838
1 -0.109302 -1.299662 -0.045801 -0.796931  0.277976  0.140880  1.108070
2 -0.673653 -2.367697  1.787792  2.345645  0.356806  3.753878 -1.188808
3 -0.480940 -2.695789  0.524340  2.881771 -0.485304  3.765186 -0.307379
4 -0.516369 -2.692792  0.334140  3.103397 -0.723453  3.866238 -0.451954

          7         8         9   …        14        15        16        17  \
0 -1.030706  0.205193 -0.264542  …  0.036644  0.295524 -0.520436 -0.474907
1 -0.726632 -0.032187  0.612274  … -0.981760 -0.647934 -0.004761  0.095950
2  0.679650 -0.924721 -0.215837  …  0.295002  0.845014 -0.352708 -0.827251
3 -0.014648 -1.239944  0.254645  …  0.240064  0.358944  0.274851 -0.782206
4  0.151803 -1.801272 -0.298136  … -0.112328 -0.216499 -0.091709 -0.203135

         18        19        20        21        22        23
0 -0.527634  0.409433 -0.334511  1.123880 -0.245582  1.369525
1  0.854517 -0.200809 -0.877111  0.627602 -0.338007  1.366848
2  0.562821  0.591346  0.884917 -0.560375  0.573807  0.656158
3  0.821065  0.616899 -0.348829 -0.331692  0.243354 -0.232898
4  0.414848  0.169492 -0.032345  0.432690  0.332758  0.286641

[5 rows x 24 columns]
```

[59]:
```
pca.explained_variance_ratio_
```

[59]:
```
array([0.11327864, 0.07799109, 0.07358181, 0.05848106, 0.04943089,
       0.04191889, 0.03310021, 0.0282729 , 0.02515469, 0.02153505,
       0.02077602, 0.01725079, 0.01505285, 0.01435205, 0.01385206,
       0.01296764, 0.01205453, 0.01092863, 0.00984203, 0.00913118,
       0.00883257, 0.00843298, 0.00823132, 0.00772469])
```

[60]:
```
#analysis of test data
```

[62]:
```
df_test = pd.read_csv("test.csv")
```

[63]:
```
df_test.head()
```

[63]:
```
   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]
```

[64]:
```
# Checking null values in test set
df_test.isnull().sum()
```

[64]:
```
ID      0
X0      0
X1      0
X2      0
X3      0
       ..
X380    0
X382    0
X383    0
X384    0
X385    0
Length: 377, dtype: int64
```

[65]:
```
df_test.nunique()
```

[65]:
```
ID      4209
X0        49
X1        27
X2        45
X3         7
        …
X380       2
X382       2
X383       2
X384       2
X385       2
Length: 377, dtype: int64
```

[66]:
```
np.ravel(df_test.nunique())
```

```
[66]: array([4209,   49,   27,   45,    7,    4,   32,   12,   25,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    1,    1,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    1,    1,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    1,    2,
          2,    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
          2,    2,    2])
```

```
[69]: df_test.shape
```

```
[69]: (4209, 377)
```

```
[70]: df_test.dtypes
```

```
[70]: ID       int64
      X0      object
      X1      object
      X2      object
      X3      object
```

```
        …
X380    int64
X382    int64
X383    int64
X384    int64
X385    int64
Length: 377, dtype: object
```

[71]:
```
# Seperating the numerical and categorical columns for test data
test_df_cat = df_test.select_dtypes(include = np.object)
test_df_num = df_test.select_dtypes(exclude = np.object)
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:2:
DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`.
To silence this warning, use `object` by itself. Doing this will not modify any
behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations

/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:3:
DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`.
To silence this warning, use `object` by itself. Doing this will not modify any
behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  This is separate from the ipykernel package so we can avoid doing imports
until
```

[72]:
```
test_df_cat.head()
```

[72]:

	X0	X1	X2	X3	X4	X5	X6	X8
0	az	v	n	f	d	t	a	w
1	t	b	ai	a	d	b	g	y
2	az	v	as	f	d	a	j	j
3	az	l	n	f	d	z	l	n
4	w	s	as	c	d	y	i	m

[73]:
```
test_df_num.head()
```

[73]:

	ID	X10	X11	X12	X13	X14	X15	X16	X17	X18	…	X375	X376	X377	\
0	1	0	0	0	0	0	0	0	0	0	…	0	0	0	
1	2	0	0	0	0	0	0	0	0	0	…	0	0	1	
2	3	0	0	0	0	1	0	0	0	0	…	0	0	0	
3	4	0	0	0	0	0	0	0	0	0	…	0	0	0	
4	5	0	0	0	0	1	0	0	0	0	…	1	0	0	

	X378	X379	X380	X382	X383	X384	X385

```
0     1     0     0     0     0     0     0
1     0     0     0     0     0     0     0
2     1     0     0     0     0     0     0
3     1     0     0     0     0     0     0
4     0     0     0     0     0     0     0

[5 rows x 369 columns]
```

[75]:
```
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(df_test)
```

```
There are no missing values in the dataframe
```

[76]:
```
test_df_num = test_df_num.drop("ID", axis = 1)
test_df_num.head()
```

[76]:
```
   X10  X11  X12  X13  X14  X15  X16  X17  X18  X19  …  X375  X376  X377  \
0    0    0    0    0    0    0    0    0    0    0  …     0     0     0
1    0    0    0    0    0    0    0    0    0    1  …     0     0     1
2    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
4    0    0    0    0    1    0    0    0    0    0  …     1     0     0

   X378  X379  X380  X382  X383  X384  X385
0     1     0     0     0     0     0     0
1     0     0     0     0     0     0     0
2     1     0     0     0     0     0     0
3     1     0     0     0     0     0     0
4     0     0     0     0     0     0     0

[5 rows x 368 columns]
```

[77]:
```
test_df_num.shape
```

[77]:
```
(4209, 368)
```

[78]:
```
test_columns = test_df_num.columns
test_columns
```

[78]:
```
Index(['X10', 'X11', 'X12', 'X13', 'X14', 'X15', 'X16', 'X17', 'X18', 'X19',
       …
       'X375', 'X376', 'X377', 'X378', 'X379', 'X380', 'X382', 'X383', 'X384',
       'X385'],
```

```
      dtype='object', length=368)
```

[79]:
```
# To apply scaling for test set
test_df_num_sc = mn.transform(test_df_num)
test_df_num_df = pd.DataFrame(test_df_num_sc, index = test_df_num.index,␣
↪columns=test_df_num.columns)
test_df_num_df.head()
```

[79]:

	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	…	X375	X376	X377	\
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	…	0.0	0.0	1.0	
2	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	…	1.0	0.0	0.0	

	X378	X379	X380	X382	X383	X384	X385
0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	1.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1.0	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

[5 rows x 368 columns]

[80]:
```
#Test Set - If for any column(s), the variance is equal to zero, then you need␣
↪to remove those variable(s)
```

[105]:
```
test_variance_df_num = test_df_num.var()
```

[106]:
```
test_variable_var_zero = [ ]

for i in range(0,len(test_variance_df_num)):
    if test_variance_df_num[i]==0: #checking if the variance for the df_num␣
↪dataframe column has zero
        test_variable_var_zero.append(test_columns[i])
```

[107]:
```
np.ravel(test_variable_var_zero)
```

[107]:
```
array(['X257', 'X258', 'X295', 'X296', 'X369'], dtype='<U4')
```

[108]:
```
# columns having Zero variance in test data set will be dropped
test_df_num_variance_with_zero_drop = test_df_num.drop(['X257', 'X258', 'X295',␣
↪'X296', 'X369'], axis = 1)
```

[109]:
```
test_df_num_variance_with_zero_drop.head()
```

[109]:

	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	…	X375	X376	X377	\
0	0	0	0	0	0	0	0	0	0	0	…	0	0	0	
1	0	0	0	0	0	0	0	0	0	1	…	0	0	1	
2	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	
4	0	0	0	0	1	0	0	0	0	0	…	1	0	0	

	X378	X379	X380	X382	X383	X384	X385
0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0
3	1	0	0	0	0	0	0
4	0	0	0	0	0	0	0

[5 rows x 363 columns]

[110]:
```
test_df_num_variance_with_zero_drop.shape
```

[110]: (4209, 363)

[111]:
```
# Applying ONE HOT encoder for test set
test_df_cat_dum = ohe.transform(test_df_cat).toarray()
test_col_names = ohe.get_feature_names()
test_col_names = np.array(test_col_names).ravel()
test_df_cat_oh  =pd.DataFrame(test_df_cat_dum, columns=test_col_names)
test_df_cat_oh.head()
```

```
/usr/local/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87:
FutureWarning: Function get_feature_names is deprecated; get_feature_names is
deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out
instead.
  warnings.warn(msg, category=FutureWarning)
```

[111]:

	x0_a	x0_aa	x0_ab	x0_ac	x0_ad	x0_af	x0_ai	x0_aj	x0_ak	x0_al	…	\
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	
3	0.0	0.0	0.0	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	0.0	0.0	0.0	…	

	x7_p	x7_q	x7_r	x7_s	x7_t	x7_u	x7_v	x7_w	x7_x	x7_y
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	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	0.0	0.0	0.0

[5 rows x 195 columns]

[112]:
```
# concatenating the both categorical and numerical features of test set
df_test_final = pd.concat([test_df_num_variance_with_zero_drop,␣
 ↪test_df_cat_oh], axis = 1)
```

[113]:
```
df_test_final.head()
```

[113]:

	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	…	x7_p	x7_q	x7_r	\
0	0	0	0	0	0	0	0	0	0	0	…	0.0	0.0	0.0	
1	0	0	0	0	0	0	0	0	0	1	…	0.0	0.0	0.0	
2	0	0	0	0	1	0	0	0	0	0	…	0.0	0.0	0.0	
3	0	0	0	0	0	0	0	0	0	0	…	0.0	0.0	0.0	
4	0	0	0	0	1	0	0	0	0	0	…	0.0	0.0	0.0	

	x7_s	x7_t	x7_u	x7_v	x7_w	x7_x	x7_y
0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	1.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	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

[5 rows x 558 columns]

[114]:
```
print(df_train_final.shape)
print(df_test_final.shape)
```

```
(4209, 551)
(4209, 558)
```

[115]:
```
# while dropping columns with 0 variance for train and test data sets feature␣
 ↪results are different,
# so to balance the feature in train and test sets, we have to add dropped␣
 ↪dummy columns with NAN values to apply PCA
# resetting the test data features to align with train features
test_df_newdata = df_test_final.reindex(labels=df_train_final.columns,axis=1)
test_df_newdata.head()
```

[115]:

	X10	X12	X13	X14	X15	X16	X17	X18	X19	X20	…	x7_p	x7_q	x7_r	\
0	0	0	0	0	0	0	0	0	0	0	…	0.0	0.0	0.0	
1	0	0	0	0	0	0	0	0	1	0	…	0.0	0.0	0.0	
2	0	0	0	1	0	0	0	0	0	0	…	0.0	0.0	0.0	
3	0	0	0	0	0	0	0	0	0	0	…	0.0	0.0	0.0	
4	0	0	0	1	0	0	0	0	0	0	…	0.0	0.0	0.0	

	x7_s	x7_t	x7_u	x7_v	x7_w	x7_x	x7_y
0	0.0	0.0	0.0	0.0	1.0	0.0	0.0

```
1   0.0   0.0   0.0   0.0   0.0   0.0   1.0
2   0.0   0.0   0.0   0.0   0.0   0.0   0.0
3   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

[5 rows x 551 columns]
```

[116]:
```
# fill the NAN values with 0 to fit to PCA
test_df_newdata["X257"] = test_df_newdata["X257"].fillna(0)
test_df_newdata["X258"] = test_df_newdata["X258"].fillna(0)
test_df_newdata["X295"] = test_df_newdata["X295"].fillna(0)
test_df_newdata["X296"] = test_df_newdata["X296"].fillna(0)
test_df_newdata["X369"] = test_df_newdata["X369"].fillna(0)
test_df_newdata.head()
```

[116]:
```
   X10  X12  X13  X14  X15  X16  X17  X18  X19  X20  …  x7_p  x7_q  x7_r  \
0    0    0    0    0    0    0    0    0    0    0  …   0.0   0.0   0.0
1    0    0    0    0    0    0    0    0    1    0  …   0.0   0.0   0.0
2    0    0    0    1    0    0    0    0    0    0  …   0.0   0.0   0.0
3    0    0    0    0    0    0    0    0    0    0  …   0.0   0.0   0.0
4    0    0    0    1    0    0    0    0    0    0  …   0.0   0.0   0.0

   x7_s  x7_t  x7_u  x7_v  x7_w  x7_x  x7_y
0   0.0   0.0   0.0   0.0   1.0   0.0   0.0
1   0.0   0.0   0.0   0.0   0.0   0.0   1.0
2   0.0   0.0   0.0   0.0   0.0   0.0   0.0
3   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

[5 rows x 551 columns]
```

[117]:
```
#Applying PCA for test dataset

test_x_pca = pca.transform(test_df_newdata)
```

[118]:
```
# X_train and y Values of train data set
X_train = df_train_final
y_train = df_train['y']
```

[119]:
```
# X_test values of test data set
X_test = test_df_newdata
```

[120]:
```
#To predict your test_df values using XGBoost.
```

[138]:
```
xgb = XGBRegressor()
```

[140]:
```
xgb.fit(X_train, y_train)
```

[140]:
```
XGBRegressor(base_score=0.5, booster=None, colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints=None,
             learning_rate=0.300000012, max_delta_step=0, max_depth=6,
             min_child_weight=1, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=0, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method=None, validate_parameters=False, verbosity=None)
```

[141]:
```
pred = xgb.predict(X_test)
```

[142]:
```
pred
```

[142]:
```
array([101.00412 , 117.822685, 104.46841 , …,  96.709   , 106.94396 ,
        94.76249 ], dtype=float32)
```

[143]:
```
df_res = pd.DataFrame(pred, columns = ["yHat"])
df_res
```

[143]:

	yHat
0	101.004120
1	117.822685
2	104.468407
3	78.878845
4	111.397011
…	…
4204	106.055649
4205	90.856476
4206	96.709000
4207	106.943962
4208	94.762489

[4209 rows x 1 columns]

[ ]:
```

```