MLproject1_benz_shibna

February 2, 2023

1 Importing important Libraries

[3]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

[4]:

```
train_data=pd.read_csv("train.csv")
test_data=pd.read_csv("test.csv")
```

1.0.1 Checking the no:of rows and columns each file has

[5]:

```
(train_data.shape , test_data.shape )
```

[5]:

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

[6]:

```
train_data.head(10)
```

[6]:

```
   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
5  18   92.93   t  b   e  c  d  g  h  s  …     0     0     1     0     0
6  24  128.76  al  r   e  f  d  f  h  s  …     0     0     0     0     0
7  25   91.91   o  l  as  f  d  f  j  a  …     0     0     0     0     0
8  27  108.67   w  s  as  e  d  f  i  h  …     1     0     0     0     0
9  30  126.99   j  b  aq  c  d  f  a  e  …     0     0     1     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     0     0     0     0     0
6     0     0     0     0     0
7     0     0     0     0     0
8     0     0     0     0     0
9     0     0     0     0     0

[10 rows x 378 columns]
```

[7]:
```
test_data.head()
```

[7]:
```
   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]
```

[8]:
```
train_data.columns
```

[8]:
```
Index(['ID', 'y', 'X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8',
       …
       'X375', 'X376', 'X377', 'X378', 'X379', 'X380', 'X382', 'X383', 'X384',
       'X385'],
      dtype='object', length=378)
```

[9]:
```
test_data.columns
```

[9]:
```
Index(['ID', 'X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8', 'X10',
       …
       'X375', 'X376', 'X377', 'X378', 'X379', 'X380', 'X382', 'X383', 'X384',
       'X385'],
      dtype='object', length=377)
```

1.0.2 Missing value

[10]:

```
missing_data = train_data.isnull().sum(axis=0).reset_index()
print(missing_data.info())
missing_data.columns = ['count_name', 'missing_count']
missing_data = missing_data.loc[missing_data['missing_count'] > 0]
missing_data= missing_data.sort_values(by='missing_count')
missing_data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 378 entries, 0 to 377
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   index   378 non-null    object
 1   0       378 non-null    int64
dtypes: int64(1), object(1)
memory usage: 6.0+ KB
None
```

[10]:

```
Empty DataFrame
Columns: [count_name, missing_count]
Index: []
```

No missing value

[11]:

```
missing_data = test_data.isnull().sum(axis=0).reset_index()
print(missing_data.info())
missing_data.columns = ['count_name', 'missing_count']
missing_data = missing_data.loc[missing_data['missing_count'] > 0]
missing_data = missing_data.sort_values(by='missing_count')
missing_data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 377 entries, 0 to 376
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   index   377 non-null    object
 1   0       377 non-null    int64
dtypes: int64(1), object(1)
memory usage: 6.0+ KB
None
```

[11]:

```
Empty DataFrame
Columns: [count_name, missing_count]
Index: []
```

no missing data in test data also

```
[ ]:
```

1.0.3 Checking the variance in data_train set

```
[12]: train_filter_var=np.var(train_data)
```

```
C:\Users\shibn\anaconda3\lib\site-packages\numpy\core\fromnumeric.py:3721:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise
TypeError.  Select only valid columns before calling the reduction.
  return var(axis=axis, dtype=dtype, out=out, ddof=ddof, **kwargs)
```

```
[13]: train_filter_var
```

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

```
[14]: print(train_filter_var==0)
```

```
ID      False
y       False
X10     False
X11      True
X12     False
        …
X380    False
X382    False
X383    False
X384    False
X385    False
Length: 370, dtype: bool
```

```
[15]: zero_variance_train=train_filter_var[train_filter_var==0]
```

```
[16]: zero_variance_train.shape
```

```
[16]: (12,)
```

```
[17]: print(zero_variance_train)
```

```
X11     0.0
X93     0.0
X107    0.0
X233    0.0
X235    0.0
X268    0.0
X289    0.0
X290    0.0
X293    0.0
X297    0.0
X330    0.0
X347    0.0
dtype: float64
```

```
[18]: zero_variance_train.keys()
```

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

```
[19]: zero_var_keydata=zero_variance_train.keys()
```

1.0.4 Eliminate the columns which has zero variance

```
[20]: new_train_data=train_data.drop(zero_var_keydata ,axis=1)
```

```
[21]: new_train_data.shape
```

```
[21]: (4209, 366)
```

1.0.5 to check which all columns are dtype=object

```
[22]: new_train_data.dtypes==object
```

```
[22]: ID       False
      y        False
      X0        True
      X1        True
      X2        True
               …
      X380     False
```

```
X382    False
X383    False
X384    False
X385    False
Length: 366, dtype: bool
```

```
[23]: x=(new_train_data.dtypes==object)
```

```
[24]: x.value_counts()
```

```
[24]: False    358
      True       8
      dtype: int64
```

```
[25]: object_train_cols = list(x[x].index)
```

```
[26]: print(object_train_cols)
```

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

```
[ ]:
```

1.0.6 Checking the variance in data_test set

```
[27]: ### using sklearn
```

```
[28]: #from sklearn.feature_selection import VarianceThreshold
      #selector = VarianceThreshold(threshold=0)
      #selector.fit_transform(test_data)
```

```
[29]: test_filter_var=np.var(test_data)
```

```
C:\Users\shibn\anaconda3\lib\site-packages\numpy\core\fromnumeric.py:3721:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise
TypeError.  Select only valid columns before calling the reduction.
  return var(axis=axis, dtype=dtype, out=out, ddof=ddof, **kwargs)
```

```
[30]: test_filter_var
```

```
[30]: ID      5.869917e+06
      X10     1.864563e-02
      X11     2.375297e-04
      X12     6.883438e-02
      X13     5.733136e-02
                  …
```

```
X380    8.012675e-03
X382    8.713410e-03
X383    4.749465e-04
X384    7.122504e-04
X385    1.660337e-03
Length: 369, dtype: float64
```

```
[31]: print(test_filter_var==0)
```

```
ID      False
X10     False
X11     False
X12     False
X13     False
        …
X380    False
X382    False
X383    False
X384    False
X385    False
Length: 369, dtype: bool
```

```
[32]: zero_test_var=test_filter_var[test_filter_var==0]
```

```
[33]: zero_test_var.shape
```

```
[33]: (5,)
```

```
[34]: zero_test_var.keys()
```

```
[34]: Index(['X257', 'X258', 'X295', 'X296', 'X369'], dtype='object')
```

```
[35]: zero_var_cols=zero_test_var.keys()
```

```
[36]: new_test_data=test_data.drop(zero_var_cols,axis=1)
```

```
[37]: new_test_data.shape
```

```
[37]: (4209, 372)
```

1.0.7 to check which all columns are object data type

```
[38]: x=(new_test_data.dtypes==object)
```

```
[39]: x.value_counts()
```

```
[39]: False    364
      True       8
      dtype: int64
```

```
[40]: object_test_cols = list(x[x].index)
```

```
[41]: print(object_test_cols)
```

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

2 Applying label encoding

```
[42]: from sklearn.preprocessing import LabelEncoder
      label_Encoder=LabelEncoder()
```

```
[43]: for i in object_train_cols:
          new_train_data[i]=label_Encoder.fit_transform(new_train_data[i])
          #new_data_test[i]=label_Encoder.fit_transform(new_test_data[i])
```

```
[44]: t=new_train_data.dtypes==object
```

```
[45]: t.value_counts()
```

```
[45]: False    366
      dtype: int64
```

The above step is to confirm that the label encoder have converted everything to integer that is why no true value is found as per the condition given any datatypes is equal to object

```
[46]: new_train_data.head()
```

```
[46]:    ID       y  X0  X1  X2  X3  X4  X5  X6  X8  …  X375  X376  X377  X378  \
      0   0  130.81  32  23  17   0   3  24   9  14  …     0     0     1     0
      1   6   88.53  32  21  19   4   3  28  11  14  …     1     0     0     0
      2   7   76.26  20  24  34   2   3  27   9  23  …     0     0     0     0
      3   9   80.62  20  21  34   5   3  27  11   4  …     0     0     0     0
      4  13   78.02  20  23  34   5   3  12   3  13  …     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 366 columns]
```

[47]:

```python
new_train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4209 entries, 0 to 4208
Columns: 366 entries, ID to X385
dtypes: float64(1), int32(8), int64(357)
memory usage: 11.6 MB
```

from the above, we came to know that the data is converted to numerical

Applying label encoding to test data

[48]:

```python
for i in object_test_cols:
    #new_train_data[i]=label_Encoder.fit_transform(new_train_data[i])
    new_test_data[i]=label_Encoder.fit_transform(new_test_data[i])
```

[49]:

```python
t=new_test_data.dtypes==object
```

[50]:

```python
t.value_counts()
```

[50]:

```
False    372
dtype: int64
```

The above step is to confirm that the label encoder have converted everything to integer that vis why no true value is found as per the condition given any datatypes is equal to object

[51]:

```python
new_test_data.head()
```

[51]:

```
   ID  X0  X1  X2  X3  X4  X5  X6  X8  X10  …  X375  X376  X377  X378  X379  \
0   1  21  23  34   5   3  26   0  22    0  …     0     0     0     1     0
1   2  42   3   8   0   3   9   6  24    0  …     0     0     1     0     0
2   3  21  23  17   5   3   0   9   9    0  …     0     0     0     1     0
3   4  21  13  34   5   3  31  11  13    0  …     0     0     0     1     0
4   5  45  20  17   2   3  30   8  12    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 372 columns]
```

[52]:

```
new_test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4209 entries, 0 to 4208
Columns: 372 entries, ID to X385
dtypes: int32(8), int64(364)
memory usage: 11.8 MB
```

from the above, we came to know that the data is converted to numerical

2.0.1 last and final stage to split as feature and target

[53]:

```
train_data_Xfeatures=new_train_data.drop(["ID","y"],axis=1)
train_data_ytarget=new_train_data['y']
```

[71]:

```
idOf_test=new_test_data['ID'].values
test_data_Xfeatures=new_test_data.drop(["ID"],axis=1)
```

3 Perform dimensionality reduction

[73]:

```
# Linear dimensionality reduction using Singular Value Decomposition of
# the data to project it to a lower dimensional space.
from sklearn.decomposition import PCA
pca=PCA(n_components=12,random_state=46)
train_data_Xfeatures=pca.fit_transform(train_data_Xfeatures)
test_data_Xfeatures=pca.fit_transform(test_data_Xfeatures)
```

4 Training using XGBoost

[74]:

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

[75]:

```
X_train,X_test,y_train,y_test=train_test_split(train_data_Xfeatures,train_data_ytarget,test_si
 ↪3,random_state=42)
```

[76]:

```
print(X_train.shape,y_train.shape)
```

```
(2946, 12) (2946,)
```

[77]:

```
print(X_test.shape,y_test.shape)
```

```
(1263, 12) (1263,)
```

[78]:

```
xgb1 = XGBRegressor()
parameters = {'nthread':[4], #when use hyperthread, xgboost may become slower
              'objective':['reg:linear'],
              'learning_rate': [0.1,0.2,.03, 0.05, .07], #so called `eta` value
              'max_depth': [1,2,3,4,5, 6, 7],
              'min_child_weight': [4],
              'silent': [1],
              'subsample': [0.7],
              'colsample_bytree': [0.7],
              'n_estimators': [500]}
```

[79]:

```
xgb_grid = GridSearchCV(xgb1,
                        parameters,
                        cv = 3,
                        n_jobs = 4,
                        verbose=True)


xgb_grid.fit(X_train,
         y_train)
```

```
Fitting 3 folds for each of 35 candidates, totalling 105 fits
[21:32:30] WARNING: C:/buildkite-agent/builds/buildkite-windows-cpu-autoscaling-
group-i-03de431ba26204c4d-1/xgboost/xgboost-ci-
windows/src/objective/regression_obj.cu:213: reg:linear is now deprecated in
favor of reg:squarederror.
[21:32:30] WARNING: C:/buildkite-agent/builds/buildkite-windows-cpu-autoscaling-
group-i-03de431ba26204c4d-1/xgboost/xgboost-ci-windows/src/learner.cc:767:
Parameters: { "silent" } are not used.
```

[79]:

```
GridSearchCV(cv=3,
             estimator=XGBRegressor(base_score=None, booster=None,
                                    callbacks=None, colsample_bylevel=None,
                                    colsample_bynode=None,
                                    colsample_bytree=None,
                                    early_stopping_rounds=None,
                                    enable_categorical=False, eval_metric=None,
                                    feature_types=None, gamma=None, gpu_id=None,
                                    grow_policy=None, importance_type=None,
                                    interaction_constraints=None,
                                    learning_rate=None, m…
                                    monotone_constraints=None, n_estimators=100,
                                    n_jobs=None, num_parallel_tree=None,
                              predictor=None, random_state=None, …),
             n_jobs=4,
             param_grid={'colsample_bytree': [0.7],
                         'learning_rate': [0.1, 0.2, 0.03, 0.05, 0.07],
                         'max_depth': [1, 2, 3, 4, 5, 6, 7],
                         'min_child_weight': [4], 'n_estimators': [500],
                         'nthread': [4], 'objective': ['reg:linear'],
                         'silent': [1], 'subsample': [0.7]},
             verbose=True)
```

[80]:

```
print(xgb_grid.best_score_)
print(xgb_grid.best_params_)
```

```
0.4693980501828196
{'colsample_bytree': 0.7, 'learning_rate': 0.03, 'max_depth': 4,
'min_child_weight': 4, 'n_estimators': 500, 'nthread': 4, 'objective':
'reg:linear', 'silent': 1, 'subsample': 0.7}
```

[81]:

```
#data_dmatrix = xgb.DMatrix(data=X,label=y)
```

[82]:

```
#from xgboost.sklearn import XGBRegressor
#from xgboost import XGBRegressor
#xg_reg = xgb1(objective="reg:linear", learning_rate = 0.03, max_depth = 4,␣
 ↪min_child_weight = 4, n_estimators = 500, subsample = 0.7)
```

4.0.1 Training using XGBoost

[83]:

```
d_train = xgb.DMatrix(X_train, label=y_train)
d_valid = xgb.DMatrix(X_test, label=y_test)
d_test = xgb.DMatrix(test_data_Xfeatures)
```

[84]:

```
params={}
params['objective'] = 'reg:linear'
params['eta'] = 0.02
params['max_depth'] = 4
def xgb_r2_score(preds, dtrain):
    labels = dtrain.get_label()
    return 'r2', r2_score(labels, preds)
watchlist = [(d_train, 'train'), (d_valid, 'valid')]
clf = xgb.train(params, d_train, 1000, watchlist, early_stopping_rounds=50,␣
 ↪feval=xgb_r2_score, maximize=True, verbose_eval=10)
```

```
[21:32:32] WARNING: C:/buildkite-agent/builds/buildkite-windows-cpu-autoscaling-
group-i-03de431ba26204c4d-1/xgboost/xgboost-ci-
windows/src/objective/regression_obj.cu:213: reg:linear is now deprecated in
favor of reg:squarederror.
```

```
[0]     train-rmse:98.73858     train-r2:-62.78156      valid-rmse:99.53598
valid-r2:-54.41767
[10]    train-rmse:80.91897     train-r2:-41.83730      valid-rmse:81.73402
valid-r2:-36.36748
[20]    train-rmse:66.39288     train-r2:-27.83795      valid-rmse:67.24898
valid-r2:-24.29644
```

```
C:\Users\shibn\anaconda3\lib\site-packages\xgboost\core.py:617: FutureWarning:
Pass `evals` as keyword args.
  warnings.warn(msg, FutureWarning)
C:\Users\shibn\anaconda3\lib\site-packages\xgboost\training.py:39: 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(
```

```
[30]    train-rmse:54.56281     train-r2:-18.47668      valid-rmse:55.47389
valid-r2:-16.21335
[40]    train-rmse:44.94648     train-r2:-12.21639      valid-rmse:45.91759
valid-r2:-10.79359
[50]    train-rmse:37.15140     train-r2:-8.02967       valid-rmse:38.19713
valid-r2:-7.16111
[60]    train-rmse:30.85581     train-r2:-5.22867       valid-rmse:31.98680
valid-r2:-4.72308
[70]    train-rmse:25.78684     train-r2:-3.35029       valid-rmse:27.01631
valid-r2:-3.08263
[80]    train-rmse:21.72926     train-r2:-2.08895       valid-rmse:23.06947
valid-r2:-1.97689
[90]    train-rmse:18.50854     train-r2:-1.24112       valid-rmse:19.97030
valid-r2:-1.23078
[100]   train-rmse:15.97889     train-r2:-0.67038       valid-rmse:17.56292
valid-r2:-0.72537
[110]   train-rmse:14.01209     train-r2:-0.28448       valid-rmse:15.71586
valid-r2:-0.38154
[120]   train-rmse:12.50901     train-r2:-0.02369       valid-rmse:14.33608
valid-r2:-0.14961
[130]   train-rmse:11.36021     train-r2:0.15571        valid-rmse:13.31214
valid-r2:0.00875
[140]   train-rmse:10.50934     train-r2:0.27744        valid-rmse:12.56300
valid-r2:0.11718
[150]   train-rmse:9.87095      train-r2:0.36256        valid-rmse:12.02727
valid-r2:0.19086
[160]   train-rmse:9.39646      train-r2:0.42237        valid-rmse:11.63483
valid-r2:0.24280
[170]   train-rmse:9.04274      train-r2:0.46504        valid-rmse:11.35249
valid-r2:0.27911
[180]   train-rmse:8.78203      train-r2:0.49544        valid-rmse:11.15587
valid-r2:0.30386
```

```
[190]   train-rmse:8.58170      train-r2:0.51820        valid-rmse:11.01577
valid-r2:0.32124
[200]   train-rmse:8.44132      train-r2:0.53383        valid-rmse:10.92084
valid-r2:0.33289
[210]   train-rmse:8.33111      train-r2:0.54592        valid-rmse:10.84662
valid-r2:0.34192
[220]   train-rmse:8.23994      train-r2:0.55581        valid-rmse:10.79402
valid-r2:0.34829
[230]   train-rmse:8.15909      train-r2:0.56448        valid-rmse:10.75521
valid-r2:0.35297
[240]   train-rmse:8.09280      train-r2:0.57153        valid-rmse:10.73246
valid-r2:0.35570
[250]   train-rmse:8.03866      train-r2:0.57725        valid-rmse:10.71860
valid-r2:0.35736
[260]   train-rmse:7.98870      train-r2:0.58248        valid-rmse:10.70360
valid-r2:0.35916
[270]   train-rmse:7.95001      train-r2:0.58652        valid-rmse:10.69745
valid-r2:0.35990
[280]   train-rmse:7.91634      train-r2:0.59001        valid-rmse:10.69069
valid-r2:0.36071
[290]   train-rmse:7.88756      train-r2:0.59299        valid-rmse:10.68239
valid-r2:0.36170
[300]   train-rmse:7.84931      train-r2:0.59693        valid-rmse:10.67797
valid-r2:0.36223
[310]   train-rmse:7.81953      train-r2:0.59998        valid-rmse:10.67649
valid-r2:0.36240
[320]   train-rmse:7.78954      train-r2:0.60304        valid-rmse:10.67590
valid-r2:0.36247
[330]   train-rmse:7.76483      train-r2:0.60556        valid-rmse:10.67846
valid-r2:0.36217
[340]   train-rmse:7.73786      train-r2:0.60829        valid-rmse:10.67514
valid-r2:0.36257
[350]   train-rmse:7.71039      train-r2:0.61107        valid-rmse:10.67867
valid-r2:0.36214
[360]   train-rmse:7.67913      train-r2:0.61421        valid-rmse:10.67556
valid-r2:0.36252
[370]   train-rmse:7.64938      train-r2:0.61720        valid-rmse:10.67616
valid-r2:0.36244
[380]   train-rmse:7.61585      train-r2:0.62055        valid-rmse:10.67623
valid-r2:0.36244
[390]   train-rmse:7.58177      train-r2:0.62394        valid-rmse:10.67473
valid-r2:0.36262
[400]   train-rmse:7.55057      train-r2:0.62702        valid-rmse:10.67351
valid-r2:0.36276
[410]   train-rmse:7.52849      train-r2:0.62920        valid-rmse:10.67617
valid-r2:0.36244
[420]   train-rmse:7.50319      train-r2:0.63169        valid-rmse:10.67677
valid-r2:0.36237
```

```
[430]   train-rmse:7.47070      train-r2:0.63487        valid-rmse:10.67192
valid-r2:0.36295
[440]   train-rmse:7.44340      train-r2:0.63754        valid-rmse:10.66981
valid-r2:0.36320
[450]   train-rmse:7.42140      train-r2:0.63968        valid-rmse:10.67071
valid-r2:0.36310
[460]   train-rmse:7.39629      train-r2:0.64211        valid-rmse:10.66964
valid-r2:0.36322
[470]   train-rmse:7.37310      train-r2:0.64435        valid-rmse:10.66897
valid-r2:0.36330
[480]   train-rmse:7.34658      train-r2:0.64690        valid-rmse:10.66844
valid-r2:0.36337
[490]   train-rmse:7.32267      train-r2:0.64920        valid-rmse:10.66970
valid-r2:0.36322
[500]   train-rmse:7.30167      train-r2:0.65121        valid-rmse:10.66919
valid-r2:0.36328
[510]   train-rmse:7.27660      train-r2:0.65360        valid-rmse:10.67193
valid-r2:0.36295
[520]   train-rmse:7.25104      train-r2:0.65603        valid-rmse:10.67042
valid-r2:0.36313
[526]   train-rmse:7.23652      train-r2:0.65741        valid-rmse:10.67395
valid-r2:0.36271
```

4.0.2 Predict the test data values using XGBoost

[85]:

```
pred_test=clf.predict(d_test)
prediction_data=pd.DataFrame()
prediction_data['ID']=idOf_test
prediction_data['y']=pred_test
prediction_data.to_csv('predXGB.csv', index=False)
prediction_data.head()
```

[85]:

```
   ID           y
0   1   77.342758
1   2   97.923904
2   3   83.637695
3   4   78.497604
4   5  110.838745
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[86]:

```
#params = {'n_estimators':[10, 20, 40, 80], 'max_depth':[1,2,3,6,10],␣
 ↪'learning_rate' :[0.1, 0.2, 0.3, 0.5], 'min_child_weight' : [1, 2, 3, 4, 5],␣
 ↪'subsample' : [0.5, 0.6, 0.7, 0.8, 1.0]}
#grid_search = GridSearchCV(xgb_clf, params, cv = 3, n_jobs = -1)
#grid_search.fit(X_train, y_train)
```

[87]:

```
#grid_search.best_params_
```

[ ]:

```
#start = time.time()
#xgb_clf.fit(X_train, y_train)
#end = time.time()

#time_elapsed = end - start
#print(time_elapsed)
```

[ ]:

```
#y_pred = xgb_clf.predict(X_test)
```

[ ]:

```
#dtrain = xgb.DMatrix(X_train, label=y_train)
#dtest = xgb.DMatrix(X_test, label=y_test)
```

[ ]:

[ ]:

[ ]: