ngumbi-blaise-machine-learning

February 15, 2024

[162]:
```
#import neccesarry libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
import xgboost as xgb
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
%matplotlib inline
```

[78]:
```
# Reading the train dataset
trn = pd.read_csv('train.csv')

#Reading the test dataset
tst = pd.read_csv('test.csv')
```

[79]:
```
trn.head()
```

[79]:
```
   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]
```

[80]:
```
tst.head()
```

[80]:

```
   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]
```

[81]:

```python
#drop the ID column as index as it is not needed for prediction
trn.drop('ID',inplace=True,axis=1)
tst.drop('ID',inplace=True,axis=1)
```

[82]:

```python
# Insights:
#----------
# There are 378 cols in train data set, X columns are named as X1, X2 etc.. and␣
↪target is the Y column
# There are 377 cols in test data set , all columns except the Y column.


# Since there are lot of columns , we can use the below aggregate function to␣
↪know about the datatypes of columns

dtype_df = trn.dtypes.reset_index()
dtype_df.columns = ["feature name","dtypes"]
dtype_df.groupby("dtypes").agg("count").reset_index()
```

[82]:

```
    dtypes  feature name
0    int64           368
1  float64             1
2   object             8
```

[83]:

```python
dtype_df = tst.dtypes.reset_index()
dtype_df.columns = ["feature name","dtypes"]
dtype_df.groupby("dtypes").agg("count").reset_index()
```

[83]:

```
   dtypes  feature name
0   int64           368
1  object             8
```

[84]:
```
# Print the column if it has any null values
for i in trn.columns:
    if trn[i].isnull().sum() > 0 :
        print(trn[i].isnull().sum())
```

[85]:
```
# No features has null values in train data
```

[86]:
```
# Print the column if it has any null values
for i in tst.columns:
    if tst[i].isnull().sum() > 0 :
        print(tst[i].isnull().sum())
```

[87]:
```
# No features has null values in test data
```

[89]:
```
# Checking the unique values for each column in train data
for i in trn.columns:
    print(i,'**',trn[i].unique())
```

```
y ** [130.81  88.53  76.26 …  85.71 108.77  87.48]
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']
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']
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']
X3 ** ['a' 'e' 'c' 'f' 'd' 'b' 'g']
X4 ** ['d' 'b' 'c' 'a']
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']
X6 ** ['j' 'l' 'd' 'h' 'i' 'a' 'g' 'c' 'k' 'e' 'f' 'b']
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']
X10 ** [0 1]
X11 ** [0]
X12 ** [0 1]
X13 ** [1 0]
X14 ** [0 1]
X15 ** [0 1]
X16 ** [0 1]
X17 ** [0 1]
X18 ** [1 0]
X19 ** [0 1]
X20 ** [0 1]
X21 ** [1 0]
X22 ** [0 1]
```

```
X23 ** [0 1]
X24 ** [0 1]
X26 ** [0 1]
X27 ** [0 1]
X28 ** [0 1]
X29 ** [0 1]
X30 ** [0 1]
X31 ** [1 0]
X32 ** [0 1]
X33 ** [0 1]
X34 ** [0 1]
X35 ** [1 0]
X36 ** [0 1]
X37 ** [1 0]
X38 ** [0 1]
X39 ** [0 1]
X40 ** [0 1]
X41 ** [0 1]
X42 ** [0 1]
X43 ** [0 1]
X44 ** [0 1]
X45 ** [0 1]
X46 ** [1 0]
X47 ** [0 1]
X48 ** [0 1]
X49 ** [0 1]
X50 ** [0 1]
X51 ** [0 1]
X52 ** [0 1]
X53 ** [0 1]
X54 ** [0 1]
X55 ** [0 1]
X56 ** [0 1]
X57 ** [0 1]
X58 ** [1 0]
X59 ** [0 1]
X60 ** [0 1]
X61 ** [0 1]
X62 ** [0 1]
X63 ** [0 1]
X64 ** [0 1]
X65 ** [0 1]
X66 ** [0 1]
X67 ** [0 1]
X68 ** [1 0]
X69 ** [0 1]
X70 ** [1 0]
X71 ** [0 1]
```

```
X73 ** [0 1]
X74 ** [1 0]
X75 ** [0 1]
X76 ** [0 1]
X77 ** [0 1]
X78 ** [0 1]
X79 ** [0 1]
X80 ** [0 1]
X81 ** [0 1]
X82 ** [0 1]
X83 ** [0 1]
X84 ** [0 1]
X85 ** [1 0]
X86 ** [0 1]
X87 ** [0 1]
X88 ** [0 1]
X89 ** [0 1]
X90 ** [0 1]
X91 ** [0 1]
X92 ** [0 1]
X93 ** [0]
X94 ** [0 1]
X95 ** [0 1]
X96 ** [0 1]
X97 ** [0 1]
X98 ** [0 1]
X99 ** [0 1]
X100 ** [0 1]
X101 ** [0 1]
X102 ** [0 1]
X103 ** [0 1]
X104 ** [0 1]
X105 ** [0 1]
X106 ** [0 1]
X107 ** [0]
X108 ** [0 1]
X109 ** [0 1]
X110 ** [0 1]
X111 ** [1 0]
X112 ** [0 1]
X113 ** [0 1]
X114 ** [1 0]
X115 ** [0 1]
X116 ** [1 0]
X117 ** [0 1]
X118 ** [1 0]
X119 ** [1 0]
X120 ** [1 0]
```

```
X122 ** [0 1]
X123 ** [0 1]
X124 ** [0 1]
X125 ** [0 1]
X126 ** [0 1]
X127 ** [0 1]
X128 ** [1 0]
X129 ** [0 1]
X130 ** [0 1]
X131 ** [1 0]
X132 ** [0 1]
X133 ** [0 1]
X134 ** [0 1]
X135 ** [0 1]
X136 ** [1 0]
X137 ** [1 0]
X138 ** [0 1]
X139 ** [0 1]
X140 ** [0 1]
X141 ** [0 1]
X142 ** [1 0]
X143 ** [0 1]
X144 ** [1 0]
X145 ** [0 1]
X146 ** [0 1]
X147 ** [0 1]
X148 ** [0 1]
X150 ** [1 0]
X151 ** [0 1]
X152 ** [0 1]
X153 ** [0 1]
X154 ** [0 1]
X155 ** [0 1]
X156 ** [1 0]
X157 ** [0 1]
X158 ** [0 1]
X159 ** [0 1]
X160 ** [0 1]
X161 ** [0 1]
X162 ** [0 1]
X163 ** [0 1]
X164 ** [0 1]
X165 ** [0 1]
X166 ** [0 1]
X167 ** [0 1]
X168 ** [0 1]
X169 ** [0 1]
X170 ** [1 0]
```

X171 ** [0 1]
X172 ** [0 1]
X173 ** [0 1]
X174 ** [0 1]
X175 ** [0 1]
X176 ** [0 1]
X177 ** [0 1]
X178 ** [0 1]
X179 ** [1 0]
X180 ** [0 1]
X181 ** [0 1]
X182 ** [0 1]
X183 ** [0 1]
X184 ** [1 0]
X185 ** [0 1]
X186 ** [0 1]
X187 ** [1 0]
X189 ** [1 0]
X190 ** [0 1]
X191 ** [0 1]
X192 ** [0 1]
X194 ** [1 0]
X195 ** [0 1]
X196 ** [0 1]
X197 ** [0 1]
X198 ** [0 1]
X199 ** [0 1]
X200 ** [0 1]
X201 ** [0 1]
X202 ** [0 1]
X203 ** [0 1]
X204 ** [1 0]
X205 ** [0 1]
X206 ** [0 1]
X207 ** [0 1]
X208 ** [0 1]
X209 ** [1 0]
X210 ** [0 1]
X211 ** [0 1]
X212 ** [0 1]
X213 ** [0 1]
X214 ** [0 1]
X215 ** [0 1]
X216 ** [0 1]
X217 ** [0 1]
X218 ** [0 1]
X219 ** [0 1]
X220 ** [1 0]

```
X221 ** [0 1]
X222 ** [0 1]
X223 ** [0 1]
X224 ** [0 1]
X225 ** [0 1]
X226 ** [0 1]
X227 ** [0 1]
X228 ** [0 1]
X229 ** [0 1]
X230 ** [0 1]
X231 ** [0 1]
X232 ** [0 1]
X233 ** [0]
X234 ** [1 0]
X235 ** [0]
X236 ** [0 1]
X237 ** [1 0]
X238 ** [0 1]
X239 ** [0 1]
X240 ** [0 1]
X241 ** [0 1]
X242 ** [0 1]
X243 ** [0 1]
X244 ** [0 1]
X245 ** [0 1]
X246 ** [0 1]
X247 ** [0 1]
X248 ** [0 1]
X249 ** [0 1]
X250 ** [0 1]
X251 ** [0 1]
X252 ** [0 1]
X253 ** [0 1]
X254 ** [0 1]
X255 ** [0 1]
X256 ** [0 1]
X257 ** [0 1]
X258 ** [0 1]
X259 ** [0 1]
X260 ** [0 1]
X261 ** [0 1]
X262 ** [1 0]
X263 ** [1 0]
X264 ** [0 1]
X265 ** [0 1]
X266 ** [1 0]
X267 ** [0 1]
X268 ** [0]
```

```
X269 ** [0 1]
X270 ** [0 1]
X271 ** [0 1]
X272 ** [0 1]
X273 ** [1 0]
X274 ** [0 1]
X275 ** [1 0]
X276 ** [0 1]
X277 ** [0 1]
X278 ** [0 1]
X279 ** [0 1]
X280 ** [0 1]
X281 ** [0 1]
X282 ** [0 1]
X283 ** [0 1]
X284 ** [0 1]
X285 ** [1 0]
X286 ** [0 1]
X287 ** [0 1]
X288 ** [0 1]
X289 ** [0]
X290 ** [0]
X291 ** [0 1]
X292 ** [0 1]
X293 ** [0]
X294 ** [0 1]
X295 ** [0 1]
X296 ** [0 1]
X297 ** [0]
X298 ** [0 1]
X299 ** [0 1]
X300 ** [0 1]
X301 ** [0 1]
X302 ** [0 1]
X304 ** [0 1]
X305 ** [0 1]
X306 ** [1 0]
X307 ** [0 1]
X308 ** [0 1]
X309 ** [0 1]
X310 ** [0 1]
X311 ** [0 1]
X312 ** [0 1]
X313 ** [0 1]
X314 ** [0 1]
X315 ** [0 1]
X316 ** [1 0]
X317 ** [0 1]
```

```
X318 ** [0 1]
X319 ** [0 1]
X320 ** [0 1]
X321 ** [0 1]
X322 ** [0 1]
X323 ** [0 1]
X324 ** [1 0]
X325 ** [0 1]
X326 ** [0 1]
X327 ** [1 0]
X328 ** [0 1]
X329 ** [1 0]
X330 ** [0]
X331 ** [0 1]
X332 ** [0 1]
X333 ** [0 1]
X334 ** [1 0]
X335 ** [0 1]
X336 ** [0 1]
X337 ** [0 1]
X338 ** [0 1]
X339 ** [0 1]
X340 ** [0 1]
X341 ** [0 1]
X342 ** [0 1]
X343 ** [0 1]
X344 ** [0 1]
X345 ** [0 1]
X346 ** [0 1]
X347 ** [0]
X348 ** [0 1]
X349 ** [0 1]
X350 ** [0 1]
X351 ** [0 1]
X352 ** [0 1]
X353 ** [0 1]
X354 ** [1 0]
X355 ** [0 1]
X356 ** [0 1]
X357 ** [0 1]
X358 ** [0 1]
X359 ** [0 1]
X360 ** [0 1]
X361 ** [1 0]
X362 ** [0 1]
X363 ** [0 1]
X364 ** [0 1]
X365 ** [0 1]
```

```
X366 ** [0 1]
X367 ** [0 1]
X368 ** [0 1]
X369 ** [0 1]
X370 ** [0 1]
X371 ** [0 1]
X372 ** [0 1]
X373 ** [0 1]
X374 ** [0 1]
X375 ** [0 1]
X376 ** [0 1]
X377 ** [1 0]
X378 ** [0 1]
X379 ** [0 1]
X380 ** [0 1]
X382 ** [0 1]
X383 ** [0 1]
X384 ** [0 1]
X385 ** [0 1]
```

[91]:
```python
# Checking the unique values for each column in test data
for i in tst.columns:
    print(i,'**',tst[i].unique())
```

```
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']
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']
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']
X3 ** ['f' 'a' 'c' 'e' 'd' 'g' 'b']
X4 ** ['d' 'b' 'a' 'c']
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']
X6 ** ['a' 'g' 'j' 'l' 'i' 'd' 'f' 'h' 'c' 'k' 'e' 'b']
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']
X10 ** [0 1]
X11 ** [0 1]
X12 ** [0 1]
X13 ** [0 1]
X14 ** [0 1]
X15 ** [0 1]
X16 ** [0 1]
X17 ** [0 1]
```

```
X18 ** [0 1]
X19 ** [0 1]
X20 ** [0 1]
X21 ** [0 1]
X22 ** [0 1]
X23 ** [0 1]
X24 ** [0 1]
X26 ** [0 1]
X27 ** [1 0]
X28 ** [1 0]
X29 ** [1 0]
X30 ** [0 1]
X31 ** [1 0]
X32 ** [0 1]
X33 ** [0 1]
X34 ** [0 1]
X35 ** [1 0]
X36 ** [0 1]
X37 ** [1 0]
X38 ** [0 1]
X39 ** [0 1]
X40 ** [0 1]
X41 ** [0 1]
X42 ** [0 1]
X43 ** [1 0]
X44 ** [0 1]
X45 ** [0 1]
X46 ** [1 0]
X47 ** [0 1]
X48 ** [0 1]
X49 ** [0 1]
X50 ** [0 1]
X51 ** [0 1]
X52 ** [0 1]
X53 ** [0 1]
X54 ** [1 0]
X55 ** [0 1]
X56 ** [0 1]
X57 ** [0 1]
X58 ** [0 1]
X59 ** [0 1]
X60 ** [0 1]
X61 ** [1 0]
X62 ** [0 1]
X63 ** [0 1]
X64 ** [0 1]
X65 ** [0 1]
X66 ** [0 1]
```

```
X67 ** [0 1]
X68 ** [0 1]
X69 ** [0 1]
X70 ** [1 0]
X71 ** [0 1]
X73 ** [0 1]
X74 ** [1 0]
X75 ** [0 1]
X76 ** [1 0]
X77 ** [0 1]
X78 ** [0 1]
X79 ** [0 1]
X80 ** [1 0]
X81 ** [0 1]
X82 ** [0 1]
X83 ** [0 1]
X84 ** [0 1]
X85 ** [0 1]
X86 ** [0 1]
X87 ** [0 1]
X88 ** [0 1]
X89 ** [0 1]
X90 ** [0 1]
X91 ** [0 1]
X92 ** [0 1]
X93 ** [0 1]
X94 ** [0 1]
X95 ** [0 1]
X96 ** [1 0]
X97 ** [0 1]
X98 ** [1 0]
X99 ** [0 1]
X100 ** [0 1]
X101 ** [1 0]
X102 ** [0 1]
X103 ** [0 1]
X104 ** [0 1]
X105 ** [0 1]
X106 ** [0 1]
X107 ** [0 1]
X108 ** [0 1]
X109 ** [0 1]
X110 ** [0 1]
X111 ** [1 0]
X112 ** [0 1]
X113 ** [0 1]
X114 ** [1 0]
X115 ** [0 1]
```

```
X116 ** [0 1]
X117 ** [0 1]
X118 ** [0 1]
X119 ** [0 1]
X120 ** [0 1]
X122 ** [0 1]
X123 ** [0 1]
X124 ** [0 1]
X125 ** [0 1]
X126 ** [0 1]
X127 ** [0 1]
X128 ** [1 0]
X129 ** [0 1]
X130 ** [0 1]
X131 ** [0 1]
X132 ** [1 0]
X133 ** [0 1]
X134 ** [0 1]
X135 ** [0 1]
X136 ** [0 1]
X137 ** [0 1]
X138 ** [0 1]
X139 ** [0 1]
X140 ** [0 1]
X141 ** [0 1]
X142 ** [0 1]
X143 ** [0 1]
X144 ** [1 0]
X145 ** [0 1]
X146 ** [0 1]
X147 ** [0 1]
X148 ** [1 0]
X150 ** [1 0]
X151 ** [0 1]
X152 ** [0 1]
X153 ** [0 1]
X154 ** [0 1]
X155 ** [0 1]
X156 ** [0 1]
X157 ** [1 0]
X158 ** [1 0]
X159 ** [0 1]
X160 ** [0 1]
X161 ** [0 1]
X162 ** [1 0]
X163 ** [0 1]
X164 ** [0 1]
X165 ** [0 1]
```

```
X166 ** [1 0]
X167 ** [0 1]
X168 ** [0 1]
X169 ** [0 1]
X170 ** [0 1]
X171 ** [0 1]
X172 ** [0 1]
X173 ** [0 1]
X174 ** [0 1]
X175 ** [0 1]
X176 ** [0 1]
X177 ** [0 1]
X178 ** [0 1]
X179 ** [1 0]
X180 ** [0 1]
X181 ** [0 1]
X182 ** [0 1]
X183 ** [0 1]
X184 ** [0 1]
X185 ** [1 0]
X186 ** [0 1]
X187 ** [0 1]
X189 ** [0 1]
X190 ** [0 1]
X191 ** [0 1]
X192 ** [0 1]
X194 ** [1 0]
X195 ** [0 1]
X196 ** [0 1]
X197 ** [0 1]
X198 ** [0 1]
X199 ** [0 1]
X200 ** [0 1]
X201 ** [0 1]
X202 ** [0 1]
X203 ** [0 1]
X204 ** [1 0]
X205 ** [0 1]
X206 ** [0 1]
X207 ** [0 1]
X208 ** [0 1]
X209 ** [1 0]
X210 ** [0 1]
X211 ** [0 1]
X212 ** [0 1]
X213 ** [0 1]
X214 ** [0 1]
X215 ** [0 1]
```

```
X216 ** [0 1]
X217 ** [0 1]
X218 ** [1 0]
X219 ** [0 1]
X220 ** [1 0]
X221 ** [0 1]
X222 ** [0 1]
X223 ** [1 0]
X224 ** [0 1]
X225 ** [0 1]
X226 ** [0 1]
X227 ** [0 1]
X228 ** [0 1]
X229 ** [0 1]
X230 ** [0 1]
X231 ** [0 1]
X232 ** [1 0]
X233 ** [0 1]
X234 ** [0 1]
X235 ** [0 1]
X236 ** [0 1]
X237 ** [0 1]
X238 ** [0 1]
X239 ** [0 1]
X240 ** [0 1]
X241 ** [0 1]
X242 ** [0 1]
X243 ** [0 1]
X244 ** [0 1]
X245 ** [0 1]
X246 ** [1 0]
X247 ** [0 1]
X248 ** [0 1]
X249 ** [0 1]
X250 ** [1 0]
X251 ** [0 1]
X252 ** [0 1]
X253 ** [0 1]
X254 ** [0 1]
X255 ** [0 1]
X256 ** [1 0]
X257 ** [0]
X258 ** [0]
X259 ** [0 1]
X260 ** [0 1]
X261 ** [0 1]
X262 ** [0 1]
X263 ** [0 1]
```

```
X264 ** [0 1]
X265 ** [0 1]
X266 ** [0 1]
X267 ** [0 1]
X268 ** [0 1]
X269 ** [0 1]
X270 ** [0 1]
X271 ** [0 1]
X272 ** [1 0]
X273 ** [1 0]
X274 ** [0 1]
X275 ** [0 1]
X276 ** [1 0]
X277 ** [0 1]
X278 ** [0 1]
X279 ** [1 0]
X280 ** [0 1]
X281 ** [0 1]
X282 ** [0 1]
X283 ** [0 1]
X284 ** [0 1]
X285 ** [0 1]
X286 ** [1 0]
X287 ** [0 1]
X288 ** [0 1]
X289 ** [0 1]
X290 ** [0 1]
X291 ** [0 1]
X292 ** [0 1]
X293 ** [0 1]
X294 ** [0 1]
X295 ** [0]
X296 ** [0]
X297 ** [0 1]
X298 ** [0 1]
X299 ** [0 1]
X300 ** [0 1]
X301 ** [0 1]
X302 ** [0 1]
X304 ** [1 0]
X305 ** [0 1]
X306 ** [0 1]
X307 ** [0 1]
X308 ** [0 1]
X309 ** [0 1]
X310 ** [0 1]
X311 ** [0 1]
X312 ** [0 1]
```

```
X313 ** [0 1]
X314 ** [0 1]
X315 ** [0 1]
X316 ** [0 1]
X317 ** [0 1]
X318 ** [0 1]
X319 ** [0 1]
X320 ** [0 1]
X321 ** [0 1]
X322 ** [0 1]
X323 ** [0 1]
X324 ** [0 1]
X325 ** [0 1]
X326 ** [0 1]
X327 ** [0 1]
X328 ** [1 0]
X329 ** [0 1]
X330 ** [0 1]
X331 ** [0 1]
X332 ** [0 1]
X333 ** [0 1]
X334 ** [1 0]
X335 ** [0 1]
X336 ** [0 1]
X337 ** [0 1]
X338 ** [0 1]
X339 ** [0 1]
X340 ** [0 1]
X341 ** [0 1]
X342 ** [0 1]
X343 ** [0 1]
X344 ** [0 1]
X345 ** [0 1]
X346 ** [0 1]
X347 ** [0 1]
X348 ** [1 0]
X349 ** [0 1]
X350 ** [1 0]
X351 ** [0 1]
X352 ** [0 1]
X353 ** [0 1]
X354 ** [0 1]
X355 ** [0 1]
X356 ** [0 1]
X357 ** [0 1]
X358 ** [1 0]
X359 ** [0 1]
X360 ** [0 1]
```

```
X361 ** [1 0]
X362 ** [0 1]
X363 ** [1 0]
X364 ** [0 1]
X365 ** [0 1]
X366 ** [0 1]
X367 ** [0 1]
X368 ** [0 1]
X369 ** [0]
X370 ** [0 1]
X371 ** [0 1]
X372 ** [0 1]
X373 ** [0 1]
X374 ** [0 1]
X375 ** [0 1]
X376 ** [0 1]
X377 ** [0 1]
X378 ** [1 0]
X379 ** [0 1]
X380 ** [0 1]
X382 ** [0 1]
X383 ** [0 1]
X384 ** [0 1]
X385 ** [0 1]
```

[92]:
```
# Insights:
#----------
# From the above, we can see most of the featues are binary from X10 onwards
# Many are constants like X11,X93 etc has only values as zeroes, we will␣
↪confirm this by finding the variance below
# After confirming the variance as zero, We can drop such cols as they will not␣
↪contribute to the model
# We can also see fetures with zero variance in train are not similar with test␣
↪data - so here we have to remove the same features from test data as well
```

[93]:
```
# To identify features with 0 variance , Since we could not find variance of␣
↪categorical features, we do this after label encoding
```

[94]:
```
# Encoding Categorical features in train data - X0
# print(trn['X0'].unique())
from sklearn.preprocessing import LabelEncoder
# enc = LabelEncoder()
# trn['X0'] = enc.fit_transform(trn['X0'])
```

[95]:
```
# Applying same label encoder to test data X0 column
# tst['X0'] = enc.transform(tst['X0'])
```

[96]:
```
# With the above error, we can understand that some of values in X0 column in␣
↪test set are not presennt in train set
# and they were not encoded, hence we get error while applying on test

# Now we will identify the uncommon values
set(trn['X0'].unique()) - (set(tst['X0'].unique()))
```

[96]:
```
{'aa', 'ab', 'ac', 'q'}
```

[97]:
```
set(tst['X0'].unique()) - (set(trn['X0'].unique()))
```

[97]:
```
{'ae', 'ag', 'an', 'av', 'bb', 'p'}
```

[105]:
```
# So we get the uncommon values in both train & test data sets. Now we will␣
↪apply label encoder using the unique values
enc = LabelEncoder()
enc.fit_transform(['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',␣
↪'ae', 'ag', 'an', 'av', 'bb', 'p'])
```

[105]:
```
array([37, 24, 46, 11, 41, 49, 36, 34, 45, 40, 23, 32, 50, 51,  9, 10, 12,
       52, 43, 18, 15, 48,  6,  0, 31,  8, 30, 16, 29,  1, 26, 17, 35, 44,
       25, 22, 28, 47,  4, 19, 39, 38, 21, 14,  3, 33,  2,  5,  7, 13, 20,
       27, 42], dtype=int64)
```

[106]:
```
trn.head()
```

[106]:
```
        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]
```

[107]:
```
# Now tranform the train data using the encoder
trn['X0'] = enc.transform(trn['X0'])
```

[108]:
```
trn.head()
```

[108]:
```
        y  X0 X1  X2 X3 X4 X5 X6 X8  X10  …  X375  X376  X377  X378  X379  \
0  130.81  37  v  at  a  d  u  j  o    0  …     0     0     1     0     0
1   88.53  37  t  av  e  d  y  l  o    0  …     1     0     0     0     0
2   76.26  24  w   n  c  d  x  j  x    0  …     0     0     0     0     0
3   80.62  24  t   n  f  d  x  l  e    0  …     0     0     0     0     0
4   78.02  24  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]
```

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

[109]:
```
   X0 X1  X2 X3 X4 X5 X6 X8  X10  X11  …  X375  X376  X377  X378  X379  \
0  az  v   n  f  d  t  a  w    0    0  …     0     0     0     1     0
1   t  b  ai  a  d  b  g  y    0    0  …     0     0     1     0     0
2  az  v  as  f  d  a  j  j    0    0  …     0     0     0     1     0
3  az  l   n  f  d  z  l  n    0    0  …     0     0     0     1     0
4   w  s  as  c  d  y  i  m    0    0  …     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 376 columns]
```

[110]:
```
# Now tranform the test data using the encoder
tst['X0'] = enc.transform(tst['X0'])
```

[111]:
```
tst.head()
```

[111]:
```
   X0 X1  X2 X3 X4 X5 X6 X8  X10  X11  …  X375  X376  X377  X378  X379  \
0  24  v   n  f  d  t  a  w    0    0  …     0     0     0     1     0
1  46  b  ai  a  d  b  g  y    0    0  …     0     0     1     0     0
```

```
2  24  v  as  f  d  a  j  j    0    0  ...    0    0    0    1    0
3  24  l   n  f  d  z  l  n    0    0  ...    0    0    0    1    0
4  49  s  as  c  d  y  i  m    0    0  ...    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 376 columns]
```

[112]:
```
## From the above we can see the value 'az' in both train & test data has been
↪decoded as 24. Now this holds good.
# This can be repeated for the columns X1-X6,X8 as well
```

[113]:
```
# Now we will identify the uncommon values for X1 column
print(set(trn['X1'].unique()) - (set(tst['X1'].unique())))
print(set(tst['X1'].unique()) - (set(trn['X1'].unique())))
```

```
set()
set()
```

[114]:
```
# From the above we can see there are no uncommon values between train & test
↪data for column X1. So we can directly apply using column
enc = LabelEncoder()
trn['X1'] = enc.fit_transform(trn['X1'])
tst['X1'] = enc.transform(tst['X1'])
```

[115]:
```
# Now we will identify the uncommon values for X1 column
print(set(trn['X2'].unique()) - (set(tst['X2'].unique())))
print(set(tst['X2'].unique()) - (set(trn['X2'].unique())))
```

```
{'c', 'l', 'aa', 'o', 'ar'}
{'u', 'ad', 'ax', 'aj', 'ab', 'w'}
```

[116]:
```
# So we have some uncommon values similar to X0 column
print(trn['X2'].unique())
print(tst['X2'].unique())
```

```
['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']
['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']
```

```
[117]: # Now we define the encoder and fit the values
encX2 = LabelEncoder()
encX2.fit_transform(['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','ad',␣
 ↪'ax', 'u', 'ab', 'w', 'aj'])
```

```
[117]: array([20, 22, 38, 29, 19, 17, 42,  9, 11, 37,  0, 35,  5, 43, 30, 28,  7,
       25,  3, 16, 31, 33, 23, 48, 26, 15, 12, 32, 47, 21, 44, 14, 49,  8,
       40, 13, 34, 41,  6, 36,  1, 27, 39, 18,  4, 24, 45,  2, 46, 10],
      dtype=int64)
```

```
[118]: # Now tranform the train & test data using the encoder
trn['X2'] = encX2.transform(trn['X2'])
tst['X2'] = encX2.transform(tst['X2'])
```

```
[119]: # Now we will identify the uncommon values for X3 column
print(set(trn['X3'].unique()) - (set(tst['X3'].unique())))
print(set(tst['X3'].unique()) - (set(trn['X3'].unique())))
```

```
set()
set()
```

```
[120]: # From the above we can see there are no uncommon values between train & test␣
 ↪data for column X3. So we can directly apply using column
encX3 = LabelEncoder()
trn['X3'] = encX3.fit_transform(trn['X3'])
tst['X3'] = encX3.transform(tst['X3'])
```

```
[121]: # Now we will identify the uncommon values for X4 column
print(set(trn['X4'].unique()) - (set(tst['X4'].unique())))
print(set(tst['X4'].unique()) - (set(trn['X4'].unique())))
```

```
set()
set()
```

```
[122]: # From the above we can see there are no uncommon values between train & test␣
 ↪data for column X4. So we can directly apply using column
encX4 = LabelEncoder()
trn['X4'] = encX4.fit_transform(trn['X4'])
tst['X4'] = encX4.transform(tst['X4'])
```

```
[123]: # Now we will identify the uncommon values for X5 column
print(set(trn['X5'].unique()) - (set(tst['X5'].unique())))
print(set(tst['X5'].unique()) - (set(trn['X5'].unique())))
```

```
{'u'}
{'t', 'z', 'a', 'b'}
```

[124]:
```
# So we have some uncommon values similar to X5 column
trn['X5'].unique()
```

[124]:
```
array(['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'], dtype=object)
```

[125]:
```
# Now we define the encoder and fit the values of column X5
encX5 = LabelEncoder()
encX5.fit_transform(['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','z', 'a', 'b', 't'])
```

[125]:
```
array([27, 31, 30, 14, 13, 12, 16, 15, 11, 10,  6,  7,  2,  3,  4,  5,  8,
       18, 17, 20, 19, 22, 23, 25, 24, 28, 29, 21,  1, 32,  0,  9, 26],
      dtype=int64)
```

[126]:
```
# Now tranform the train & test data using the encoder
trn['X5'] = encX5.transform(trn['X5'])
tst['X5'] = encX5.transform(tst['X5'])
```

[127]:
```
# Now we will identify the uncommon values for X6 column
print(set(trn['X6'].unique()) - (set(tst['X6'].unique())))
print(set(tst['X6'].unique()) - (set(trn['X6'].unique())))
```

```
set()
set()
```

[128]:
```
# From the above we can see there are no uncommon values between train & test␣
 ↪data for column X6.
# So we can directly apply using column
encX6 = LabelEncoder()
trn['X6'] = encX6.fit_transform(trn['X6'])
tst['X6'] = encX6.transform(tst['X6'])
```

[130]:
```
# Now we will identify the uncommon values for X8 column
print(set(trn['X8'].unique()) - (set(tst['X8'].unique())))
print(set(tst['X8'].unique()) - (set(trn['X8'].unique())))
```

```
set()
set()
```

[131]:
```
# From the above we can see there are no uncommon values between train & test␣
↪data for column X8.
# So we can directly apply using column
encX8 = LabelEncoder()
trn['X8'] = encX8.fit_transform(trn['X8'])
tst['X8'] = encX8.transform(tst['X8'])
```

[132]:
```
trn.head()
```

[132]:
```
        y  X0  X1  X2  X3  X4  X5  X6  X8  X10  …  X375  X376  X377  X378  \
0  130.81  37  23  20   0   3  27   9  14    0  …     0     0     1     0
1   88.53  37  21  22   4   3  31  11  14    0  …     1     0     0     0
2   76.26  24  24  38   2   3  30   9  23    0  …     0     0     0     0
3   80.62  24  21  38   5   3  30  11   4    0  …     0     0     0     0
4   78.02  24  23  38   5   3  14   3  13    0  …     0     0     0     0

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

[5 rows x 377 columns]
```

[133]:
```
# Identify features with 0 variance , Since we could not find variance of␣
↪categorical features, we do this after label encoding
temp = []
for i in trn.columns:
    if trn[i].var()==0:
        temp.append(i)

print('No. of features in train data has zero variance:',len(temp))
print('List here:',temp)
```

```
No. of features in train data has zero variance: 12
List here: ['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290',
'X293', 'X297', 'X330', 'X347']
```

[134]:
```
# Dropping cols with Zero variance

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

[134]:
```
           y  X0  X1  X2  X3  X4  X5  X6  X8  X10  …  X375  X376  X377  \
0     130.81  37  23  20   0   3  27   9  14    0  …     0     0     1
1      88.53  37  21  22   4   3  31  11  14    0  …     1     0     0
```

```
2        76.26  24  24  38  2  3  30   9  23    0  ...     0     0     0
3        80.62  24  21  38  5  3  30  11   4    0  ...     0     0     0
4        78.02  24  23  38  5  3  14   3  13    0  ...     0     0     0
...        ...  ..  ..  ..  .. ..  ..  ..  ...  ...   ...   ...   ...
4204    107.39  10  20  19  2  3   1   3  16    0  ...     1     0     0
4205    108.77  36  16  44  3  3   1   7   7    0  ...     0     1     0
4206    109.22  10  23  42  0  3   1   6   4    0  ...     0     0     1
4207     87.48  11  19  29  5  3   1  11  20    0  ...     0     0     0
4208    110.85  52  19   5  2  3   1   6  22    0  ...     1     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
...    ...   ...   ...   ...   ...   ...   ...
4204     0     0     0     0     0     0     0
4205     0     0     0     0     0     0     0
4206     0     0     0     0     0     0     0
4207     0     0     0     0     0     0     0
4208     0     0     0     0     0     0     0

[4209 rows x 365 columns]
```

[135]:

```
# Since features with 0 variance are removed from train data, the same should
↪be removed from test data

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

[135]:

```
      X0  X1  X2  X3  X4  X5  X6  X8  X10  X12  ...  X375  X376  X377  X378  \
0     24  23  38   5   3  26   0  22    0    0  ...     0     0     0     1
1     46   3   9   0   3   9   6  24    0    0  ...     0     0     1     0
2     24  23  19   5   3   0   9   9    0    0  ...     0     0     0     1
3     24  13  38   5   3  32  11  13    0    0  ...     0     0     0     1
4     49  20  19   2   3  31   8  12    0    0  ...     1     0     0     0
...   ..  ..  ..  ..  ..  ..  ..  ..  ...  ...  ...   ...   ...   ...   ...
4204   9   9  19   5   3   1   9   4    0    0  ...     0     0     0     0
4205  46   1   9   3   3   1   9  24    0    0  ...     0     1     0     0
4206  51  23  19   5   3   1   3  22    0    0  ...     0     0     0     0
4207  10  23  19   0   3   1   2  16    0    0  ...     0     0     1     0
4208  46   1   9   2   3   1   6  17    0    0  ...     1     0     0     0

      X379  X380  X382  X383  X384  X385
0        0     0     0     0     0     0
1        0     0     0     0     0     0
```

```
2         0    0    0    0    0    0
3         0    0    0    0    0    0
4         0    0    0    0    0    0
…    …    …    …    …    …    …
4204      0    0    0    0    0    0
4205      0    0    0    0    0    0
4206      0    0    0    0    0    0
4207      0    0    0    0    0    0
4208      0    0    0    0    0    0

[4209 rows x 364 columns]
```

[136]:
```python
# Checking whether all features in train are numerical datatype to go before PCA
dtype_df = trn.dtypes.reset_index()
dtype_df.columns = ["feature name","dtypes"]
dtype_df.groupby("dtypes").agg("count").reset_index()
```

[136]:
```
    dtypes  feature name
0    int32             8
1    int64           368
2  float64             1
```

[141]:
```python
# Checking whether all features in test are numerical datatype to go before PCA
dtype_df = tst.dtypes.reset_index()
dtype_df.columns = ["feature name","dtypes"]
dtype_df.groupby("dtypes").agg("count").reset_index()
```

[141]:
```
  dtypes  feature name
0  int32             8
1  int64           368
```

[148]:
```python
# Dropping ID from both train & test as it will not be used by model

trnPca=trn.drop(['y'],axis=1)
tstPca=tst
```

[150]:
```python
# Perform dimensionality reduction using PCA
from sklearn.decomposition import PCA
n_comp = 12
pca = PCA(n_components=n_comp, random_state=420)
trnPca= pca.fit_transform(trnPca)
tstPca = pca.transform(tstPca)
```

[152]:
```python
trnPca.shape
```

[152]:
```
(4209, 12)
```

[153]:
```
tstPca.shape
```

[153]:
```
(4209, 12)
```

[154]:
```
# Now the total 377 X columns are reduced into 12 columns after applying PCA
```

[155]:
```
# ML Modeling with XGboost
import xgboost as xgb
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split

# Defining train & test for model input
train_X = trnPca
train_y = trn['y']

# Splitting
x_train, x_valid, y_train, y_valid = train_test_split(train_X, train_y,␣
 ↪test_size=0.2, random_state=420)

# Defining feature set
d_train = xgb.DMatrix(x_train, label=y_train)
d_valid = xgb.DMatrix(x_valid, label=y_valid)
d_test = xgb.DMatrix(tstPca)
xgb_params = {
 'n_trees': 500,
 'eta': 0.0050,
 'max_depth': 3,
 'subsample': 0.95,
 'objective': 'reg:linear',
 'eval_metric': 'rmse',
 'base_score': np.mean(train_y), # base prediction = mean(target)
 'silent': 1
}

# Creating a function for the predicting score
def xgb_r2_score(preds, dtrain):
 labels = dtrain.get_label()
 return 'r2', r2_score(labels, preds)
watchlist = [(d_train, 'train'), (d_valid, 'valid')]
mdl = xgb.train(xgb_params, d_train, 1050 , watchlist,␣
 ↪early_stopping_rounds=50, feval=xgb_r2_score, maximize=True, verbose_eval=10)
```

```
C:\Users\private admin\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
12_qbz5n2kfra8p0\LocalCache\local-packages\Python312\site-
packages\xgboost\core.py:727: FutureWarning: Pass `evals` as keyword args.
  warnings.warn(msg, FutureWarning)
C:\Users\private admin\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
```

```
12_qbz5n2kfra8p0\LocalCache\local-packages\Python312\site-
packages\xgboost\training.py:38: UserWarning: `feval` is deprecated, use
`custom_metric` instead.  They have different behavior when custom objective is
also used.See
https://xgboost.readthedocs.io/en/latest/tutorials/custom_metric_obj.html for
details on the `custom_metric`.
  warnings.warn(
C:\Users\private admin\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
12_qbz5n2kfra8p0\LocalCache\local-packages\Python312\site-
packages\xgboost\core.py:160: UserWarning: [17:14:45] WARNING: C:\buildkite-
agent\builds\buildkite-windows-cpu-autoscaling-
group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-
windows\src\objective\regression_obj.cu:209: reg:linear is now deprecated in
favor of reg:squarederror.
  warnings.warn(smsg, UserWarning)
C:\Users\private admin\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
12_qbz5n2kfra8p0\LocalCache\local-packages\Python312\site-
packages\xgboost\core.py:160: UserWarning: [17:14:45] WARNING: C:\buildkite-
agent\builds\buildkite-windows-cpu-autoscaling-
group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-windows\src\learner.cc:742:
Parameters: { "n_trees", "silent" } are not used.

  warnings.warn(smsg, UserWarning)
```

```
[0]     train-rmse:12.76632     train-r2:0.00282        valid-rmse:12.22346
valid-r2:0.00315
[10]    train-rmse:12.58994     train-r2:0.03018        valid-rmse:12.03918
valid-r2:0.03298
[20]    train-rmse:12.42778     train-r2:0.05501        valid-rmse:11.86667
valid-r2:0.06049
[30]    train-rmse:12.27615     train-r2:0.07793        valid-rmse:11.70943
valid-r2:0.08522
[40]    train-rmse:12.13594     train-r2:0.09887        valid-rmse:11.56285
valid-r2:0.10798
[50]    train-rmse:12.00744     train-r2:0.11785        valid-rmse:11.42841
valid-r2:0.12861
[60]    train-rmse:11.88739     train-r2:0.13540        valid-rmse:11.30409
valid-r2:0.14746
[70]    train-rmse:11.77693     train-r2:0.15139        valid-rmse:11.18683
valid-r2:0.16506
[80]    train-rmse:11.67307     train-r2:0.16630        valid-rmse:11.07694
valid-r2:0.18138
[90]    train-rmse:11.57470     train-r2:0.18029        valid-rmse:10.97360
valid-r2:0.19658
[100]   train-rmse:11.48251     train-r2:0.19329        valid-rmse:10.87597
valid-r2:0.21082
[110]   train-rmse:11.39772     train-r2:0.20516        valid-rmse:10.78788
valid-r2:0.22355
```

```
[120]   train-rmse:11.31858     train-r2:0.21616        valid-rmse:10.70671
valid-r2:0.23519
[130]   train-rmse:11.24455     train-r2:0.22638        valid-rmse:10.63086
valid-r2:0.24599
[140]   train-rmse:11.17504     train-r2:0.23592        valid-rmse:10.56167
valid-r2:0.25577
[150]   train-rmse:11.11121     train-r2:0.24462        valid-rmse:10.49609
valid-r2:0.26498
[160]   train-rmse:11.05126     train-r2:0.25275        valid-rmse:10.43434
valid-r2:0.27360
[170]   train-rmse:10.99549     train-r2:0.26027        valid-rmse:10.37837
valid-r2:0.28138
[180]   train-rmse:10.94018     train-r2:0.26770        valid-rmse:10.32296
valid-r2:0.28903
[190]   train-rmse:10.88566     train-r2:0.27498        valid-rmse:10.26987
valid-r2:0.29632
[200]   train-rmse:10.83248     train-r2:0.28204        valid-rmse:10.21682
valid-r2:0.30357
[210]   train-rmse:10.78167     train-r2:0.28876        valid-rmse:10.16907
valid-r2:0.31007
[220]   train-rmse:10.73572     train-r2:0.29481        valid-rmse:10.12440
valid-r2:0.31612
[230]   train-rmse:10.69270     train-r2:0.30045        valid-rmse:10.08296
valid-r2:0.32170
[240]   train-rmse:10.65222     train-r2:0.30574        valid-rmse:10.04409
valid-r2:0.32692
[250]   train-rmse:10.61439     train-r2:0.31066        valid-rmse:10.00726
valid-r2:0.33185
[260]   train-rmse:10.57420     train-r2:0.31587        valid-rmse:9.97133
valid-r2:0.33664
[270]   train-rmse:10.53794     train-r2:0.32056        valid-rmse:9.93756
valid-r2:0.34113
[280]   train-rmse:10.50124     train-r2:0.32528        valid-rmse:9.90403
valid-r2:0.34557
[290]   train-rmse:10.46465     train-r2:0.32998        valid-rmse:9.87036
valid-r2:0.35001
[300]   train-rmse:10.42880     train-r2:0.33456        valid-rmse:9.83753
valid-r2:0.35432
[310]   train-rmse:10.39135     train-r2:0.33933        valid-rmse:9.80544
valid-r2:0.35853
[320]   train-rmse:10.36112     train-r2:0.34317        valid-rmse:9.77928
valid-r2:0.36195
[330]   train-rmse:10.32914     train-r2:0.34721        valid-rmse:9.75321
valid-r2:0.36534
[340]   train-rmse:10.29690     train-r2:0.35128        valid-rmse:9.72447
valid-r2:0.36908
[350]   train-rmse:10.26592     train-r2:0.35518        valid-rmse:9.69836
valid-r2:0.37246
```

```
[360]   train-rmse:10.23622     train-r2:0.35891        valid-rmse:9.67313
valid-r2:0.37572
[370]   train-rmse:10.20392     train-r2:0.36295        valid-rmse:9.64734
valid-r2:0.37905
[380]   train-rmse:10.17518     train-r2:0.36653        valid-rmse:9.62368
valid-r2:0.38209
[390]   train-rmse:10.14940     train-r2:0.36974        valid-rmse:9.60276
valid-r2:0.38477
[400]   train-rmse:10.12067     train-r2:0.37330        valid-rmse:9.57803
valid-r2:0.38794
[410]   train-rmse:10.09396     train-r2:0.37660        valid-rmse:9.55632
valid-r2:0.39071
[420]   train-rmse:10.06958     train-r2:0.37961        valid-rmse:9.53636
valid-r2:0.39325
[430]   train-rmse:10.04528     train-r2:0.38260        valid-rmse:9.51590
valid-r2:0.39585
[440]   train-rmse:10.02172     train-r2:0.38549        valid-rmse:9.49943
valid-r2:0.39794
[450]   train-rmse:9.99827      train-r2:0.38837        valid-rmse:9.48069
valid-r2:0.40031
[460]   train-rmse:9.97815      train-r2:0.39082        valid-rmse:9.46538
valid-r2:0.40225
[470]   train-rmse:9.95523      train-r2:0.39362        valid-rmse:9.44789
valid-r2:0.40446
[480]   train-rmse:9.93555      train-r2:0.39602        valid-rmse:9.43283
valid-r2:0.40636
[490]   train-rmse:9.91534      train-r2:0.39847        valid-rmse:9.41881
valid-r2:0.40812
[500]   train-rmse:9.89562      train-r2:0.40086        valid-rmse:9.40573
valid-r2:0.40976
[510]   train-rmse:9.87671      train-r2:0.40315        valid-rmse:9.39310
valid-r2:0.41135
[520]   train-rmse:9.85611      train-r2:0.40564        valid-rmse:9.37965
valid-r2:0.41303
[530]   train-rmse:9.83743      train-r2:0.40789        valid-rmse:9.36678
valid-r2:0.41464
[540]   train-rmse:9.81871      train-r2:0.41014        valid-rmse:9.35297
valid-r2:0.41636
[550]   train-rmse:9.80216      train-r2:0.41213        valid-rmse:9.34153
valid-r2:0.41779
[560]   train-rmse:9.78552      train-r2:0.41412        valid-rmse:9.32912
valid-r2:0.41934
[570]   train-rmse:9.76758      train-r2:0.41626        valid-rmse:9.31736
valid-r2:0.42080
[580]   train-rmse:9.75133      train-r2:0.41820        valid-rmse:9.30491
valid-r2:0.42235
[590]   train-rmse:9.73539      train-r2:0.42011        valid-rmse:9.29731
valid-r2:0.42329
```

```
[600]	train-rmse:9.72005	train-r2:0.42193	valid-rmse:9.28727
valid-r2:0.42453
[610]	train-rmse:9.70620	train-r2:0.42358	valid-rmse:9.27810
valid-r2:0.42567
[620]	train-rmse:9.69138	train-r2:0.42534	valid-rmse:9.26913
valid-r2:0.42678
[630]	train-rmse:9.67671	train-r2:0.42707	valid-rmse:9.25945
valid-r2:0.42798
[640]	train-rmse:9.66220	train-r2:0.42879	valid-rmse:9.25116
valid-r2:0.42900
[650]	train-rmse:9.64637	train-r2:0.43066	valid-rmse:9.24041
valid-r2:0.43033
[660]	train-rmse:9.62982	train-r2:0.43262	valid-rmse:9.23353
valid-r2:0.43117
[670]	train-rmse:9.61581	train-r2:0.43427	valid-rmse:9.22539
valid-r2:0.43218
[680]	train-rmse:9.60232	train-r2:0.43585	valid-rmse:9.21778
valid-r2:0.43311
[690]	train-rmse:9.58969	train-r2:0.43733	valid-rmse:9.20967
valid-r2:0.43411
[700]	train-rmse:9.57619	train-r2:0.43892	valid-rmse:9.20253
valid-r2:0.43499
[710]	train-rmse:9.56421	train-r2:0.44032	valid-rmse:9.19639
valid-r2:0.43574
[720]	train-rmse:9.55281	train-r2:0.44165	valid-rmse:9.18979
valid-r2:0.43655
[730]	train-rmse:9.54163	train-r2:0.44296	valid-rmse:9.18218
valid-r2:0.43748
[740]	train-rmse:9.52881	train-r2:0.44446	valid-rmse:9.17414
valid-r2:0.43847
[750]	train-rmse:9.51660	train-r2:0.44588	valid-rmse:9.16682
valid-r2:0.43937
[760]	train-rmse:9.50471	train-r2:0.44726	valid-rmse:9.16027
valid-r2:0.44017
[770]	train-rmse:9.49032	train-r2:0.44893	valid-rmse:9.15541
valid-r2:0.44076
[780]	train-rmse:9.47812	train-r2:0.45035	valid-rmse:9.14912
valid-r2:0.44153
[790]	train-rmse:9.46445	train-r2:0.45193	valid-rmse:9.14199
valid-r2:0.44240
[800]	train-rmse:9.45327	train-r2:0.45323	valid-rmse:9.13635
valid-r2:0.44309
[810]	train-rmse:9.43921	train-r2:0.45485	valid-rmse:9.12952
valid-r2:0.44392
[820]	train-rmse:9.42726	train-r2:0.45623	valid-rmse:9.12265
valid-r2:0.44476
[830]	train-rmse:9.41428	train-r2:0.45773	valid-rmse:9.11631
valid-r2:0.44553
```

```
[840]	train-rmse:9.40109	train-r2:0.45925	valid-rmse:9.10819
valid-r2:0.44651
[850]	train-rmse:9.38566	train-r2:0.46102	valid-rmse:9.10208
valid-r2:0.44726
[860]	train-rmse:9.37510	train-r2:0.46223	valid-rmse:9.10015
valid-r2:0.44749
[870]	train-rmse:9.36275	train-r2:0.46365	valid-rmse:9.09400
valid-r2:0.44824
[880]	train-rmse:9.35296	train-r2:0.46477	valid-rmse:9.08864
valid-r2:0.44889
[890]	train-rmse:9.34291	train-r2:0.46592	valid-rmse:9.08280
valid-r2:0.44960
[900]	train-rmse:9.33454	train-r2:0.46688	valid-rmse:9.07955
valid-r2:0.44999
[910]	train-rmse:9.32275	train-r2:0.46822	valid-rmse:9.07388
valid-r2:0.45068
[920]	train-rmse:9.31317	train-r2:0.46932	valid-rmse:9.06793
valid-r2:0.45139
[930]	train-rmse:9.29970	train-r2:0.47085	valid-rmse:9.06203
valid-r2:0.45211
[940]	train-rmse:9.28690	train-r2:0.47230	valid-rmse:9.05697
valid-r2:0.45272
[950]	train-rmse:9.27279	train-r2:0.47391	valid-rmse:9.05008
valid-r2:0.45355
[960]	train-rmse:9.26134	train-r2:0.47520	valid-rmse:9.04477
valid-r2:0.45419
[970]	train-rmse:9.24692	train-r2:0.47684	valid-rmse:9.03808
valid-r2:0.45500
[980]	train-rmse:9.23432	train-r2:0.47826	valid-rmse:9.03220
valid-r2:0.45571
[990]	train-rmse:9.22130	train-r2:0.47973	valid-rmse:9.02691
valid-r2:0.45635
[1000]	train-rmse:9.21174	train-r2:0.48081	valid-rmse:9.02343
valid-r2:0.45677
[1010]	train-rmse:9.20134	train-r2:0.48198	valid-rmse:9.01705
valid-r2:0.45754
[1020]	train-rmse:9.18885	train-r2:0.48339	valid-rmse:9.01302
valid-r2:0.45802
[1030]	train-rmse:9.17817	train-r2:0.48459	valid-rmse:9.01007
valid-r2:0.45837
[1040]	train-rmse:9.16464	train-r2:0.48611	valid-rmse:9.00451
valid-r2:0.45904
[1049]	train-rmse:9.15415	train-r2:0.48728	valid-rmse:8.99823
valid-r2:0.45980
```

[156]:

```python
# Predicting on test set
p_test = mdl.predict(d_test)
```

```
p_test
```

[156]:
```
array([ 77.044106,  96.41954 ,  83.96951 , …, 102.60102 , 107.38367 ,
        96.2188  ], dtype=float32)
```

[160]:
```
Predicted_Data = pd.DataFrame()
Predicted_Data['y'] = p_test
Predicted_Data.head()
```

[160]:

	y
0	77.044106
1	96.419540
2	83.969513
3	77.468704
4	110.402405

[161]:
```
# With the above model, we have the below metrics:
# train-rmse:9.05751        train-r2:0.49805
# valid-rmse:9.00293        valid-r2:0.45923
```