IMPORT DATASETS

In [8]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from tqdm import tqdm
import warnings
warnings.filterwarnings('ignore')
import time
from sklearn.ensemble import RandomForestRegressor,AdaBoostRegressor, GradientBoostingRegressor , StackingRegressor
from sklearn.feature_selection import SelectFromModel
from sklearn.model_selection import train_test_split,GridSearchCV, RandomizedSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error,  make_scorer
from sklearn.model_selection import KFold,  cross_val_score,  RepeatedKFold
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, PowerTransformer,  StandardScaler
from scipy.stats import kurtosis, skew
from scipy.special import boxcox, inv_boxcox
from xgboost import XGBRFRegressor, Booster
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import Lasso, Ridge, ElasticNet
from sklearn.svm import SVR
from lightgbm import LGBMRegressor
from sklearn.neighbors import KNeighborsRegressor
from prettytable import PrettyTable
import joblib
from sklearn.pipeline import Pipeline
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [4]:
```
def r2_score_transform(train_pred, test_pred, train_label, test_label):
```

```
  '''
  This function convert data into inverse yeo-johnson transformation and give a r2 score
  Input - train and test predication array, train and test label
  Output - inverse transform the array and compute the R2 score

   '''
  # Inverse transform the arrays
  a, b    = pt.inverse_transform(train_pred.reshape(-1,1)), pt.inverse_transform(test_pred.reshape(-1,1))
  x, y    = pt.inverse_transform(train_label.reshape(-1,1)), pt.inverse_transform(test_label.reshape(-1,1))

  # get r2 score of train and test data
  train = r2_score(x,a)
  test  = r2_score(y,b)

  return train, test
```

In [5]:

```
def kfold_grid_search(clf, params, train, label, fold, kfold, search = 'grid'):

  '''
  This function compute the grid or random search cross validation and taken best estimator compute
  R^2 score with kfold validation.
  Input :
    clf = model
    params = dict of parameters for ranfom or grid search
    train = train dataset
    label = train data label
    fold = grid or random cross validation folds
    kfold = fold for kfold CV
    search  = grid or random

  '''


  start = time.time()

  # declare the grid search
  if search == 'grid' :
    clf_grid = GridSearchCV(estimator = clf,
                        param_grid = params,
                        n_jobs = -1,
                        scoring = 'r2',
                        cv = fold,
```

```
                        verbose = 1)
    # fit data into the grid model
    clf_grid.fit(train, label)

    # best estimator for the model
    print('\n')
    print(f'Best Estimator : {clf_grid.best_estimator_}')
    print('\n')

    # Model as best estimator
    clf = clf_grid.best_estimator_

  # declre the random search
  elif search == 'random' :
    clf_grid = RandomizedSearchCV(estimator = clf,
                        param_distributions = params,
                        n_jobs = -1,
                        scoring = 'r2',
                        cv = fold,
                        verbose = 1,
                        )
    # fit data into the grid model
    clf_grid.fit(train, label)

    # best estimator for the model
    print('\n')
    print(f'Best Estimator : {clf_grid.best_estimator_}')
    print('\n')

    # Model as best estimator
    clf = clf_grid.best_estimator_

  else:
    clf = clf


  # genrate folds
  kfold = KFold(n_splits= kfold, random_state= 42)

  tr = [ ]
  te = [ ]
  i = 0
```

```
print('R2 metric : ')
for train_in, test_in in kfold.split(train):

  # set indices of train and test data
  xtrain, xtest = train.iloc[train_in], train.iloc[test_in]
  ytrain, ytest = label[train_in], label[test_in]

  # set best estimator as model
  model = clf

  # fit data into the best estimator
  model.fit(xtrain, ytrain)

  # predict in train and test data
  train_pred = model.predict(xtrain)
  test_pred = model.predict(xtest)

   # get r2 score of train and test data
  tr_pre  = r2_score(ytrain, train_pred)
  te_pre  = r2_score(ytest, test_pred)

  tr.append(tr_pre)
  te.append(te_pre)

  print(f'{i} r2 score of the train data {tr_pre} and test data {te_pre}')

  i += 1

print(f'\nAvg r2_score of train {np.mean(tr)} and test {np.mean(te)}')


end = time.time()
print(f'Time taken - {(end-start)/60} min')
```

Models with PCA + Synthetic and Binary features

- This datasets contain PCA (5) features, Binary features, label encoded features and created synthetic features.

In [ ]:
```
# Import the train and test csv files

train = pd.read_csv(r'/content/drive/MyDrive/Datasets/pca+feature_train.csv')
test = pd.read_csv(r'/content/drive/MyDrive/Datasets/pca+feature_test.csv')

train = train.drop(['Unnamed: 0'], axis = 1)
test = test.drop(['Unnamed: 0'], axis = 1)
train.head()
```

Out[ ]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	X20	X21	X22	X23	X24	X26	X27	X28
0	0	130.81	k	v	at	a	d	u	j	o	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0
1	6	88.53	k	t	av	e	d	y	l	o	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
2	7	76.26	az	w	n	c	d	x	j	x	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1
3	9	80.62	az	t	n	f	d	x	l	e	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
4	13	78.02	az	v	n	f	d	h	d	n	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

5 rows × 404 columns

In [ ]:
```
#Check the Shape of the test and train data
print('train data shape - ', train.shape)
print('test data shape  - ', test.shape)
```

```
train data shape -  (4209, 404)
test data shape  -  (4209, 403)
```

In [ ]:
```
# declare the train set and label
y2 = train.y
x2 = train.iloc[:, 10:]
```

In [ ]:
```
# Split train data into train-set and test-set
x_train2, x_test2, y_train2, y_test2 = train_test_split(x2,y2, test_size = 0.33,random_state= 42)

print(x_train2.shape, y_train2.shape)
print(x_test2.shape, y_test2.shape)
```

```
(2820, 394) (2820,)
(1389, 394) (1389,)
```

Decision Tree

In [ ]:

```
clf = DecisionTreeRegressor()

params = {'max_depth' : [2,3,4,8,10,15],
          'max_features' : ['auto', 'sqrt', 'log2'],
          'random_state' : [5,10,20,30],
          }

kfold_grid_search(clf, params, x2, y2, 10, kfold = 15, search= 'random') #(0.5780)
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:    3.9s finished

Best Estimator : DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=8,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=10, splitter='best')


R2 metric :
0 r2 score of the train data 0.6813107472320485 and test data 0.5767082371099517
1 r2 score of the train data 0.621586039510241 and test data 0.708513052109607
2 r2 score of the train data 0.630831290790349 and test data 0.5631604647843083
3 r2 score of the train data 0.6668606818390898 and test data 0.2787127830840411
4 r2 score of the train data 0.6380355155912681 and test data 0.06995160992884453
5 r2 score of the train data 0.6311910610649891 and test data 0.5245457377028628
6 r2 score of the train data 0.63061835338783 and test data 0.6205419720909853
7 r2 score of the train data 0.6221887781115217 and test data 0.6688339348247292
8 r2 score of the train data 0.6419099223480337 and test data 0.46515590899698256
9 r2 score of the train data 0.6279626162710936 and test data 0.5227980632203781
10 r2 score of the train data 0.6308182124521897 and test data 0.4505968373538033
11 r2 score of the train data 0.6259084033271662 and test data 0.5516506258243561
12 r2 score of the train data 0.622938628458103 and test data 0.5444741383056872
13 r2 score of the train data 0.6294581983558882 and test data 0.5987995140027604
```

```
14 r2 score of the train data 0.6261923851524602 and test data 0.6724568866375927

Avg r2_score of train 0.6351873889261516 and test 0.5211266510651261
Time taken - 0.09999754031499226 min
```

In [ ]:

```
dt = DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=3,
                           max_features='auto',
                           min_impurity_decrease=0.0,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, presort='deprecated',
                           splitter='best')


a = cross_val_score(dt, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.5734756571589295 data
```

Random Forest

In [ ]:

```
clf = RandomForestRegressor()

params = {'n_estimators':[40,50,60,70,100],
             'max_depth':[3,5,6,7,8],
             'min_samples_split':[2,3,4,5,6,7,8,9,10],
             'max_features': [0.80,.95, 1.0],
             'min_samples_leaf': [1, 2,3,4,5,6,7,8,9],
             'min_impurity_decrease':[1e-5,1e-4,1e-3,1e-2,1e-1,0,1,10]}

kfold_grid_search(clf, params, x2, y2, fold = 10, kfold=15, search= 'random' )
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  2.8min finished

Best Estimator : RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=3, max_features=0.8, max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=1e-05,
                      min_impurity_split=None, min_samples_leaf=5,
                      min_samples_split=9, min_weight_fraction_leaf=0.0,
```

```
                       n_estimators=40, n_jobs=None, oob_score=False,
                       random_state=None, verbose=0, warm_start=False)


R2 metric :
0 r2 score of the train data 0.5715153440871907 and test data 0.6075821902602618
1 r2 score of the train data 0.5674883076551979 and test data 0.6511201254823615
2 r2 score of the train data 0.5730553993723171 and test data 0.6021445241149124
3 r2 score of the train data 0.60601763083342 and test data 0.30708047504635183
4 r2 score of the train data 0.5773519833442724 and test data 0.5444356633636224
5 r2 score of the train data 0.575621620235263 and test data 0.5670822703371299
6 r2 score of the train data 0.5696064626862604 and test data 0.6596220214589319
7 r2 score of the train data 0.5671942422662717 and test data 0.7064270195874032
8 r2 score of the train data 0.5826081962660211 and test data 0.46894952095071496
9 r2 score of the train data 0.5748447938417933 and test data 0.5578146764787673
10 r2 score of the train data 0.5786768813257471 and test data 0.4979097062115938
11 r2 score of the train data 0.5727663404360401 and test data 0.5977121928418523
12 r2 score of the train data 0.569489839201403 and test data 0.6548764845232085
13 r2 score of the train data 0.5702566085082328 and test data 0.6372309483627063
14 r2 score of the train data 0.5687708169055417 and test data 0.6442986162389157

Avg r2_score of train 0.5750176311309981 and test 0.5802857623505823
Time taken - 3.083255358537038 min
```

In [ ]:

```
rf = RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                       max_depth=3, max_features=0.8, max_leaf_nodes=None,
                       max_samples=None, min_impurity_decrease=1e-05,
                       min_impurity_split=None, min_samples_leaf=5,
                       min_samples_split=9, min_weight_fraction_leaf=0.0,
                       n_estimators=40, n_jobs=None, oob_score=False,
                       random_state=None, verbose=0, warm_start=False)

a = cross_val_score(rf, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.5745664501511195 data
```

XGBoost-Regressor

In [ ]:

```
clf = XGBRFRegressor(silent=True)

xparams = {'learning_rate':[0.1,0.5,0.8,1],
```

```
              'n_estimators':[70,80,100],
              'max_depth':[2,3,4],
              'colsample_bytree':[0.1,0.5,0.7,0.9,1],
              'subsample':[0.2,0.3,0.5,1],
              'gamma':[0.0001,0.001,0,0.1,0.01,0.5,1],
              'reg_alpha':[0.00001,0.0001,0.001,0.01,0.1]}


kfold_grid_search(clf, params, x2, y2, fold = 10, kfold=15, search= 'random')
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  3.5min finished

Best Estimator : XGBRFRegressor(base_score=0.5, colsample_bylevel=1, colsample_bynode=0.8,
               colsample_bytree=1, gamma=0, learning_rate=1, max_delta_step=0,
               max_depth=15, max_features='log2', min_child_weight=1,
               missing=None, n_estimators=100, n_jobs=1, nthread=None,
               objective='reg:linear', random_state=5, reg_alpha=0,
               reg_lambda=1, scale_pos_weight=1, seed=None, silent=True,
               subsample=0.8, verbosity=1)


R2 metric :
0 r2 score of the train data 0.5591106706797955 and test data 0.5969348853865986
1 r2 score of the train data 0.5567761302990156 and test data 0.6392482807896993
2 r2 score of the train data 0.5589944292139439 and test data 0.6035569639748404
3 r2 score of the train data 0.5925375912978497 and test data 0.3071454665475717
4 r2 score of the train data 0.5640973355720406 and test data 0.5338131420792587
5 r2 score of the train data 0.5621640135494697 and test data 0.5595743834445313
6 r2 score of the train data 0.5563947957633792 and test data 0.6555906505571116
7 r2 score of the train data 0.553735574181428 and test data 0.7032605078583825
8 r2 score of the train data 0.5690729048532857 and test data 0.4700063619141459
9 r2 score of the train data 0.5620211189566908 and test data 0.5543351501847651
10 r2 score of the train data 0.5666264217016058 and test data 0.48986189262393154
11 r2 score of the train data 0.560688699269444 and test data 0.5770509865715794
12 r2 score of the train data 0.5561268382448219 and test data 0.6574833593687027
13 r2 score of the train data 0.5575377447394365 and test data 0.6320507247078625
14 r2 score of the train data 0.5558804535718433 and test data 0.6482600205565894

Avg r2_score of train 0.56211764812627 and test 0.5752115184377047
Time taken - 4.294117295742035 min
```

In [ ]:
```
 xg = XGBRFRegressor(colsample_bylevel=1,
                colsample_bynode=0.8, colsample_bytree=1, gamma=0,
                learning_rate=1, max_delta_step=0, max_depth=5,
                max_features=0.95, min_child_weight=1, min_impurity_decrease=1,
                min_samples_leaf=1, min_samples_split=5, missing=None,
                n_estimators=100, n_jobs=1, nthread=None, objective='reg:linear',
                random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
                seed=None, silent=True, subsample=0.8, verbosity=1)

a = cross_val_score(xg, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.5718237374738755 data
```

AdaBoost-Regressor

In [ ]:
```
clf = AdaBoostRegressor()

params = {'n_estimators'  : [100, 150, 200, ],
          'learning_rate' :[0.0001,0.001,0.01, 0.1],
          'loss' : [ 'linear', 'square', 'exponential'],
          'random_state' : [10,20,30]
          }

kfold_grid_search(clf, params, x2, y2, fold = 10, kfold = 15, search = 'random')
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:  6.3min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed: 13.6min finished

Best Estimator : AdaBoostRegressor(base_estimator=None, learning_rate=0.0001, loss='linear',
                  n_estimators=100, random_state=10)


R2 metric :
0 r2 score of the train data 0.5720113699608151 and test data 0.6158808626834753
1 r2 score of the train data 0.5619763801409192 and test data 0.6355339181615801
2 r2 score of the train data 0.5707625343770122 and test data 0.606265905792721
3 r2 score of the train data 0.6058312584998451 and test data 0.3073106403664668
4 r2 score of the train data 0.5721886323759073 and test data 0.5351559426964696
5 r2 score of the train data 0.5744676443771899 and test data 0.5623683551296568
```

```
6 r2 score of the train data 0.5648681504239315 and test data 0.6552271007623669
7 r2 score of the train data 0.5655959785551095 and test data 0.7061852214705656
8 r2 score of the train data 0.580944156330872 and test data 0.4719427448074006
9 r2 score of the train data 0.5714446221740298 and test data 0.5538494689733773
10 r2 score of the train data 0.5756288720379016 and test data 0.49051296386692245
11 r2 score of the train data 0.5737776034123998 and test data 0.5774840019937164
12 r2 score of the train data 0.5684756314832162 and test data 0.6583991657352518
13 r2 score of the train data 0.5664297487945196 and test data 0.6061219537121953
14 r2 score of the train data 0.5647298013360682 and test data 0.6489853522620401

Avg r2_score of train 0.5726088256186492 and test 0.5754149065609471
Time taken - 15.350130430857341 min
```

In [ ]:

```
 ab =  AdaBoostRegressor(base_estimator=None, learning_rate=0.0001, loss='linear',
                  n_estimators=100, random_state=20)

a = cross_val_score(ab, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.5703529803771688 data
```

Gradient Boosting Regressor

In [ ]:

```
clf = GradientBoostingRegressor())

params = {'n_estimators' : [800,1000, 1500, 2000, 2500],
          'loss' : [ 'huber', 'exponential'],
          'learning_rate' : [0.01, 0.01, 0.1],
          'max_depth' : [3,4,5,7]
          }


kfold_grid_search(clf, params, x2, y2, fold = 10, kfold = 15, search = 'random') #0.6900, 0.6677
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  86 tasks      | elapsed:  6.1min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  7.2min finished

Best Estimator : GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.01, loss='huber',
```

```
                           max_depth=3, max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=2000,
                           n_iter_no_change=11, presort='deprecated',
                           random_state=10, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0, warm_start=False)


R2 metric :
0 r2 score of the train data 0.5640270026713142 and test data 0.5844962160053929
1 r2 score of the train data 0.5545558017178335 and test data 0.6263858956242915
2 r2 score of the train data 0.5683858422857833 and test data 0.5882478867422942
3 r2 score of the train data 0.6032325417831674 and test data 0.28827689355115715
4 r2 score of the train data 0.5693710703909289 and test data 0.5268253421455653
5 r2 score of the train data 0.5753977828991846 and test data 0.5509074520288433
6 r2 score of the train data 0.5659724999522501 and test data 0.6529337483797
7 r2 score of the train data 0.5692350047381489 and test data 0.7043729009907196
8 r2 score of the train data 0.5827182335013722 and test data 0.4542972897883143
9 r2 score of the train data 0.574540245256353 and test data 0.5319726953736295
10 r2 score of the train data 0.5814156987703438 and test data 0.5028486395211187
11 r2 score of the train data 0.5719466824608403 and test data 0.5601058443767275
12 r2 score of the train data 0.56953918599406 and test data 0.6726943220669401
13 r2 score of the train data 0.5654950604450828 and test data 0.6548945771237917
14 r2 score of the train data 0.5668172693254028 and test data 0.6872046861726407

Avg r2_score of train 0.572176661479471 and test 0.5724309593260751
Time taken - 11.144120848178863 min
```

In [ ]:

```
gb =  GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                           init=None, learning_rate=0.01, loss='huber',
                           max_depth=3, max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=2000,
                           n_iter_no_change=11, presort='deprecated',
                           random_state=10, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0, warm_start=False)

a = cross_val_score(gb, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.5696356812530452 data
```

LGBM Regressor

In [ ]:

```
clf =  LGBMRegressor()

params = {'min_child_samples' : [10, 20,50],
          'num_leaves' : [5,6],
          'max_depth' : [2, 3, 5],
          'n_estimators' : [1000,2000,4000,5000],
          'learning_rate' : [0.0001,0.001,0.01,0.1]
          }

kfold_grid_search(clf, params, x2, y2.ravel(), fold = 10, kfold = 15, search = 'random') #0.6920, 0.6622
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:  4.7min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  8.9min finished

Best Estimator : LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
              importance_type='split', learning_rate=0.01, max_depth=3,
              min_child_samples=50, min_child_weight=0.001, min_split_gain=0.0,
              n_estimators=4000, n_jobs=-1, num_leaves=5, objective=None,
              random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True,
              subsample=1.0, subsample_for_bin=200000, subsample_freq=0)


R2 metric :
0 r2 score of the train data 0.6539376054229651 and test data 0.5745765027557168
1 r2 score of the train data 0.6479659676917198 and test data 0.6541898772014111
2 r2 score of the train data 0.6553573129657073 and test data 0.5877012528118586
3 r2 score of the train data 0.6804781717894042 and test data 0.2926826947679162
4 r2 score of the train data 0.6577511726844416 and test data 0.5400561398326873
5 r2 score of the train data 0.6563520755321524 and test data 0.5388823520975055
6 r2 score of the train data 0.6502514789337726 and test data 0.6091735373791863
7 r2 score of the train data 0.6476857317677991 and test data 0.6896003172617456
8 r2 score of the train data 0.6622299194857179 and test data 0.47845027538518325
9 r2 score of the train data 0.6614143575698203 and test data 0.5262909902666517
10 r2 score of the train data 0.6610274429566736 and test data 0.5078239590939534
11 r2 score of the train data 0.6547838989872861 and test data 0.5718646213126627
12 r2 score of the train data 0.6524788377378758 and test data 0.6331363491047368
13 r2 score of the train data 0.6494901652393692 and test data 0.6363914114588893
14 r2 score of the train data 0.6552123606829265 and test data 0.6434023316558577
```

```
Avg r2_score of train 0.656427766629842 and test 0.5656148408257309
Time taken - 11.340572607517242 min
```

In [ ]:

```
lb =  LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
              importance_type='split', learning_rate=0.01, max_depth=3,
              min_child_samples=50, min_child_weight=0.001, min_split_gain=0.0,
              n_estimators=5000, n_jobs=-1, num_leaves=5, objective=None,
              random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True,
              subsample=1.0, subsample_for_bin=200000, subsample_freq=0)

a = cross_val_score(lb, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.5555207657145106 data
```

In [ ]:

```
estimators = [ ('rf', RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                       max_depth=5, max_features=0.95, max_leaf_nodes=None,
                       max_samples=None, min_impurity_decrease=0.001,
                       min_impurity_split=None, min_samples_leaf=2,
                       min_samples_split=8, min_weight_fraction_leaf=0.0,
                       n_estimators=70, n_jobs=None, oob_score=False,
                       random_state=None, verbose=0, warm_start=False)),

                  ('xg', XGBRFRegressor(colsample_bylevel=1,
                colsample_bynode=0.8, colsample_bytree=1, gamma=0,
                learning_rate=1, max_delta_step=0, max_depth=5,
                max_features=0.95, min_child_weight=1, min_impurity_decrease=1,
                min_samples_leaf=1, min_samples_split=5, missing=None,
                n_estimators=100, n_jobs=1, nthread=None, objective='reg:linear',
                random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
                seed=None, silent=True, subsample=0.8, verbosity=1)),


               ('gb', GradientBoostingRegressor(max_depth= 2, learning_rate= 0.1, random_state= 10,n_estimators=5000,
                                n_iter_no_change = 11))

              ]

stack = StackingRegressor(estimators= estimators,
                          final_estimator= Ridge(),
```

```
                            )

cv_score = cross_val_score(stack, x2, y2.ravel(), scoring='r2', cv=10, verbose=5, n_jobs=-1)
print('Mean Score:',cv_score.mean())
print('Standard Deviation:',cv_score.std())
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
Mean Score: 0.5762229576080763
Standard Deviation: 0.08756604747415982
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed: 12.2min remaining:    0.0s
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed: 12.2min finished
```

Models fitting

In [ ]:

```
for model, label in zip([dt, rf, xg, ab, gb, lb, stack], ['decision_tree','random_forest', 'xgboost',
                                                          'adaboost', 'gradient_boost', 'lighgbm', 'stack-ensemble']):

  model.fit(x2, y2)

  predict = model.predict(test.iloc[:, 9:])

  df = pd.DataFrame()

  df['ID'] = test.ID
  df['y']  = predict

  df.to_csv(label + '_pca+features+no-clip+no-transform.csv', index= False)
```

R^2 Score

In [ ]:

```
col = ['MODELS', 'FEATURES', 'CROSS-VALIDATION' , 'PUBLIC SCORE', 'PRIVATE SCORE']

tb = PrettyTable()

tb.add_column(col[0], ['DECISION TREE', 'RANDOM FOREST', 'XGBOOST', 'ADA-BOOST', 'GRADIENT BOOSTING', 'LIGHT GBM', 'S

tb.add_column(col[1], ['PCA(5) + BINARY + LABEL + SYNTHETIC FEATURES', 'PCA(5) + BINARY + LABEL + SYNTHETIC FEATURES
                       'PCA(5) + BINARY + LABEL + SYNTHETIC FEATURES','PCA(5) + BINARY + LABEL + SYNTHETIC FEATURES',
```

```
                      'PCA(5) + BINARY + LABEL + SYNTHETIC FEATURES'])
tb.add_column(col[2], ['0.5211', '0.5808', '0.5752', '0.5754', '0.5724', '0.5661', '0.5762'])
tb.add_column(col[3], ['0.55187', '0.55327', '0.54654', '0.54911', '0.54427', '0.54420', '0.55610'])
tb.add_column(col[4], ['0.54823', '0.54488', '0.54390', '0.54551', '0.53709', '0.52651', '0.54816'])
print(tb)
```

```
+------------------+-----------------------------------------------+-----------------+-------------+--------------
+
|       MODELS     |                    FEATURES                   | CROSS-VALIDATION | PUBLIC SCORE | PRIVATE SCORE
|
+------------------+-----------------------------------------------+-----------------+-------------+--------------
+
|   DECISION TREE  | PCA(5) + BINARY + LABEL + SYNTHETIC FEATURES |      0.5211      |   0.55187    |    0.54823
|
|   RANDOM FOREST  | PCA(5) + BINARY + LABEL + SYNTHETIC FEATURES |      0.5808      |   0.55327    |    0.54488
|
|      XGBOOST     | PCA(5) + BINARY + LABEL + SYNTHETIC FEATURES |      0.5752      |   0.54654    |    0.54390
|
|     ADA-BOOST    | PCA(5) + BINARY + LABEL + SYNTHETIC FEATURES |      0.5754      |   0.54911    |    0.54551
|
| GRADIENT BOOSTING | PCA(5) + BINARY + LABEL + SYNTHETIC FEATURES |      0.5724      |   0.54427    |    0.53709
|
|     LIGHT GBM    | PCA(5) + BINARY + LABEL + SYNTHETIC FEATURES |      0.5661      |   0.54420    |    0.52651
|
|  STCKED-ENSEMBLE  | PCA(5) + BINARY + LABEL + SYNTHETIC FEATURES |      0.5762      |   0.55610    |    0.54816
|
+------------------+-----------------------------------------------+-----------------+-------------+--------------
+
```

OBSERVATION

- This data set created like using PCA method and synthetic features.
- Using PCA method created 5 features and created 22 synthetic features like difference between twi feature and ratio of another features.
- In these models use 10 k fold cross validation and grid or random search cross validation for the best parameters.

- We can see that above table random forest well performe in the cross validation and also good perform in kaggle public score.
- Stacked ensemble got higher score in public and private score.

Models with PCA + synthetic + Binary features and y target value clip 150sec

In [9]:
```
# Import the train and test csv files
# Import the train and test csv files

train = pd.read_csv('C:/Users/Dell/Python/Python/Python AAIC File/Assignments/CASE-STUDY ML/mercedes-benz-greener-mar
test = pd.read_csv('C:/Users/Dell/Python/Python/Python AAIC File/Assignments/CASE-STUDY ML/mercedes-benz-greener-manu

train = train.drop(['Unnamed: 0'], axis = 1)
test = test.drop(['Unnamed: 0'], axis = 1)
train.head()
```

Out[9]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X46_263_119_261	X136_118_136_60	qua_encode_1	qua_encode_2	qua_encode_3	qua_encode
0	0	130.81	k	v	at	a	d	u	j	o	...	1.0	1.0	6.901179	80.729628	14.457824	8
1	6	88.53	k	t	av	e	d	y	l	o	...	0.5	1.0	28.816796	7.221336	13.855619	8
2	7	76.26	az	w	n	c	d	x	j	x	...	1.0	0.0	567.801379	9.717606	935.797419	8
3	9	80.62	az	t	n	f	d	x	l	e	...	1.0	0.0	674.842909	10.394944	930.330382	8
4	13	78.02	az	v	n	f	d	h	d	n	...	1.0	0.0	642.841100	250.973811	963.246231	8

5 rows × 404 columns

In [10]:
```
#Check the Shape of the test and train data
print('train data shape - ', train.shape)
print('test data shape  - ', test.shape)
```

```
train data shape -  (4209, 404)
```

```
test data shape  -  (4209, 403)
```

In [11]:
```
train.loc[train['y'] > 150] = 150
```

In [12]:
```
y2 = train.y
x2 = train.iloc[:, 10:]

# Split train data into train-set and test-set
x_train2, x_test2, y_train2, y_test2 = train_test_split(x2,y2, test_size = 0.33,random_state= 42)

print(x_train2.shape, y_train2.shape)
print(x_test2.shape, y_test2.shape)
```

```
(2820, 394) (2820,)
(1389, 394) (1389,)
```

Decision Tree

In [ ]:
```
clf = DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=3,
                            max_features='auto', max_leaf_nodes=None,
                            min_impurity_decrease=0.0, min_impurity_split=None,
                            min_samples_leaf=1, min_samples_split=2,
                            min_weight_fraction_leaf=0.0, presort='deprecated',
                            random_state=None, splitter='best')

params = {'max_depth' : [2,3,4,8,10,15],
          'max_features' : ['auto', 'sqrt', 'log2'],
          'random_state' : [5,10,20,30],
          }

kfold_grid_search(clf, params, x2, y2, 10, kfold = 15, search= 'random')
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  62 tasks      | elapsed:    4.3s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:    5.6s finished

Best Estimator : DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=3,
                      max_features='auto', max_leaf_nodes=None,
```

```
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=5, splitter='best')


R2 metric :
0 r2 score of the train data 0.6480538816132388 and test data 0.6693714310012511
1 r2 score of the train data 0.642465953902116 and test data 0.7702358085844839
2 r2 score of the train data 0.6533336168366211 and test data 0.6034552624278658
3 r2 score of the train data 0.650727064754295 and test data 0.626995114554626
4 r2 score of the train data 0.6481815912325416 and test data 0.6753606349895027
5 r2 score of the train data 0.6495238656607814 and test data 0.6594321202166429
6 r2 score of the train data 0.6496479097970977 and test data 0.6594673475081894
7 r2 score of the train data 0.6440153039529424 and test data 0.752330200943577
8 r2 score of the train data 0.6627475307237207 and test data 0.4674067355568491
9 r2 score of the train data 0.6516848915900538 and test data 0.6280380207465941
10 r2 score of the train data 0.6597386061652126 and test data 0.5142825155435712
11 r2 score of the train data 0.6504902324792055 and test data 0.6472108509331767
12 r2 score of the train data 0.649356510657001 and test data 0.6486385218504416
13 r2 score of the train data 0.6473059031423953 and test data 0.669704349641284
14 r2 score of the train data 0.6490679486603913 and test data 0.6551988155854951

Avg r2_score of train 0.6504227207445076 and test 0.6431418486722366
Time taken - 0.11364267269770305 min
```

In [ ]:

```
dt = DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=3,
                       max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=5, splitter='best')

a = cross_val_score(dt, x2, y2, scoring= 'r2', cv= 5)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.6355296996079519 data
```

AdaBoost-Regressor

In [ ]:

```
clf = AdaBoostRegressor(base_estimator=None, learning_rate=0.0001,
```

```
                        n_estimators=300, random_state=None, loss= 'linear')

params = {'n_estimators'  : [100, 150, 200, ],
          'learning_rate' :[0.0001,0.001,0.01, 0.1],
          'loss' : [ 'linear', 'square', 'exponential'],
          'random_state' : [10,20,30]
          }

kfold_grid_search(clf, params, x2, y2, fold = 10, kfold = 15, search = 'random')
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:  6.5min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed: 14.6min finished

Best Estimator : AdaBoostRegressor(base_estimator=None, learning_rate=0.0001, loss='exponential',
                  n_estimators=200, random_state=10)


R2 metric :
0 r2 score of the train data 0.6477109410342181 and test data 0.6667582180352138
1 r2 score of the train data 0.6411183093664439 and test data 0.769834809588402
2 r2 score of the train data 0.6519445325653355 and test data 0.6035245529110724
3 r2 score of the train data 0.6501761962040864 and test data 0.6403407813302944
4 r2 score of the train data 0.6467394871648264 and test data 0.6752756035497736
5 r2 score of the train data 0.6482150084579972 and test data 0.6593962447784378
6 r2 score of the train data 0.6483409079900484 and test data 0.6595427982210769
7 r2 score of the train data 0.6427737983809947 and test data 0.7524045226988414
8 r2 score of the train data 0.6612296144447891 and test data 0.4693381380158933
9 r2 score of the train data 0.6503698925698844 and test data 0.6281263795271057
10 r2 score of the train data 0.656219322638109 and test data 0.489240343280833
11 r2 score of the train data 0.649137793025413 and test data 0.6470255138363956
12 r2 score of the train data 0.6479060792497031 and test data 0.6619152500286267
13 r2 score of the train data 0.6458712207158706 and test data 0.66957122946993
14 r2 score of the train data 0.6476381814013082 and test data 0.6553517810957608

Avg r2_score of train 0.6490260856806018 and test 0.6431764110911772
Time taken - 18.583150271574656 min
```

In [ ]:

```
ab =  AdaBoostRegressor(base_estimator=None, learning_rate=0.0001, loss='exponential',
                  n_estimators=200, random_state=10)
```

```
a = cross_val_score(ab, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.6424381050598855 data
```

Gradient Boosting Regressor

In [ ]:

```
clf = GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                           init=None, learning_rate=0.01, loss='huber',
                           max_depth=3, max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=800,
                           n_iter_no_change=11, presort='deprecated',
                           random_state=10, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0, warm_start=False)

params = {'n_estimators' : [500,800,1000, 1500, 2000],
          'loss' : [ 'huber', 'exponential'],
          'learning_rate' : [0.01, 0.01, 0.1],
          'max_depth' : [3,4,5,7]
          }


kfold_grid_search(clf, params, x2, y2, fold = 10, kfold = 15, search = 'random')
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  74 tasks      | elapsed: 12.7min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed: 16.7min finished

Best Estimator : GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                           init=None, learning_rate=0.01, loss='huber',
                           max_depth=3, max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=1000,
                           n_iter_no_change=11, presort='deprecated',
                           random_state=10, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0, warm_start=False)
```

```
R2 metric :
0 r2 score of the train data 0.6440326836974183 and test data 0.6493814982039965
1 r2 score of the train data 0.6447564492798792 and test data 0.7714889267681285
2 r2 score of the train data 0.6556808940577739 and test data 0.5873595530545905
3 r2 score of the train data 0.6602751070405202 and test data 0.6296922521241193
4 r2 score of the train data 0.6567517205050433 and test data 0.6810093083229571
5 r2 score of the train data 0.6438252314922404 and test data 0.641228026796587
6 r2 score of the train data 0.6528491710955487 and test data 0.6555477858191263
7 r2 score of the train data 0.643880177692111 and test data 0.7560991766099738
8 r2 score of the train data 0.6759464362526924 and test data 0.4576462792867908
9 r2 score of the train data 0.6641764492261998 and test data 0.6133198511826266
10 r2 score of the train data 0.6675695377306117 and test data 0.49966612497125706
11 r2 score of the train data 0.6580900059344754 and test data 0.6316905816914543
12 r2 score of the train data 0.6539165032379398 and test data 0.6742449715204522
13 r2 score of the train data 0.6514083980245011 and test data 0.7091407057621535
14 r2 score of the train data 0.6594997815908921 and test data 0.6877275468300248

Avg r2_score of train 0.6555105697905231 and test 0.6430161725962826
Time taken - 21.6081263701121 min
```

In [ ]:

```
gb =  GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                           init=None, learning_rate=0.01, loss='huber',
                           max_depth=3, max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=1000,
                           n_iter_no_change=11, presort='deprecated',
                           random_state=10, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0, warm_start=False)

a = cross_val_score(gb, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.640877235785154 data
```

Random Forest

In [ ]:

```
clf =  RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=5, max_features=0.95, max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.001,
                      min_impurity_split=None, min_samples_leaf=2,
```

```
                      min_samples_split=8, min_weight_fraction_leaf=0.0,
                      n_estimators=70, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)

params = {'n_estimators':[40,50,60,70,100],
              'max_depth':[3,5,6,7,8],
              'min_samples_split':[2,3,4,5,6,7,8,9,10],
              'max_features': [0.80,.95, 1.0],
              'min_samples_leaf': [1, 2,3,4,5,6,7,8,9],
              'min_impurity_decrease':[1e-5,1e-4,1e-3,1e-2,1e-1,0,1,10]}

kfold_grid_search(clf, params, x2, y2, fold = 10, kfold=15, search= 'random' )
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:  1.9min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  3.6min finished

Best Estimator : RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=6, max_features=1.0, max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.1,
                      min_impurity_split=None, min_samples_leaf=6,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=70, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)


R2 metric :
0 r2 score of the train data 0.6761806107128834 and test data 0.6573744354284433
1 r2 score of the train data 0.6690398809859387 and test data 0.7684369053571742
2 r2 score of the train data 0.6804516359221022 and test data 0.5979577019930118
3 r2 score of the train data 0.6738827442338753 and test data 0.6428903435953389
4 r2 score of the train data 0.6723676756508512 and test data 0.6868064575030692
5 r2 score of the train data 0.6752737867274246 and test data 0.6627912033311976
6 r2 score of the train data 0.6748901863798457 and test data 0.6620421469359128
7 r2 score of the train data 0.670449835669485 and test data 0.7515754444268572
8 r2 score of the train data 0.6894411286450259 and test data 0.47322601756498917
9 r2 score of the train data 0.6776469544688636 and test data 0.6343698165475739
10 r2 score of the train data 0.6835132636421446 and test data 0.516588880256428
11 r2 score of the train data 0.676175147732486 and test data 0.6506305811277528
12 r2 score of the train data 0.673793912539155 and test data 0.6604770020924078
13 r2 score of the train data 0.6715570812360592 and test data 0.705112780339884
14 r2 score of the train data 0.6744616810048314 and test data 0.6628970220046523
```

```
Avg r2_score of train 0.6759417017033982 and test 0.648878449233646
Time taken - 4.554728877544403 min
```

In [ ]:

```
rf = RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=6, max_features=1.0, max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.1,
                      min_impurity_split=None, min_samples_leaf=6,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=700, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)


a = cross_val_score(rf, x2, y_trans, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.6457039283917678 data
```

LGBM Regressor

In [ ]:

```
clf =  LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
               importance_type='split', learning_rate=0.01, max_depth=5,
               min_child_samples=50, min_child_weight=0.001, min_split_gain=0.0,
               n_estimators=1000, n_jobs=-1, num_leaves=5, objective=None,
               random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True,
               subsample=1.0, subsample_for_bin=200000, subsample_freq=0)

params = {'min_child_samples' : [10, 20,50],
          'num_leaves' : [5,6],
          'max_depth' : [2, 3, 5],
          'n_estimators' : [1000,2000,4000,5000],
          'learning_rate' : [0.0001,0.001,0.01,0.1]
          }

kfold_grid_search(clf, params, x2, y2.ravel(), fold = 10, kfold = 15, search = 'random')
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:  3.9min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  8.6min finished
```

```
Best Estimator : LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
              importance_type='split', learning_rate=0.001, max_depth=3,
              min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
              n_estimators=5000, n_jobs=-1, num_leaves=6, objective=None,
              random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True,
              subsample=1.0, subsample_for_bin=200000, subsample_freq=0)


R2 metric :
0 r2 score of the train data 0.6646104803919637 and test data 0.6630022586838633
1 r2 score of the train data 0.6583630541223624 and test data 0.7675817081799015
2 r2 score of the train data 0.6690928334118285 and test data 0.6058607904596616
3 r2 score of the train data 0.6649248016084909 and test data 0.6436142891192629
4 r2 score of the train data 0.6624097369466116 and test data 0.6936298300139243
5 r2 score of the train data 0.6646374197084737 and test data 0.665528759256929
6 r2 score of the train data 0.6650009587077517 and test data 0.6631473971131017
7 r2 score of the train data 0.6604176474827126 and test data 0.7501455280855138
8 r2 score of the train data 0.6798367687788365 and test data 0.47817342733889345
9 r2 score of the train data 0.666810659145471 and test data 0.6324631406650509
10 r2 score of the train data 0.6751084633746813 and test data 0.5094524762619161
11 r2 score of the train data 0.6656215129264706 and test data 0.6408139966621724
12 r2 score of the train data 0.6642275397074902 and test data 0.6602457548831437
13 r2 score of the train data 0.6622259856150257 and test data 0.7103376271877867
14 r2 score of the train data 0.6632183999773167 and test data 0.6660837115764423

Avg r2_score of train 0.6657670841270324 and test 0.650005379699171
Time taken - 13.400730057557423 min
```

In [ ]:

```
lb =  LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
              importance_type='split', learning_rate=0.001, max_depth=3,
              min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
              n_estimators=5000, n_jobs=-1, num_leaves=6, objective=None,
              random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True,
              subsample=1.0, subsample_for_bin=200000, subsample_freq=0)

a = cross_val_score(lb, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.6468964578401072 data
```

XGBoost-Regressor

In [ ]:
```
clf = XGBRFRegressor(colsample_bylevel=1,colsample_bynode=0.8, colsample_bytree=1, gamma=0,
                learning_rate=1, max_delta_step=0, max_depth=5,
                max_features=0.95, min_child_weight=1, min_impurity_decrease=1,
                min_samples_leaf=1, min_samples_split=5, missing=None,
                n_estimators=100, n_jobs=1, nthread=None, objective='reg:linear',
                random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
                seed=None, silent=True, subsample=0.8, verbosity=1)

xparams = {'learning_rate':[0.1,0.5,0.8,1],
              'n_estimators':[70,80,100],
              'max_depth':[2,3,4],
              'colsample_bytree':[0.1,0.5,0.7,0.9,1],
              'subsample':[0.2,0.3,0.5,1],
              'gamma':[0.0001,0.001,0,0.1,0.01,0.5,1],
              'reg_alpha':[0.00001,0.0001,0.001,0.01,0.1]}


kfold_grid_search(clf, params, x2, y2, fold = 10, kfold=15, search= False )
```

```
R2 metric :
0 r2 score of the train data 0.6066547132057518 and test data 0.63525224317498
1 r2 score of the train data 0.6034804178965002 and test data 0.6963600576690997
2 r2 score of the train data 0.6105771133880324 and test data 0.601203463972948
3 r2 score of the train data 0.6128226383721311 and test data 0.5564458465086671
4 r2 score of the train data 0.6111184302393099 and test data 0.5830068184166133
5 r2 score of the train data 0.6082304797363157 and test data 0.6281456387239466
6 r2 score of the train data 0.6089669154327122 and test data 0.6561857533878476
7 r2 score of the train data 0.6035336362444628 and test data 0.7099815395322746
8 r2 score of the train data 0.6238022287495313 and test data 0.4687577983273177
9 r2 score of the train data 0.6125290949699321 and test data 0.5628813136358713
10 r2 score of the train data 0.6200341125828022 and test data 0.48810031559351164
11 r2 score of the train data 0.6088743143552269 and test data 0.6168210561799822
12 r2 score of the train data 0.6083074897370442 and test data 0.658639519541895
13 r2 score of the train data 0.6073444564835199 and test data 0.6416319396499979
14 r2 score of the train data 0.607225555256518 and test data 0.6514214996312307

Avg r2_score of train 0.6102334397766528 and test 0.6103223202630789
Time taken - 0.7519578218460083 min
```

In [ ]:
```
 xg = XGBRFRegressor(colsample_bylevel=1,
                colsample_bynode=0.8, colsample_bytree=1, gamma=0,
                learning_rate=1, max_delta_step=0, max_depth=5,
```

```
                max_features=0.95, min_child_weight=1, min_impurity_decrease=1,
                min_samples_leaf=1, min_samples_split=5, missing=None,
                n_estimators=100, n_jobs=1, nthread=None, objective='reg:linear',
                random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
                seed=None, silent=True, subsample=0.8, verbosity=1)

a = cross_val_score(xg, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.6084653791278974 data
```

Stack ensemble

In [14]:

```
estimators = [ ('rf', RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                       max_depth=5, max_features=0.95, max_leaf_nodes=None,
                       max_samples=None, min_impurity_decrease=0.001,
                       min_impurity_split=None, min_samples_leaf=2,
                       min_samples_split=8, min_weight_fraction_leaf=0.0,
                       n_estimators=70, n_jobs=None, oob_score=False,
                       random_state=None, verbose=0, warm_start=False)),

                  ('xg', XGBRFRegressor(colsample_bylevel=1,
                colsample_bynode=0.8, colsample_bytree=1, gamma=0,
                learning_rate=1, max_delta_step=0, max_depth=5,
                max_features=0.95, min_child_weight=1, min_impurity_decrease=1,
                min_samples_leaf=1, min_samples_split=5, missing=None,
                n_estimators=100, n_jobs=1, nthread=None, objective='reg:linear',
                random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
                seed=None, silent=True, subsample=0.8, verbosity=0)),


               ('lg', LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
               importance_type='split', learning_rate=0.001, max_depth=3,
               min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
               n_estimators=5000, n_jobs=-1, num_leaves=6, objective=None,
               random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True,
               subsample=1.0, subsample_for_bin=200000, subsample_freq=0))

              ]
```

```
stack = StackingRegressor(estimators= estimators,
                          final_estimator= Ridge(),
                          )
```

In [ ]:

```
cv_score = cross_val_score(stack, x2, y2.ravel(), scoring='r2', cv= 5, verbose=5, n_jobs=-1)
print('Mean Score:',cv_score.mean())
print('Standard Deviation:',cv_score.std())   #0.6404, 0.64200,
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
Mean Score: 0.6412507934357022
Standard Deviation: 0.030613234929429335
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed: 22.7min finished
```

In [17]:

```
stack = stack.fit(x2.values, y2)
```

In [18]:

```
joblib.dump(stack, 'stack_model.joblib.dat')

print('Model saved in disk!')
```

```
Model saved in disk!
```

Models fitting

In [ ]:

```
for model, label in zip([dt, rf, xg, ab, gb, lb, stack], ['decision_tree','random_forest', 'xgboost',
                                    'adaboost', 'gradient_boost', 'lighgbm', 'stack_ensemble']):

  model.fit(x2, y2)

  predict = model.predict(test.iloc[:, 9:])

  df = pd.DataFrame()

  df['ID'] = test.ID
  df['y']  = predict

  df.to_csv(label + '_pca+features+clip+no-transform.csv', index= False)
```

R^2 Score

In [56]:

```
col = ['MODELS', 'FEATURES', 'CROSS-VALIDATION' , 'PUBLIC SCORE', 'PRIVATE SCORE']

tb = PrettyTable()

tb.add_column(col[0], ['DECISION TREE', 'RANDOM FOREST', 'XGBOOST', 'ADA-BOOST', 'GRADIENT BOOSTING', 'LIGHT GBM', 'S

tb.add_column(col[1], ['PCA(5) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES','PCA(5) + BINARY + LABEL + Y 150 CL
                       'PCA(5) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES','PCA(5) + BINARY + LABEL + Y 150 CL
                       'PCA(5) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES','PCA(5) + BINARY + LABEL + Y 150 CL
                       'PCA(5) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES',])

tb.add_column(col[2], ['0.6451', '0.6488', '0.6102', '0.6431', '0.6430', '0.6500', '0.6412'])

tb.add_column(col[3], ['0.53938', '0.55102', '0.54616', '0.53998', '0.54294', '0.55541', '0.55514'])

tb.add_column(col[4], ['0.53968', '0.54857', '0.53822', '0.53963', '0.53637', '0.54895', '0.54951'])

print(tb)
```

```
+-------------------+------------------------------------------------------------+------------------+-------------+---
------------+
|       MODELS      |                          FEATURES                          | CROSS-VALIDATION | PUBLIC SCORE | PR
IVATE SCORE |
+-------------------+------------------------------------------------------------+------------------+-------------+---
------------+
|   DECISION TREE   | PCA(5) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES |      0.6451      |   0.53938    |
0.53968    |
|   RANDOM FOREST   | PCA(5) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES |      0.6488      |   0.55102    |
0.54857    |
|      XGBOOST      | PCA(5) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES |      0.6102      |   0.54616    |
0.53822    |
|     ADA-BOOST     | PCA(5) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES |      0.6431      |   0.53998    |
0.53963    |
| GRADIENT BOOSTING | PCA(5) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES |      0.6430      |   0.54294    |
0.53637    |
|     LIGHT GBM     | PCA(5) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES |      0.6500      |   0.55541    |
0.54895    |
|  STCKED-ENSEMBLE  | PCA(5) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES |      0.6412      |   0.55514    |
0.54951    |
```

```
+------------------+-------------------------------------------------------------+-----------------+-------------+---
------------+
```

OBSERVATION

- This dataset contain 5 PCA feature, binary features, label encoded features and synthetic featues.
- This dataset with some experiment found that when we clip our y target with threshold 150secs then we got good score.
- We got good score in this data higher than another datasets, we see that in above table stacked ensemble get best score in public 0.55514 and private 0.54951 score.
- All models well perform in cross validation sets with k fold method.

Model with important feature of models (synethic features + 100 PCA + Binary)

In [11]:
```
# Import the train and test csv files

train = pd.read_csv(r'/content/drive/MyDrive/top_feature_train.csv')
test = pd.read_csv(r'/content/drive/MyDrive/top_feature_test.csv')

train = train.drop(['Unnamed: 0'], axis = 1)
test = test.drop(['Unnamed: 0'], axis = 1)
train.head()
```

Out[11]:

	ID	X0	X1	X2	X3	X5	X6	X8	X10	X12	X13	X14	X19	X20	X22	X23	X27	X28	X29	X31	X32	X35	X37	X38	X41	X43	X44	X45	X
0	0	k	v	at	a	u	j	o	0	0	1	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	
1	6	k	t	av	e	y	l	o	0	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0	0	0	0	
2	7	az	w	n	c	x	j	x	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	0	0	1	0	0	
3	9	az	t	n	f	x	l	e	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	0	0	1	0	0	
4	13	az	v	n	f	h	d	n	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	0	0	1	0	0	

5 rows × 350 columns

In [12]:
```
#Check the Shape of the test and train data
print('train data shape - ', train.shape)
print('test data shape  - ', test.shape)
```

```
train data shape -  (4209, 350)
test data shape  -  (4209, 349)
```

In [13]:
```
train.loc[train['y'] > 150] = 150

y2 = train.y
move = [        'ID' ,  'X0' ,  'X1' ,  'X2'    ,'X3' ,'X5' ,   'X6' ,  'X8', 'y']
x2 = train.drop(move, axis= 1)
```

In [14]:
```
# Split train data into train-set and test-set
x_train2, x_test2, y_train2, y_test2 = train_test_split(x2,y2, test_size = 0.33,random_state= 42)

print(x_train2.shape, y_train2.shape)
print(x_test2.shape, y_test2.shape)
```

```
(2820, 341) (2820,)
(1389, 341) (1389,)
```

Decision Tree

In [15]:
```
clf = DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=3,
                            max_features='auto', max_leaf_nodes=None,
                            min_impurity_decrease=0.0, min_impurity_split=None,
                            min_samples_leaf=1, min_samples_split=2,
                            min_weight_fraction_leaf=0.0, presort='deprecated',
                            random_state=None, splitter='best')

params = {'max_depth' : [2,3,4,8,10,15],
          'max_features' : ['auto', 'sqrt', 'log2'],
          'random_state' : [5,10,20,30],
```

```
        }

kfold_grid_search(clf, params, x2, y2, 10, kfold = 15, search= 'grid')
```

```
Fitting 10 folds for each of 72 candidates, totalling 720 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:    4.0s
[Parallel(n_jobs=-1)]: Done 466 tasks      | elapsed:   33.9s
[Parallel(n_jobs=-1)]: Done 717 out of 720 | elapsed:  1.3min remaining:    0.3s
[Parallel(n_jobs=-1)]: Done 720 out of 720 | elapsed:  1.3min finished

Best Estimator : DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=3,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=20, splitter='best')


R2 metric :
0 r2 score of the train data 0.5727993448261308 and test data 0.5517333068615851
1 r2 score of the train data 0.5653722857340067 and test data 0.6649931070125705
2 r2 score of the train data 0.5734600812347554 and test data 0.6037041847200402
3 r2 score of the train data 0.6079480564620057 and test data 0.295841387440966
4 r2 score of the train data 0.5780829128488911 and test data 0.5094437738972812
5 r2 score of the train data 0.5770455134950616 and test data 0.5546179692984541
6 r2 score of the train data 0.5714621388760321 and test data 0.63931768707822
7 r2 score of the train data 0.5679556024559522 and test data 0.7047013902313545
8 r2 score of the train data 0.58480623748573 and test data 0.4574489093196963
9 r2 score of the train data 0.57678165997314 and test data 0.5543289851177138
10 r2 score of the train data 0.5838871469899598 and test data 0.40937726622385506
11 r2 score of the train data 0.5742070049919803 and test data 0.5752652423899043
12 r2 score of the train data 0.5703728938595942 and test data 0.6575806943704936
13 r2 score of the train data 0.5716939319770864 and test data 0.598721533902179
14 r2 score of the train data 0.5701840060357541 and test data 0.6472752206392525

Avg r2_score of train 0.5764039211497387 and test 0.5616233772335711
Time taken - 1.3359219153722128 min
```

In [22]:

```
dt =  DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=3,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
```

```
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=20, splitter='best')

a = cross_val_score(dt, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.5645815440213546 data
```

Random Forest

In [16]:

```
clf = RandomForestRegressor()

params = {'n_estimators':[40,50,60,70,100],
              'max_depth':[3,5,6,7,8],
              'min_samples_split':[2,3,4,5,6,7,8,9,10],
              'max_features': [0.80,.95, 1.0],
              'min_samples_leaf': [1, 2,3,4,5,6,7,8,9],
              'min_impurity_decrease':[1e-5,1e-4,1e-3,1e-2,1e-1,0,1,10]}

kfold_grid_search(clf, params, x2, y2, fold = 10, kfold=15, search= 'random')
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:  5.3min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed: 13.0min finished

Best Estimator : RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=3, max_features=0.95, max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=1e-05,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=3, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)


R2 metric :
0 r2 score of the train data 0.577087291813353 and test data 0.6132711188897396
1 r2 score of the train data 0.5703996644280093 and test data 0.6609684185922715
2 r2 score of the train data 0.578452544680822 and test data 0.6028701420087133
3 r2 score of the train data 0.6125421985716001 and test data 0.30357217989243657
4 r2 score of the train data 0.582456005055181 and test data 0.5311574076974697
5 r2 score of the train data 0.581347147499635 and test data 0.558037383796935
```

```
6 r2 score of the train data 0.5755225465600952 and test data 0.653998063252534
7 r2 score of the train data 0.571482877918104 and test data 0.704416810834859
8 r2 score of the train data 0.5880711478936098 and test data 0.4654762165866627
9 r2 score of the train data 0.581164795256482 and test data 0.554894060385233
10 r2 score of the train data 0.5867241938482605 and test data 0.46223717679878484
11 r2 score of the train data 0.5771074180696558 and test data 0.5786484609591314
12 r2 score of the train data 0.5749890540584395 and test data 0.6536878481220965
13 r2 score of the train data 0.5753089240842089 and test data 0.6261519508819551
14 r2 score of the train data 0.5744542619612625 and test data 0.645344399612696

Avg r2_score of train 0.5804740047799146 and test 0.5743154425541012
Time taken - 15.681973775227865 min
```

In [23]:

```
rf =  RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=3, max_features=0.95, max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=1e-05,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=3, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)

a = cross_val_score(rf, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.5722703062634319 data
```

XGBoost-Regressor

In [17]:

```
clf = XGBRFRegressor(silent=True)

xparams = {'learning_rate':[0.1,0.5,0.8,1],
             'n_estimators':[70,80,100],
             'max_depth':[2,3,4],
             'colsample_bytree':[0.1,0.5,0.7,0.9,1],
             'subsample':[0.2,0.3,0.5,1],
             'gamma':[0.0001,0.001,0,0.1,0.01,0.5,1],
             'reg_alpha':[0.00001,0.0001,0.001,0.01,0.1]}


kfold_grid_search(clf, params, x2, y2, fold = 10, kfold=15, search= 'random')
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  3.3min finished

Best Estimator : XGBRFRegressor(base_score=0.5, colsample_bylevel=1, colsample_bynode=0.8,
               colsample_bytree=1, gamma=0, learning_rate=1, max_delta_step=0,
               max_depth=7, max_features=0.8, min_child_weight=1,
               min_impurity_decrease=0.001, min_samples_leaf=4,
               min_samples_split=2, missing=None, n_estimators=40, n_jobs=1,
               nthread=None, objective='reg:linear', random_state=0,
               reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
               silent=True, subsample=0.8, verbosity=1)


R2 metric :
0 r2 score of the train data 0.5607519129595249 and test data 0.612350904071567
1 r2 score of the train data 0.5577277386771264 and test data 0.6554340512207613
2 r2 score of the train data 0.5617422277259452 and test data 0.6032007707051259
3 r2 score of the train data 0.5954512272952628 and test data 0.306385454262185
4 r2 score of the train data 0.567338555840642 and test data 0.5264216134296651
5 r2 score of the train data 0.5652550760126759 and test data 0.554397766212482
6 r2 score of the train data 0.5588987037502464 and test data 0.6583819998690197
7 r2 score of the train data 0.556428402163345 and test data 0.703099768991829
8 r2 score of the train data 0.5716366105241955 and test data 0.47008749323113264
9 r2 score of the train data 0.5650894144901676 and test data 0.5520598424562043
10 r2 score of the train data 0.5698685506395984 and test data 0.486998617080977
11 r2 score of the train data 0.5633495452041231 and test data 0.5760128251942953
12 r2 score of the train data 0.558672400266895 and test data 0.6593011513004952
13 r2 score of the train data 0.5600793827235035 and test data 0.63677973271229
14 r2 score of the train data 0.5586598565803855 and test data 0.6518594742101987

Avg r2_score of train 0.5647299736569091 and test 0.5768514309965485
Time taken - 3.8507372379302978 min
```

In [24]:

```
xg =  XGBRFRegressor(base_score=0.5, colsample_bylevel=1, colsample_bynode=0.8,
               colsample_bytree=1, gamma=0, learning_rate=1, max_delta_step=0,
               max_depth=7, max_features=0.8, min_child_weight=1,
               min_impurity_decrease=0.001, min_samples_leaf=4,
               min_samples_split=2, missing=None, n_estimators=40, n_jobs=1,
               nthread=None, objective='reg:linear', random_state=0,
               reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
               silent=True, subsample=0.8, verbosity=1)
```

```
a = cross_val_score(xg, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.5730571735111031 data
```

AdaBoost-Regressor

In [18]:

```
clf = AdaBoostRegressor(base_estimator=None, learning_rate=0.0001,
                        n_estimators=300, random_state=None, loss= 'linear')

params = {'n_estimators'  : [100, 150, 200, ],
          'learning_rate' :[0.0001,0.001,0.01, 0.1],
          'loss' : [ 'linear', 'square', 'exponential'],
          'random_state' : [10,20,30]
          }

kfold_grid_search(clf, params, x2, y2, fold = 10, kfold = 15, search = 'random')
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed: 14.2min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed: 32.0min finished

Best Estimator : AdaBoostRegressor(base_estimator=None, learning_rate=0.0001, loss='square',
                  n_estimators=150, random_state=30)


R2 metric :
0 r2 score of the train data 0.5646822713230479 and test data 0.6066783130115726
1 r2 score of the train data 0.5577141547036 and test data 0.6649903462149656
2 r2 score of the train data 0.5651393370662766 and test data 0.6033862254239895
3 r2 score of the train data 0.5989253378984944 and test data 0.3064867419902134
4 r2 score of the train data 0.5703661713907457 and test data 0.532300026745163
5 r2 score of the train data 0.5687084265678528 and test data 0.5544782697611486
6 r2 score of the train data 0.5626966400650278 and test data 0.6543380452537592
7 r2 score of the train data 0.5566910900648376 and test data 0.7050220948487183
8 r2 score of the train data 0.5756621801071371 and test data 0.46433697036193844
9 r2 score of the train data 0.5682103360047548 and test data 0.5530743664423573
10 r2 score of the train data 0.5787962790410337 and test data 0.40115919393392063
11 r2 score of the train data 0.5669787210234464 and test data 0.5789391829398253
```

```
12 r2 score of the train data 0.5617779914575695 and test data 0.6614607088330683
13 r2 score of the train data 0.5632549126709887 and test data 0.6362939377829919
14 r2 score of the train data 0.563774845950836 and test data 0.6504187322795321

Avg r2_score of train 0.5682252463557101 and test 0.5715575437215442
Time taken - 39.760059813658394 min
```

In [25]:

```
ab =   AdaBoostRegressor(base_estimator=None, learning_rate=0.0001, loss='square',
                  n_estimators=150, random_state=30)

a = cross_val_score(ab, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.5743669401083786 data
```

Gradient Boosting Regressor

In [19]:

```
clf = GradientBoostingRegressor(max_depth= 2, learning_rate= 0.1, random_state= 10,n_estimators=5000,
                                n_iter_no_change = 11)

params = {'n_estimators' : [500,800,1000, 1500, 2000],
          'loss' : [ 'huber', 'exponential'],
          'learning_rate' : [0.01, 0.01, 0.1],
          'max_depth' : [3,4,5,7]
          }


kfold_grid_search(clf, params, x2, y2, fold = 10, kfold = 15, search = 'random')
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  62 tasks      | elapsed: 26.1min
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed: 48.8min finished

Best Estimator : GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.01, loss='huber',
                          max_depth=3, max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=1500,
                          n_iter_no_change=11, presort='deprecated',
```

```
                           random_state=10, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0, warm_start=False)


R2 metric :
0 r2 score of the train data 0.572231628764966 and test data 0.6037034985933161
1 r2 score of the train data 0.5621641874751212 and test data 0.6414798160235033
2 r2 score of the train data 0.5867592358384793 and test data 0.5801527896688994
3 r2 score of the train data 0.6211634297112336 and test data 0.2921090496165797
4 r2 score of the train data 0.5748619162004004 and test data 0.5269244313269791
5 r2 score of the train data 0.5872705137166562 and test data 0.549299214543042
6 r2 score of the train data 0.5855561004342713 and test data 0.6578667585099021
7 r2 score of the train data 0.570752093827166 and test data 0.702863121942162
8 r2 score of the train data 0.5842016164058297 and test data 0.4572405045212359
9 r2 score of the train data 0.5798527918648275 and test data 0.5332054489958042
10 r2 score of the train data 0.5820903105918269 and test data 0.4970823405618191
11 r2 score of the train data 0.5844905309440279 and test data 0.5477665372635308
12 r2 score of the train data 0.5732064802854702 and test data 0.6722729775204286
13 r2 score of the train data 0.5794606697712166 and test data 0.6724396565967647
14 r2 score of the train data 0.5778799612181722 and test data 0.6866115001225563

Avg r2_score of train 0.5814627644699777 and test 0.5747345097204349
Time taken - 64.8858946720759 min
```

In [26]:

```
gb = GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                           init=None, learning_rate=0.01, loss='huber',
                           max_depth=3, max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=1500,
                           n_iter_no_change=11, presort='deprecated',
                           random_state=10, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0, warm_start=False)

a = cross_val_score(gb, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.5731033817060677 data
```

LGBM Regressor

In [21]:

```
clf =   LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
              importance_type='split', learning_rate=0.001, max_depth=3,
              min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
              n_estimators=5000, n_jobs=-1, num_leaves=6, objective=None,
              random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True,
              subsample=1.0, subsample_for_bin=200000, subsample_freq=0)

params = {'min_child_samples' : [10, 20,50],
          'num_leaves' : [5,6],
          'max_depth' : [2, 3, 5],
          'n_estimators' : [1000,2000,4000,5000],
          'learning_rate' : [0.0001,0.001,0.01,0.1]
          }

kfold_grid_search(clf, params, x2, y2.ravel(), fold = 10, kfold = 15, search = False)
```

```
R2 metric :
0 r2 score of the train data 0.5997441404083625 and test data 0.6096233813618482
1 r2 score of the train data 0.5958881929472761 and test data 0.6445588724260033
2 r2 score of the train data 0.602526807697845 and test data 0.6007657087156126
3 r2 score of the train data 0.6327607569536936 and test data 0.3070436943331817
4 r2 score of the train data 0.6063989312693985 and test data 0.5403439972577462
5 r2 score of the train data 0.6060046445533136 and test data 0.5558067093017696
6 r2 score of the train data 0.5993389114393196 and test data 0.6537402142783227
7 r2 score of the train data 0.5958897819905952 and test data 0.6964613240982211
8 r2 score of the train data 0.6112823739876911 and test data 0.48196247922911584
9 r2 score of the train data 0.6066215294634596 and test data 0.5536277630993041
10 r2 score of the train data 0.607835933029663 and test data 0.5027366167589261
11 r2 score of the train data 0.6068813272907176 and test data 0.5586564302402194
12 r2 score of the train data 0.5974123440555079 and test data 0.6520383073567042
13 r2 score of the train data 0.5979811420951997 and test data 0.6558092545600763
14 r2 score of the train data 0.5989684485447244 and test data 0.6523321985442174

Avg r2_score of train 0.6043690177151179 and test 0.577700463437418
Time taken - 9.296628991762796 min
```

In [28]:

```
lg = LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
              importance_type='split', learning_rate=0.001, max_depth=3,
              min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
              n_estimators=5000, n_jobs=-1, num_leaves=6, objective=None,
              random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True,
              subsample=1.0, subsample_for_bin=200000, subsample_freq=0)
```

```
a = cross_val_score(lg, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.5736623038048128 data
```

In [29]:

```
estimators = [ ('rf', RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                       max_depth=6, max_features=1.0, max_leaf_nodes=None,
                       max_samples=None, min_impurity_decrease=0.0001,
                       min_impurity_split=None, min_samples_leaf=4,
                       min_samples_split=10, min_weight_fraction_leaf=0.0,
                       n_estimators=100, n_jobs=None, oob_score=False,
                       random_state=None, verbose=0, warm_start=False)),

               ('xgb', XGBRFRegressor(base_score=0.5, colsample_bylevel=1, colsample_bynode=0.8,
                colsample_bytree=1, gamma=0, learning_rate=1, max_delta_step=0,
                max_depth=5, max_features=0.95, min_child_weight=1,
                min_impurity_decrease=0.0001, min_samples_leaf=2,
                min_samples_split=4, missing=None, n_estimators=50, n_jobs=1,
                nthread=None, objective='reg:linear', random_state=0,
                reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
                silent=True, subsample=0.8, verbosity=1)),

               ('lb', LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
               importance_type='split', learning_rate=0.001, max_depth=3,
               min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
               n_estimators=5000, n_jobs=-1, num_leaves=6, objective=None,
               random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True,
               subsample=1.0, subsample_for_bin=200000, subsample_freq=0))
               ]
```

In [30]:

```
stack = StackingRegressor(estimators= estimators,
                          final_estimator= Ridge(),
                          )


cv_score = cross_val_score(stack, x2, y2.ravel(), scoring='r2', cv= 10, verbose=5, n_jobs=-1)
print('Mean Score:',cv_score.mean())
print('Standard Deviation:',cv_score.std())
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
Mean Score: 0.575714581019241
Standard Deviation: 0.08823219159340108
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed: 39.0min remaining:    0.0s
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed: 39.0min finished
```

Models fitting

In [36]:

```
for model, label in zip([dt, rf, xg, ab, gb, lg, stack], ['decision_tree','random_forest', 'xgboost',
                                        'adaboost', 'gradient_boost', 'lighgbm', 'stack_ensemble']):

  model.fit(x2, y2)

  predict = model.predict(test.iloc[:, 8:])

  df = pd.DataFrame()

  df['ID'] = test.ID
  df['y']  = predict

  df.to_csv(label + '_top_features(pca-100)_with_clip.csv', index= False)

  print('csv file compeleted ' + label)
```

```
csv file compeleted decision_tree
csv file compeleted random_forest
csv file compeleted xgboost
csv file compeleted adaboost
csv file compeleted gradient_boost
csv file compeleted lighgbm
csv file compeleted stack_ensemble
```

In [57]:

```
col = ['MODELS', 'FEATURES', 'CROSS-VALIDATION' , 'PUBLIC SCORE', 'PRIVATE SCORE']

tb = PrettyTable()

tb.add_column(col[0], ['DECISION TREE', 'RANDOM FOREST', 'XGBOOST', 'ADA-BOOST', 'GRADIENT BOOSTING', 'LIGHT GBM', 'S

tb.add_column(col[1], ['PCA(100) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES','PCA(100) + BINARY + LABEL + Y 15
```

```
                      'PCA(100) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES','PCA(100) + BINARY + LABEL + Y 15
                      'PCA(100) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES','PCA(100) + BINARY + LABEL + Y 15
                      'PCA(100) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES',])

tb.add_column(col[2], ['0.5616', '0.5743', '0.5768', '0.5715', '0.5847', '0.5775', '0.5757'])

tb.add_column(col[3], ['0.54541', '0.54556', '0.54345', '0.54539', '0.54321', '0.55793', '0.55442'])

tb.add_column(col[4], ['0.52796', '0.53966', '0.53673', '0.53996', '0.53868', '0.54630', '0.54515'])

print(tb)
```

```
+-------------------+-------------------------------------------------------------+-----------------+-------------+-
-------------+
|       MODELS      |                          FEATURES                           | CROSS-VALIDATION | PUBLIC SCORE |
PRIVATE SCORE |
+-------------------+-------------------------------------------------------------+-----------------+-------------+-
-------------+
|   DECISION TREE   | PCA(100) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES |      0.5616      |   0.54541    |
   0.52796    |
|   RANDOM FOREST   | PCA(100) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES |      0.5743      |   0.54556    |
   0.53966    |
|      XGBOOST      | PCA(100) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES |      0.5768      |   0.54345    |
   0.53673    |
|     ADA-BOOST     | PCA(100) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES |      0.5715      |   0.54539    |
   0.53996    |
| GRADIENT BOOSTING | PCA(100) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES |      0.5847      |   0.54321    |
   0.53868    |
|     LIGHT GBM     | PCA(100) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES |      0.5775      |   0.55793    |
   0.54630    |
|  STCKED-ENSEMBLE  | PCA(100) + BINARY + LABEL + Y 150 CLIP +SYNTHETIC FEATURES |      0.5757      |   0.55442    |
   0.54515    |
+-------------------+-------------------------------------------------------------+-----------------+-------------+-
-------------+
```

OBSERVATION

- This dataset contain 100 PCA features from binary and label encode features. With target y clip at 150secs
- In this dataset light gradient boosting perform well than other. LGB got 0.55793 in public and 0.54630 private score.

Models with selected top features using selectkbest method

In [37]:
```
# Import the train and test csv files

train = pd.read_csv(r'/content/drive/MyDrive/Datasets/selectK_train_feature.csv')
test = pd.read_csv(r'/content/drive/MyDrive/Datasets/selectK_test_feature.csv')

train = train.drop(['Unnamed: 0'], axis = 1)
test = test.drop(['Unnamed: 0'], axis = 1)
train.head()
```

Out[37]:

	X10	X12	X13	X14	X19	X20	X22	X23	X27	X28	X29	X31	X35	X37	X43	X44	X45	X46	X47	X48	X50	X51	X52	X54	X56	X57	X6
0	0	0	1	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	
2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	1	0	0	0	1	0	1	0	0	
3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0	1	0	0	
4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	1	0	0	0	1	0	1	0	0	

5 rows × 252 columns

In [38]:
```
#Check the Shape of the test and train data
print('train data shape - ', train.shape)
print('test data shape  - ', test.shape)
```

```
train data shape -  (4209, 252)
test data shape  -  (4209, 251)
```

In [39]:
```
# declare the train set and label
train.loc[train['y'] > 150] = 150
y2 = train.y
x2 = train.drop(['y', 'ID'], axis= 1)
```

In [40]:
```
# Split train data into train-set and test-set
x_train2, x_test2, y_train2, y_test2 = train_test_split(x2,y2, test_size = 0.33,random_state= 42)

print(x_train2.shape, y_train2.shape)
print(x_test2.shape, y_test2.shape)
```

```
(2820, 250) (2820,)
(1389, 250) (1389,)
```

Decision Tree

In [ ]:
```
clf = DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=8,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=10, splitter='best')

params = {'max_depth' : [2,3,4,8,10,15],
          'max_features' : ['auto', 'sqrt', 'log2'],
          'random_state' : [5,10,20,30],
          }

kfold_grid_search(clf, params, x2, y2, 10, kfold = 15, search= False)
```

```
R2 metric :
0 r2 score of the train data 0.7003943217596944 and test data 0.5912088567314289
1 r2 score of the train data 0.684772877128098 and test data 0.7461049970562634
2 r2 score of the train data 0.7024668433891952 and test data 0.5695775837295918
3 r2 score of the train data 0.6955558898354829 and test data 0.6247256843562159
4 r2 score of the train data 0.6943306686873321 and test data 0.6938806609949865
5 r2 score of the train data 0.6917795390935819 and test data 0.5715765496011846
6 r2 score of the train data 0.6950587650032459 and test data 0.6254903902405267
7 r2 score of the train data 0.6899252956182615 and test data 0.7482496095026618
8 r2 score of the train data 0.7066464250517441 and test data 0.4451958898852242
9 r2 score of the train data 0.699592124849862 and test data 0.5409245741315845
10 r2 score of the train data 0.7090500610772879 and test data 0.3688190526192887
11 r2 score of the train data 0.6980781796780453 and test data 0.632043078306469
12 r2 score of the train data 0.6959999890239887 and test data 0.6536206550723016
13 r2 score of the train data 0.6938200362681952 and test data 0.6973450444266769
14 r2 score of the train data 0.6936533776743699 and test data 0.5191145080931587
```

```
Avg r2_score of train 0.6967416262758924 and test 0.6018584756498376
Time taken - 0.0696996808052063 min
```

In [41]:

```
dt = DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=3,
                           max_features='auto',
                           min_impurity_decrease=0.0,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, presort='deprecated',
                           splitter='best')


a = cross_val_score(dt, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.6350922926910991 data
```

Random Forest

In [ ]:

```
clf = RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=3, max_features=0.8, max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=1e-05,
                      min_impurity_split=None, min_samples_leaf=5,
                      min_samples_split=9, min_weight_fraction_leaf=0.0,
                      n_estimators=40, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)

params = {'n_estimators':[40,50,60,70,100],
             'max_depth':[3,5,6,7,8],
             'min_samples_split':[2,3,4,5,6,7,8,9,10],
             'max_features': [0.80,.95, 1.0],
             'min_samples_leaf': [1, 2,3,4,5,6,7,8,9],
             'min_impurity_decrease':[1e-5,1e-4,1e-3,1e-2,1e-1,0,1,10]}

kfold_grid_search(clf, params, x2, y2, fold = 10, kfold=15, search= False )
```

```
R2 metric :
0 r2 score of the train data 0.6491851497180579 and test data 0.6722657818421716
1 r2 score of the train data 0.6428875293997556 and test data 0.7650168179513238
2 r2 score of the train data 0.6543022991441494 and test data 0.6009548522547989
3 r2 score of the train data 0.651093238820595 and test data 0.6413429640388058
4 r2 score of the train data 0.6486153400378359 and test data 0.6732278735764214
```

```
5 r2 score of the train data 0.6500914264009571 and test data 0.6524032274886278
6 r2 score of the train data 0.650940549896302 and test data 0.6542854706156918
7 r2 score of the train data 0.6447753756785972 and test data 0.7542422390060914
8 r2 score of the train data 0.6659204711262184 and test data 0.46907503683814333
9 r2 score of the train data 0.6517637548732664 and test data 0.6296769532934854
10 r2 score of the train data 0.6605605207274278 and test data 0.49104247104708254
11 r2 score of the train data 0.6507621664241039 and test data 0.6474150149129719
12 r2 score of the train data 0.6492585389063131 and test data 0.659040789441843
13 r2 score of the train data 0.6478413435266211 and test data 0.6878571251898438
14 r2 score of the train data 0.6498242678129074 and test data 0.6548025156628345

Avg r2_score of train 0.6511881314995407 and test 0.6435099422106758
Time taken - 0.5376898010571798 min
```

In [42]:

```
rf = RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                           max_depth=3, max_features=0.8, max_leaf_nodes=None,
                           max_samples=None, min_impurity_decrease=1e-05,
                           min_impurity_split=None, min_samples_leaf=5,
                           min_samples_split=9, min_weight_fraction_leaf=0.0,
                           n_estimators=40, n_jobs=None, oob_score=False,
                           random_state=None, verbose=0, warm_start=False)

a = cross_val_score(rf, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.6411188337826735 data
```

XGBoost-Regressor

In [ ]:

```
clf = XGBRFRegressor(base_score=0.5, colsample_bylevel=1, colsample_bynode=0.8,
                colsample_bytree=1, gamma=0, learning_rate=1, max_delta_step=0,
                max_depth=15, max_features='log2', min_child_weight=1,
                missing=None, n_estimators=100, n_jobs=1, nthread=None,
                objective='reg:linear', random_state=5, reg_alpha=0,
                reg_lambda=1, scale_pos_weight=1, seed=None, silent=True,
                subsample=0.8, verbosity=1)

xparams = {'learning_rate':[0.1,0.5,0.8,1],
              'n_estimators':[70,80,100],
              'max_depth':[2,3,4],
              'colsample_bytree':[0.1,0.5,0.7,0.9,1],
```

```
              'subsample':[0.2,0.3,0.5,1],
              'gamma':[0.0001,0.001,0,0.1,0.01,0.5,1],
              'reg_alpha':[0.00001,0.0001,0.001,0.01,0.1]}


kfold_grid_search(clf, params, x2, y2, fold = 10, kfold=15, search= False)
```

```
R2 metric :
0 r2 score of the train data 0.6067502601868848 and test data 0.6374083132665591
1 r2 score of the train data 0.6037579887656499 and test data 0.6953451163213371
2 r2 score of the train data 0.6121444549439352 and test data 0.6011016381585581
3 r2 score of the train data 0.6129187708599382 and test data 0.5565082186028582
4 r2 score of the train data 0.6113841767426726 and test data 0.582448928624506
5 r2 score of the train data 0.6078223907106232 and test data 0.6276658565426563
6 r2 score of the train data 0.6084467931525492 and test data 0.6559734999802659
7 r2 score of the train data 0.603205051745912 and test data 0.7092297719647569
8 r2 score of the train data 0.6226020717930794 and test data 0.46860069407268745
9 r2 score of the train data 0.6124662945950516 and test data 0.5626195379397068
10 r2 score of the train data 0.6185700634680764 and test data 0.48692268287664975
11 r2 score of the train data 0.608856439703718 and test data 0.6154646014250558
12 r2 score of the train data 0.6061588075086194 and test data 0.6581597523169269
13 r2 score of the train data 0.6072817595908033 and test data 0.6407171369439164
14 r2 score of the train data 0.6060176585456536 and test data 0.6507545607215774

Avg r2_score of train 0.6098921988208779 and test 0.6099280206505344
Time taken - 0.6381794929504394 min
```

In [43]:

```
 xg = XGBRFRegressor(colsample_bylevel=1,
                colsample_bynode=0.8, colsample_bytree=1, gamma=0,
                learning_rate=1, max_delta_step=0, max_depth=5,
                max_features=0.95, min_child_weight=1, min_impurity_decrease=1,
                min_samples_leaf=1, min_samples_split=5, missing=None,
                n_estimators=100, n_jobs=1, nthread=None, objective='reg:linear',
                random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
                seed=None, silent=True, subsample=0.8, verbosity=1)

a = cross_val_score(xg, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.6096710572369947 data
```

AdaBoost-Regressor

In [ ]:
```
clf =  AdaBoostRegressor(base_estimator=None, learning_rate=0.0001, loss='linear',
                  n_estimators=100, random_state=10)


params = {'n_estimators'  : [100, 150, 200, ],
          'learning_rate' :[0.0001,0.001,0.01, 0.1],
          'loss' : [ 'linear', 'square', 'exponential'],
          'random_state' : [10,20,30]
          }

kfold_grid_search(clf, params, x2, y2, fold = 10, kfold = 15, search = False)
```

```
R2 metric :
0 r2 score of the train data 0.6406019024644523 and test data 0.6641913063532133
1 r2 score of the train data 0.6341117815098307 and test data 0.7675518739100321
2 r2 score of the train data 0.6448193857404521 and test data 0.601364590744293
3 r2 score of the train data 0.6420025644845055 and test data 0.6401058176407162
4 r2 score of the train data 0.6426822251515699 and test data 0.6709984599032441
5 r2 score of the train data 0.6412649710061986 and test data 0.6561873993797418
6 r2 score of the train data 0.6414199311868302 and test data 0.6550053083325136
7 r2 score of the train data 0.635692428613147 and test data 0.750110156147811
8 r2 score of the train data 0.6566658201904927 and test data 0.4667308312720433
9 r2 score of the train data 0.6433253839092598 and test data 0.6252818172649386
10 r2 score of the train data 0.6530673875304205 and test data 0.4887717640412784
11 r2 score of the train data 0.6424865119330938 and test data 0.6470462212100361
12 r2 score of the train data 0.6408561893286236 and test data 0.6616694666615984
13 r2 score of the train data 0.6397242178987328 and test data 0.6801148107924511
14 r2 score of the train data 0.6405964615369728 and test data 0.6550850005146253

Avg r2_score of train 0.6426211441656389 and test 0.6420143216112356
Time taken - 2.685768520832062 min
```

In [44]:
```
 ab =   AdaBoostRegressor(base_estimator=None, learning_rate=0.0001, loss='linear',
                  n_estimators=100, random_state=10)


a = cross_val_score(ab, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.6395565714731158 data
```

Gradient Boosting Regressor

In [ ]:

```
clf = GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.01, loss='huber',
                          max_depth=3, max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=2000,
                          n_iter_no_change=11, presort='deprecated',
                          random_state=10, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)

params = {'n_estimators' : [800,1000, 1500, 2000, 2500],
          'loss' : [ 'huber', 'exponential'],
          'learning_rate' : [0.01, 0.01, 0.1],
          'max_depth' : [3,4,5,7]
          }


kfold_grid_search(clf, params, x2, y2, fold = 10, kfold = 15, search = False)
```

```
R2 metric :
0 r2 score of the train data 0.646898352654415 and test data 0.6463232721630487
1 r2 score of the train data 0.6445815700448241 and test data 0.7630028652459149
2 r2 score of the train data 0.6638708224569014 and test data 0.5820050454376444
3 r2 score of the train data 0.6643729193689359 and test data 0.6298306073660422
4 r2 score of the train data 0.6452857437229575 and test data 0.6703858624196001
5 r2 score of the train data 0.6467163192346321 and test data 0.6466422501820477
6 r2 score of the train data 0.6520411010671734 and test data 0.6518687281616424
7 r2 score of the train data 0.6511587363040845 and test data 0.7573584179324647
8 r2 score of the train data 0.6688157545177016 and test data 0.453579856520536
9 r2 score of the train data 0.6498989292469194 and test data 0.6088874846193633
10 r2 score of the train data 0.6721223434084025 and test data 0.49579802946005624
11 r2 score of the train data 0.6556792100878011 and test data 0.6329712960936003
12 r2 score of the train data 0.6613676334095353 and test data 0.6671419303636512
13 r2 score of the train data 0.6624697875464138 and test data 0.7195585718225759
14 r2 score of the train data 0.6567325364921577 and test data 0.6894664334886403

Avg r2_score of train 0.6561341173041902 and test 0.6409880434184552
Time taken - 8.37012676000595 min
```

In [45]:

```
gb =  GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.01, loss='huber',
```

```
                           max_depth=3, max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=2000,
                           n_iter_no_change=11, presort='deprecated',
                           random_state=10, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0, warm_start=False)

a = cross_val_score(gb, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.6401325221008258 data
```

LGBM Regressor

In [ ]:

```
clf = LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
              importance_type='split', learning_rate=0.01, max_depth=3,
              min_child_samples=50, min_child_weight=0.001, min_split_gain=0.0,
              n_estimators=4000, n_jobs=-1, num_leaves=5, objective=None,
              random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True,
              subsample=1.0, subsample_for_bin=200000, subsample_freq=0)


params = {'min_child_samples' : [10, 20,50],
          'num_leaves' : [5,6],
          'max_depth' : [2, 3, 5],
          'n_estimators' : [1000,2000,4000,5000],
          'learning_rate' : [0.0001,0.001,0.01,0.1]
          }

kfold_grid_search(clf, params, x2, y2.ravel(), fold = 10, kfold = 15, search = False)
```

```
R2 metric :
0 r2 score of the train data 0.7572398922837325 and test data 0.648629927585285
1 r2 score of the train data 0.7559447382612352 and test data 0.7524994967038349
2 r2 score of the train data 0.7635946493631556 and test data 0.5796051680603062
3 r2 score of the train data 0.7616028837108777 and test data 0.6283320314911397
4 r2 score of the train data 0.756111951588603 and test data 0.6669960134479473
5 r2 score of the train data 0.7592501493147876 and test data 0.6344725589160572
6 r2 score of the train data 0.757831426306504 and test data 0.6637281324013
7 r2 score of the train data 0.7549205437978412 and test data 0.7264772198521539
```

```
8 r2 score of the train data 0.7704056362229155 and test data 0.47265909739244094
9 r2 score of the train data 0.7606748853035004 and test data 0.629765276411993
10 r2 score of the train data 0.7635879477252655 and test data 0.49182201718973395
11 r2 score of the train data 0.7608877785525503 and test data 0.6206099747367849
12 r2 score of the train data 0.7592847885219717 and test data 0.6312332206937824
13 r2 score of the train data 0.7559390326059556 and test data 0.7038297346424771
14 r2 score of the train data 0.755889171858096 and test data 0.6467577178304664

Avg r2_score of train 0.7595443650277994 and test 0.6331611724903802
Time taken - 4.181476438045502 min
```

In [46]:

```
lb =  LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
              importance_type='split', learning_rate=0.01, max_depth=3,
              min_child_samples=50, min_child_weight=0.001, min_split_gain=0.0,
              n_estimators=4000, n_jobs=-1, num_leaves=5, objective=None,
              random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True,
              subsample=1.0, subsample_for_bin=200000, subsample_freq=0)


a = cross_val_score(lb, x2, y2, scoring= 'r2', cv= 10)
print(f'R^2 Score {np.mean(a)} data' )
```

```
R^2 Score 0.6280864312506385 data
```

In [ ]:

```
estimators = [ ('rf', RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=5, max_features=0.95, max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.001,
                      min_impurity_split=None, min_samples_leaf=2,
                      min_samples_split=8, min_weight_fraction_leaf=0.0,
                      n_estimators=70, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)),

                ('xg', XGBRFRegressor(colsample_bylevel=1,
               colsample_bynode=0.8, colsample_bytree=1, gamma=0,
               learning_rate=1, max_delta_step=0, max_depth=5,
               max_features=0.95, min_child_weight=1, min_impurity_decrease=1,
               min_samples_leaf=1, min_samples_split=5, missing=None,
               n_estimators=100, n_jobs=1, nthread=None, objective='reg:linear',
               random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
               seed=None, silent=True, subsample=0.8, verbosity=1)),
```

```
             ('gb', GradientBoostingRegressor(max_depth= 2, learning_rate= 0.1, random_state= 10,n_estimators=5000,
                                 n_iter_no_change = 11))

            ]

stack = StackingRegressor(estimators= estimators,
                          final_estimator= Ridge(),
                          )
```

In [ ]:

```
cv_score = cross_val_score(stack, x2, y2.ravel(), scoring='r2', cv=10, verbose=5, n_jobs=-1)
print('Mean Score:',cv_score.mean())
print('Standard Deviation:',cv_score.std())
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
Mean Score: 0.5762229576080763
Standard Deviation: 0.08756604747415982
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed: 12.2min remaining:    0.0s
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed: 12.2min finished
```

Models fitting

In [50]:

```
for model, label in zip([dt, rf, xg, ab, gb, lb, stack], ['decision_tree','random_forest', 'xgboost',
                                                          'adaboost', 'gradient_boost', 'lighgbm', 'stack-ensemble']):

  model.fit(x2, y2)

  predict = model.predict(test.iloc[:, :-1])

  df = pd.DataFrame()

  df['ID'] = test.ID
  df['y']  = predict

  df.to_csv(label + '_select_k_best.csv', index= False)

  print('csv file done ! ' + label)
```

```
csv file done ! decision_tree
csv file done ! random_forest
csv file done ! xgboost
csv file done ! adaboost
csv file done ! gradient_boost
csv file done ! lighgbm
csv file done ! stack-ensemble
```

R^2 Score

In [55]:

```
col = ['MODELS', 'FEATURES', 'CROSS-VALIDATION' , 'PUBLIC SCORE', 'PRIVATE SCORE']

tb = PrettyTable()

tb.add_column(col[0], ['DECISION TREE', 'RANDOM FOREST', 'XGBOOST', 'ADA-BOOST', 'GRADIENT BOOSTING', 'LIGHT GBM', 'S

tb.add_column(col[1], ['TOP FEATURES USING SELECT K BEST METHOD', 'TOP FEATURES USING SELECT K BEST METHOD', 'TOP FEA
                       'TOP FEATURES USING SELECT K BEST METHOD', 'TOP FEATURES USING SELECT K BEST METHOD', 'TOP FEA
                       'TOP FEATURES USING SELECT K BEST METHOD'])

tb.add_column(col[2], ['0.6018', '0.6435', '0.6100', '0.6420', '0.6409', '0.6331', '0.5762'])

tb.add_column(col[3], ['0.54021', '0.54543', '0.54561', '0.53577', '0.53822', '0.53822', '0.55039'])

tb.add_column(col[4], ['0.52598', '0.54326', '0.53864', '0.53955', '0.53663', '0.54449', '0.54710'])

print(tb)
```

```
+-------------------+-----------------------------------------+------------------+--------------+---------------+
|       MODELS      |                 FEATURES                | CROSS-VALIDATION | PUBLIC SCORE | PRIVATE SCORE |
+-------------------+-----------------------------------------+------------------+--------------+---------------+
|   DECISION TREE   | TOP FEATURES USING SELECT K BEST METHOD |      0.6018      |   0.54021    |    0.52598    |
|   RANDOM FOREST   | TOP FEATURES USING SELECT K BEST METHOD |      0.6435      |   0.54543    |    0.54326    |
|      XGBOOST      | TOP FEATURES USING SELECT K BEST METHOD |      0.6100      |   0.54561    |    0.53864    |
|     ADA-BOOST     | TOP FEATURES USING SELECT K BEST METHOD |      0.6420      |   0.53577    |    0.53955    |
| GRADIENT BOOSTING | TOP FEATURES USING SELECT K BEST METHOD |      0.6409      |   0.53822    |    0.53663    |
|     LIGHT GBM     | TOP FEATURES USING SELECT K BEST METHOD |      0.6331      |   0.53822    |    0.54449    |
|  STCKED-ENSEMBLE  | TOP FEATURES USING SELECT K BEST METHOD |      0.5762      |   0.55039    |    0.54710    |
+-------------------+-----------------------------------------+------------------+--------------+---------------+
```

OBSERVATION

- This dataset created with help of SelectKBest feature selection method, we selected top 250 most important features. But we can see that these features are not get good score than other.
- In this sets stacked ensembles perform best and also another datasets, stacked model got 0.55039 in public and 0.54710 in private leaderboard.