Project Name: Mercedes-Benz Greener Manufacturing

In [1]:

```
#Load required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
from sklearn.metrics import mean_squared_error
%matplotlib inline
import seaborn as sns
```

LOADING THE TRAINING DATASETS

In [2]:

```
df_train=pd.read_csv(r"train.csv")
df_test=pd.read_csv(r"test.csv")
```

In [3]:

```
df_train.shape
```

Out[3]:

```
(4209, 378)
```

In [4]:

```
df_test.shape
```

Out[4]:

```
(4209, 377)
```

In [5]:

```
df_test.describe()
```

Out[5]:

	ID	X10	X11	X12	X13	X14	X15	X16	X17	
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.00(
mean	4211.039202	0.019007	0.000238	0.074364	0.061060	0.427893	0.000713	0.002613	0.008791	0.01(
std	2423.078926	0.136565	0.015414	0.262394	0.239468	0.494832	0.026691	0.051061	0.093357	0.10(
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00(
25%	2115.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00(
50%	4202.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00(
75%	6310.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.00(
max	8416.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00(

8 rows × 369 columns

In [6]:

```
df_test.columns
```

Out[6]:

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

In [7]:

```
df_train.columns
```

Out[7]:

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

In [8]:

```
df_test.dtypes
```

Out[8]:

```
ID        int64
X0       object
X1       object
X2       object
X3       object
          ...
X380      int64
X382      int64
X383      int64
X384      int64
X385      int64
Length: 377, dtype: object
```

In [9]:

```
df_train.dtypes
```

Out[9]:

```
ID         int64
y        float64
X0        object
X1        object
X2        object
          ...
X380       int64
X382       int64
X383       int64
X384       int64
X385       int64
Length: 378, dtype: object
```

In [10]:

```
df_train.head()
```

Out[10]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	0	130.81	k	v	at	a	d	u	j	o	...	0	0	1	0	0	0	0	0	0	0
1	6	88.53	k	t	av	e	d	y	l	o	...	1	0	0	0	0	0	0	0	0	0
2	7	76.26	az	w	n	c	d	x	j	x	...	0	0	0	0	0	0	1	0	0	0

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
3	9	80.62	az	t	n	f	d	x	l	e	...	0	0	0	0	0	0	0	0	0	0
4	13	78.02	az	v	n	f	d	h	d	n	...	0	0	0	0	0	0	0	0	0	0

5 rows × 378 columns

In [11]:

```
df_test.head()
```

Out[11]:

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	1	az	v	n	f	d	t	a	w	0	...	0	0	0	1	0	0	0	0	0	0
1	2	t	b	ai	a	d	b	g	y	0	...	0	0	1	0	0	0	0	0	0	0
2	3	az	v	as	f	d	a	j	j	0	...	0	0	0	1	0	0	0	0	0	0
3	4	az	l	n	f	d	z	l	n	0	...	0	0	0	1	0	0	0	0	0	0
4	5	w	s	as	c	d	y	i	m	0	...	1	0	0	0	0	0	0	0	0	0

5 rows × 377 columns

In [12]:

```
df_train.drop("ID",inplace=True,axis=1)
df_test.drop("ID",inplace=True,axis=1)
```

In [13]:

```
df_train.head()
```

Out[13]:

	y	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	130.81	k	v	at	a	d	u	j	o	0	...	0	0	1	0	0	0	0	0	0	0
1	88.53	k	t	av	e	d	y	l	o	0	...	1	0	0	0	0	0	0	0	0	0
2	76.26	az	w	n	c	d	x	j	x	0	...	0	0	0	0	0	0	1	0	0	0
3	80.62	az	t	n	f	d	x	l	e	0	...	0	0	0	0	0	0	0	0	0	0
4	78.02	az	v	n	f	d	h	d	n	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 377 columns

If for any column(s), the variance is equal to zero, then you need to remove those variable(s).

In [14]:

```
zero_variance_columns=df_train.var()[df_train.var()==0].index.values
```

In [15]:

```
zero_variance_columns
```

Out[15]:

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

Lets drop the zero variance columns

In [16]:

```
df_train.drop(zero_variance_columns, inplace=True, axis=1)
```

```
df_test.drop(zero_variance_columns, inplace=True, axis=1)
```

In [17]:

```
df_train.shape
```

Out[17]:

```
(4209, 365)
```

Lets check if any columns has null or NaN values

In [18]:

```
np.sum(df_train.isnull().sum())
```

Out[18]:

```
0
```

In [19]:

```
df_train.isnull().sum()
```

Out[19]:

```
y       0
X0      0
X1      0
X2      0
X3      0
       ..
X380    0
X382    0
X383    0
X384    0
X385    0
Length: 365, dtype: int64
```

In [20]:

```
#That's good...nothing needs to be done
```

Which columns are non numeric?

In [21]:

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

Out[21]:

	X0	X1	X2	X3	X4	X5	X6	X8
count	4209	4209	4209	4209	4209	4209	4209	4209
unique	47	27	44	7	4	29	12	25
top	z	aa	as	c	d	v	g	j
freq	360	833	1659	1942	4205	231	1042	277

In [22]:

```
objectColumns=df_train.describe(include=['object']).columns.values
```

In [23]:

```
objectColumns
```

```
objectColumns
```

Out[23]:

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

In [24]:

```
#Lets label encode them|same as factoring the char in R,
#we do this because ML model cant proces non numerical data
```

In [25]:

```
encoder=LabelEncoder()
```

In [26]:

```
for col in objectColumns:
    encoder.fit(df_train[col].append(df_test[col]).values)
    df_train[col] = encoder.transform(df_train[col])
    df_test[col] = encoder.transform(df_test[col])
    print("Unique classes for {} are:  {}\n".format(col,encoder.classes_ ))
```

```
Unique classes for X0 are:  ['a' 'aa' 'ab' 'ac' 'ad' 'ae' 'af' 'ag' 'ai' 'aj' 'ak' 'al' 'am' 'an'
'ao'
 'ap' 'aq' 'as' 'at' 'au' 'av' 'aw' 'ax' 'ay' 'az' 'b' 'ba' 'bb' 'bc' 'c'
 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n' 'o' 'p' 'q' 'r' 's' 't' 'u'
 'v' 'w' 'x' 'y' 'z']

Unique classes for X1 are:  ['a' 'aa' 'ab' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n' 'o'
'p'
 'q' 'r' 's' 't' 'u' 'v' 'w' 'y' 'z']

Unique classes for X2 are:  ['a' 'aa' 'ab' 'ac' 'ad' 'ae' 'af' 'ag' 'ah' 'ai' 'aj' 'ak' 'al' 'am'
'an'
 'ao' 'ap' 'aq' 'ar' 'as' 'at' 'au' 'av' 'aw' 'ax' 'ay' 'b' 'c' 'd' 'e'
 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n' 'o' 'p' 'q' 'r' 's' 't' 'u' 'w' 'x'
 'y' 'z']

Unique classes for X3 are:  ['a' 'b' 'c' 'd' 'e' 'f' 'g']

Unique classes for X4 are:  ['a' 'b' 'c' 'd']

Unique classes for X5 are:  ['a' 'aa' 'ab' 'ac' 'ad' 'ae' 'af' 'ag' 'ah' 'b' 'c' 'd' 'f' 'g' 'h' '
i'
 'j' 'k' 'l' 'm' 'n' 'o' 'p' 'q' 'r' 's' 't' 'u' 'v' 'w' 'x' 'y' 'z']

Unique classes for X6 are:  ['a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l']

Unique classes for X8 are:  ['a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n' 'o' 'p' 'q'
'r'
 's' 't' 'u' 'v' 'w' 'x' 'y']
```

In [27]:

```
df_train.head()
```

Out[27]:

	y	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	130.81	37	23	20	0	3	27	9	14	0	...	0	0	1	0	0	0	0	0	0	0
1	88.53	37	21	22	4	3	31	11	14	0	...	1	0	0	0	0	0	0	0	0	0
2	76.26	24	24	38	2	3	30	9	23	0	...	0	0	0	0	0	0	1	0	0	0
3	80.62	24	21	38	5	3	30	11	4	0	...	0	0	0	0	0	0	0	0	0	0
4	78.02	24	23	38	5	3	14	3	13	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 365 columns

In [28]:

```
#Now the data preprocessing is done
```

We have too many columns, so lets try and pack the information in fewer columns

In [29]:

```
#we need atleat 98% information to be reatined, and accordingly have
#the extracted features of highest variance
pcamodel=PCA(0.98,svd_solver='full')
X=df_train.drop('y',axis=1)
y=df_train['y']
pcamodel.fit(X)
```

Out[29]:

```
PCA(copy=True, iterated_power='auto', n_components=0.98, random_state=None,
    svd_solver='full', tol=0.0, whiten=False)
```

In [30]:

```
# these components are orthogonal to each other and are independent to each other
#these features are of highest variance and are itself calculating infogain
pcamodel.n_components_
```

Out[30]:

```
12
```

In [31]:

```
#In 12 derived columns we are able to fit 98% of information..though we had 365 columns
```

In [32]:

```
pcamodel.explained_variance_ratio_
```

Out[32]:

```
array([0.40868988, 0.21758508, 0.13120081, 0.10783522, 0.08165248,
       0.0140934 , 0.00660951, 0.00384659, 0.00260289, 0.00214378,
       0.00209857, 0.00180388])
```

In [33]:

```
#in the above data, we can see 0.40 says it has 40 % of information and so on
```

In [34]:

```
#Divide the training data between training and validation
X_train,X_val,y_train,y_val=train_test_split(X,y,test_size=0.2,random_state=42)
```

In [35]:

```
compactedDerivedFeatures_train=pd.DataFrame(pcamodel.transform(X_train))
```

In [36]:

```
compactedDerivedFeatures_train.describe()
```

Out[36]:

	0	1	2	3	4	5	6	7	8	
count	3367.000000	3367.000000	3367.000000	3367.000000	3367.000000	3367.000000	3367.000000	3367.000000	3367.000000	3367.000
mean	-0.026799	-0.211816	0.029763	-0.065402	-0.063075	0.015586	-0.007407	-0.006182	0.000833	0.009
std	15.648904	11.364890	8.933146	8.056159	7.043068	2.896704	1.995723	1.528100	1.261684	1.143
min	-23.071496	-22.697173	-17.883045	-19.616685	-13.583307	-5.040693	-4.377823	-3.289078	-3.196062	-2.844
25%	-13.331227	-7.660939	-8.274029	-6.497304	-5.920797	-2.130067	-1.941183	-1.503764	-0.993944	-0.634
50%	-4.380650	-2.516139	0.897065	-1.182782	-0.603521	-0.205659	0.408907	0.405359	-0.028099	-0.053
75%	12.427669	6.220204	7.167600	6.542448	5.957571	1.240799	1.497018	1.179973	0.808253	0.521
max	39.025839	32.845313	18.839900	20.409804	14.308816	7.519440	4.685064	3.447685	3.612572	5.094

In [37]:

```
compactedDerivedFeatures_train.head()
```

Out[37]:

	0	1	2	3	4	5	6	7	8	9	10	11
0	-2.419452	-0.611007	-3.843534	15.051223	10.517966	-3.291361	2.242806	-1.391341	-1.641953	-0.781943	-0.670463	0.249792
1	-3.754406	-0.156986	-6.634242	12.931294	5.700700	0.689493	0.326794	-0.910293	-0.805958	1.301613	0.272010	1.246175
2	4.723647	15.727164	10.172595	-12.071983	-6.283282	-0.600279	0.773690	0.710189	-2.106015	-2.510619	-0.689292	-0.937849
3	-0.926505	11.287117	-3.012766	-10.112258	-6.317545	-0.661537	-0.239300	0.559985	-1.189790	2.587546	-0.680997	0.724748
4	10.395983	-3.471925	-11.784807	-0.533004	-0.975479	-4.639907	1.411170	-0.739972	3.299822	0.739134	-0.783735	-0.130903

Lets check the correlation between extracted features and output var

In [38]:

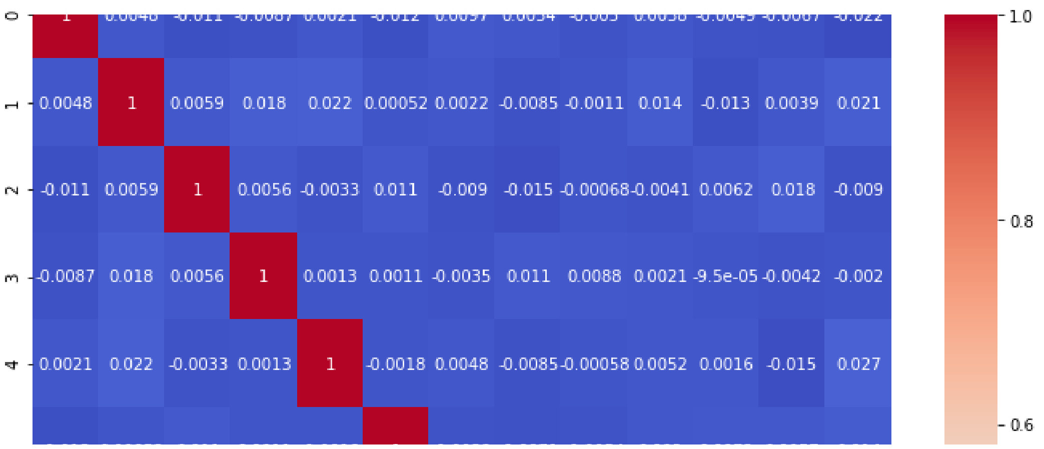
```
compactedDerivedFeaturesAndTarget_train = pd.concat([compactedDerivedFeatures_train, y_train], axi
s=1)
```

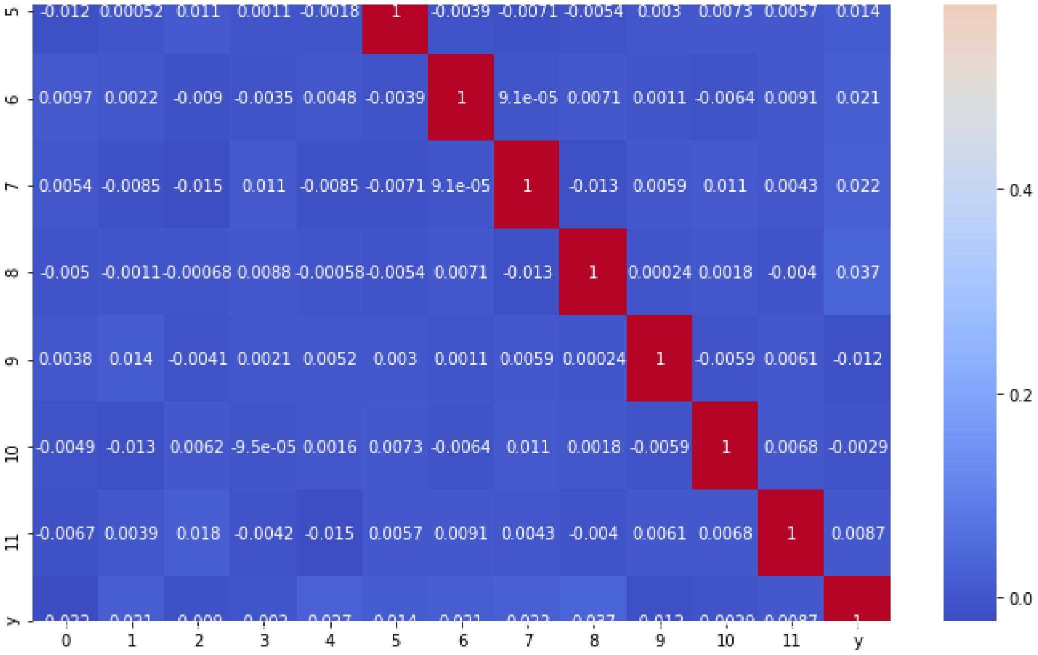
In [39]:

```
plt.figure(figsize=(12,12))
corrMap=compactedDerivedFeaturesAndTarget_train.corr()
sns.heatmap(corrMap,annot=True,cmap='coolwarm')
```

Out[39]:

<matplotlib.axes._subplots.AxesSubplot at 0x20a4b5e48>





In [40]:

```
compactedDerivedFeatures_val = pd.DataFrame(pcamodel.transform(X_val))
compactedDerivedFeatures_test = pd.DataFrame(pcamodel.transform(df_test))
```

In [41]:

```
compactedDerivedFeatures_val
```

Out[41]:

	0	1	2	3	4	5	6	7	8	9	10	11
0	20.612140	-16.319893	-7.939029	2.691741	-0.674573	-2.747793	-2.127039	2.941831	1.305954	-0.095223	2.018229	-1.100170
1	-1.123824	-16.036199	0.415082	2.916770	9.355356	-1.364116	-1.878937	1.862838	-1.381127	-0.221185	-1.173805	-1.616512
2	-2.749224	4.808027	2.000871	-10.738490	3.243479	-3.692114	-0.056620	-0.294929	-1.078660	0.316824	0.755149	0.619547
3	14.199442	3.150340	3.270692	10.577979	-0.734433	-2.593174	1.135354	1.247012	0.809166	-0.595677	-0.603789	4.048736
4	23.355595	-15.268156	5.729429	5.951943	4.790395	-1.792474	0.231251	0.651460	-0.068891	-0.218094	0.152707	-1.589765
...	...	...	...	...	...	...	...	...	...	...	...	...
837	-12.702669	19.891930	-6.001308	-3.784697	2.403210	4.510144	1.121552	-0.027753	0.852773	-0.758725	-0.661310	-1.200741
838	-9.475260	18.149236	12.168198	-9.639324	11.277108	-1.476472	0.884837	1.099494	-0.202077	-1.778138	0.242657	-1.791338
839	-17.944041	8.607653	-9.346593	-3.419816	6.829281	0.506923	-0.082796	-0.175111	-0.994451	0.131447	0.515479	0.645915
840	1.186011	-15.753999	11.508313	8.828140	-1.383177	0.163606	2.650603	0.864410	-1.211257	-1.432974	-0.346209	-0.081356
841	-12.272783	2.793723	6.587915	12.654497	-11.902774	-2.731905	-1.639985	-1.921521	1.377776	1.122107	-0.306503	-0.806541

842 rows × 12 columns

In [42]:

```
compactedDerivedFeatures_train
```

Out[42]:

	0	1	2	3	4	5	6	7	8	9	10	11
0	-2.419452	-0.611007	-3.843534	15.051223	10.517966	-3.291361	2.242806	-1.391341	-1.641953	-0.781943	-0.670463	0.249792
1	-3.754406	-0.156986	-6.634242	12.931294	5.700700	0.689493	0.326794	-0.910293	-0.805958	1.301613	0.272010	1.246175
2	4.723647	15.727164	10.172595	-12.071983	-6.283282	-0.600279	0.773690	0.710189	-2.106015	-2.510619	-0.689292	-0.937849
3	-0.926505	11.287117	-3.012766	-10.112258	-6.317545	-0.661537	-0.239300	0.559985	-1.189790	2.587546	-0.680997	0.724748
4	10.395983	-3.471925	-11.784807	-0.533004	-0.975479	-4.639907	1.411170	-0.739972	3.299822	0.739134	-0.783735	-0.130903
...	...	...	...	...	...	...	...	...	...	...	...	...
3362	-2.347723	-1.290701	8.343259	-3.238105	5.252336	-4.444889	1.539155	-1.945461	2.149679	0.170514	-1.249758	-0.640747
3363	14.998405	6.241171	-0.571118	11.500606	-2.515762	-2.548913	0.856422	0.674280	1.068246	-1.643639	-0.688946	3.266046
3364	-16.415233	-10.734672	11.499726	12.948982	-11.896522	-2.130543	0.981542	1.621301	1.858401	1.066422	0.879612	0.406846
3365	-16.057694	-5.014249	14.981020	6.639199	-12.324817	-1.734501	-1.966853	0.751376	1.483096	0.279720	-0.188072	-0.566702
3366	2.991791	-12.884183	-12.100166	-7.750691	5.775726	-1.999601	0.600172	1.729449	-1.690808	0.128228	2.863109	0.173386

3367 rows × 12 columns

Lets use the Xgboost

Xgboost is used when we want incrementally strenghthen the model

In [43]:

```
import xgboost as xgb
from sklearn.ensemble import RandomForestRegressor
regressor=xgb.XGBRegressor(objective='reg:linear',learning_rate=0.1)
```

In [44]:

```
compactedDerivedFeatures_train.columns
```

Out[44]:

```
RangeIndex(start=0, stop=12, step=1)
```

In [45]:

```
compactedDerivedFeatures_train.columns
```

Out[45]:

```
RangeIndex(start=0, stop=12, step=1)
```

In [46]:

```
compactedDerivedFeatures_train.shape
```

Out[46]:

```
(3367, 12)
```

In [47]:

```
compactedDerivedFeatures_train.columns
```

Out[47]:

```
RangeIndex(start=0, stop=12, step=1)
```

In [48]:

```
compactedDerivedFeatures_train.dtypes
```

Out[48]:

```
0     float64
1     float64
2     float64
3     float64
4     float64
5     float64
6     float64
7     float64
8     float64
9     float64
10    float64
11    float64
dtype: object
```

In [49]:

```
regressor.fit(compactedDerivedFeatures_train,y_train)
```

```
[20:30:31] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.0.0/src/objective/regression_obj.cu:167: reg:linear is now deprecated in favor of
reg:squarederror.
```

Out[49]:

```
XGBRegressor(base_score=0.5, booster=None, colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints=None,
             learning_rate=0.1, max_delta_step=0, max_depth=6,
             min_child_weight=1, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=0, num_parallel_tree=1,
             objective='reg:linear', random_state=0, reg_alpha=0, reg_lambda=1,
             scale_pos_weight=1, subsample=1, tree_method=None,
             validate_parameters=False, verbosity=None)
```

In [50]:

```
compactedDerivedFeatures_val.shape
```

Out[50]:

```
(842, 12)
```

In [51]:

```
compactedDerivedFeatures_val.dtypes
```

Out[51]:

```
0     float64
1     float64
2     float64
3     float64
4     float64
5     float64
6     float64
7     float64
8     float64
9     float64
10    float64
11    float64
dtype: object
```

In [52]:

```
predictions_val = regressor.predict(compactedDerivedFeatures_val)
```

In [53]:

```
# Predicted values for test_df
predictions_val
```

Out[53]:

```
array([ 92.372284,  97.92539 , 105.53712 ,  78.17668 , 111.48294 ,
       102.18538 ,  91.53497 , 103.4876  , 107.94279 , 117.08574 ,
        78.733955,  95.34357 ,  93.147675, 102.92732 ,  95.4187  ,
        95.12832 , 109.39629 ,  94.63033 ,  94.97428 , 114.50211 ,
       115.29926 ,  93.76525 ,  94.8049  ,  97.27465 ,  95.59408 ,
       111.38121 ,  95.15951 ,  77.95871 ,  93.70483 ,  92.989525,
        93.38617 , 102.854965,  96.8341  , 108.51486 , 102.876366,
       120.019844, 110.03971 ,  99.36152 ,  92.67663 , 101.31356 ,
       123.72252 , 107.032   , 124.59482 , 109.74334 ,  94.58666 ,
       101.45895 ,  96.27568 , 103.98878 , 109.776634, 103.63502 ,
        94.32581 , 100.031364, 102.83121 , 106.03851 ,  97.32185 ,
        98.27858 ,  96.63149 , 113.18765 ,  94.50534 ,  96.85498 ,
       107.394   ,  76.671   ,  95.28154 ,  93.61442 ,  77.42669 ,
       100.67115 ,  94.01579 , 103.824356, 103.63502 ,  96.15673 ,
        93.90952 ,  91.80348 ,  95.663475, 108.11635 ,  93.111626,
       108.647644,  95.69076 , 112.29617 ,  99.84363 ,  97.930504,
       109.50842 , 117.984764, 108.40691 , 111.03229 , 108.722755,
        97.0893  ,  98.254845, 107.03092 ,  97.99063 , 108.46279 ,
        93.42674 ,  99.713066,  96.761955, 107.49975 , 110.060936,
        94.226524, 100.2418  , 114.169876, 100.63462 ,  95.36932 ,
        80.34224 , 101.879265,  94.30691 ,  91.697876, 102.08015 ,
       104.365585,  94.22806 ,  97.64746 , 107.19142 ,  97.81396 ,
       101.879265,  96.870766, 102.25266 , 102.72184 , 102.001495,
        76.04996 , 102.17107 ,  97.2712  , 102.65381 , 111.44459 ,
       105.757935, 110.69644 ,  99.043495, 111.750496, 100.76185 ,
        96.07676 ,  97.09051 , 114.40499 ,  94.29758 , 100.37597 ,
       109.16254 ,  95.589005,  75.81683 , 100.489044, 116.69893 ,
       100.370544,  99.21952 , 102.464836, 110.506546,  92.80124 ,
        77.513374, 106.43431 , 101.25086 , 116.28364 ,  93.01373 ,
        93.53316 , 108.8488  , 107.19905 , 111.17154 , 106.072426,
       110.90172 ,  95.457146,  96.440865, 110.65191 , 114.85608 ,
       115.04763 ,  95.86993 ,  95.95457 , 100.53513 , 109.52572 ,
       105.28474 ,  97.53115 ,  91.56795 ,  91.697876, 106.95908 ,
        95.90212 ,  97.46205 , 109.38023 , 130.63069 , 104.51891 ,
        76.75685 , 101.617096,  91.61373 , 100.344765,  94.96339 ,
       114.38973 ,  93.2888  ,  92.49183 ,  97.45872 , 110.11005 ,
       104.13437 , 102.693565, 112.948456,  92.48128 , 110.87021 ,
        93.08315 , 107.43688 , 101.90082 , 100.19304 , 102.11465 ,
       113.93996 , 115.087585,  92.802   , 108.027756,  93.06365 ,
        99.0497  , 101.121056,  91.352425, 105.87193 , 109.756775,
        91.437256, 105.92437 , 105.98011 ,  94.81802 ,  93.54442 ,
       111.267586, 105.32394 , 103.873955, 102.6198  ,  97.79788 ,
        94.80204 ,  91.697876,  92.36509 ,  93.111626,  93.96984 ,
        92.16666 ,  93.01521 ,  96.544395, 106.10317 , 101.92842 ,
       112.307304,  94.23288 , 103.16806 ,  96.09123 , 114.92254 ,
        95.06207 ,  94.8325  , 110.265656, 111.009254,  96.02191 ,
        94.94552 ,  91.95992 ,  94.10852 ,  94.16548 , 109.44839 ,
       105.220085,  93.722374, 107.14183 ,  93.047966,  90.13656 ,
        77.634445, 102.8339  , 123.34458 , 101.66218 ,  91.89429 ,
        93.5122  ,  92.20242 , 103.92506 ,  99.78765 ,  99.32283 ,
       104.410355, 110.22748 ,  95.27985 ,  95.58439 , 102.96478 ,
        94.43409 ,  94.631775, 106.46239 , 117.53893 , 104.74167 ,
       100.50076 , 109.93844 , 110.138374,  92.764725, 105.04438 ,
       110.86548 ,  94.553604,  86.045456, 110.71386 ,  94.171844,
        99.38696 ,  94.097435, 106.89591 , 108.58108 , 108.42717 ,
       103.620964, 107.44431 ,  94.28479 ,  97.9174  ,  98.52912 ,
       102.88307 ,  96.50058 , 100.61016 ,  93.83179 ,  79.66141 ,
        93.980644,  95.774345, 107.75355 ,  92.236755, 110.40243 ,
        88.81942 , 103.66948 ,  99.49752 , 112.72576 ,  94.39859 ,
       107.464645,  80.6786  ,  95.47053 , 110.243225, 108.13457 ,
       111.039955, 105.8738  , 102.6198  ,  97.18123 , 106.389534,
       101.895004, 110.48612 , 103.31475 ,  92.756805,  92.651215,
        95.39629 ,  94.44916 ,  96.354485, 113.83557 , 102.53062 ,
        97.32046 ,  91.46458 , 101.538956,  93.8774  ,  98.414375,
       117.70199 ,  94.29758 , 111.14415 ,  92.08201 ,  79.82938 ,
```

```
108.40691 ,  95.17218 , 108.548164, 117.66598 , 111.80499 ,
118.85252 ,  93.56604 , 113.32111 ,  98.581436,  98.04511 ,
 96.802505, 106.82985 ,  99.59975 , 108.13099 , 104.25873 ,
109.15482 , 101.61291 ,  93.884254,  91.433624, 101.45621 ,
110.48612 , 110.691086,  96.063286, 111.119675, 112.2176  ,
 92.78115 , 111.7571  ,  78.39214 , 111.35966 ,  92.00956 ,
112.11057 ,  93.776436, 102.33588 , 110.408585,  94.38422 ,
 91.80315 , 111.13994 , 109.153046,  96.34493 ,  96.70456 ,
 93.59838 ,  98.69023 ,  93.26431 , 106.31313 , 103.97752 ,
 93.275604, 100.47921 , 102.382805,  93.70573 , 112.061485,
121.21636 ,  91.44441 ,  97.01022 , 101.52149 ,  95.356544,
105.289986,  95.52699 , 100.41251 , 113.27163 , 108.58781 ,
 99.11913 ,  95.91885 , 103.50487 , 114.20498 , 100.85101 ,
104.771965, 118.279076, 102.772964, 108.25443 ,  89.5525  ,
 98.618805, 103.4495  , 103.916336,  91.10401 ,  99.572754,
 76.929   , 111.54525 , 113.47724 , 108.40934 ,  94.28982 ,
110.924385, 110.7134  , 109.95891 ,  77.95871 , 103.50236 ,
105.713936,  93.581024, 101.6648  , 109.28757 ,  98.58232 ,
111.96635 , 110.48612 ,  99.19668 , 108.07523 , 106.08276 ,
105.39758 , 105.533   ,  94.4049  , 111.389885, 107.06107 ,
111.72625 , 117.1876  , 107.07971 ,  93.04381 , 114.08615 ,
104.23645 , 109.45363 ,  94.665565, 113.76102 , 109.01413 ,
102.136566, 103.17307 , 101.92842 , 106.875984,  94.14619 ,
103.497765,  96.03041 ,  99.12688 , 107.93565 ,  93.99632 ,
 93.29401 ,  94.37826 , 103.39481 , 112.29617 ,  96.2155  ,
 97.475685,  95.11213 ,  95.57758 ,  94.00284 , 105.44322 ,
 94.71584 , 122.91531 , 102.07409 , 101.34018 , 102.067215,
 99.23614 , 100.45857 ,  99.26577 ,  98.9998  , 105.760605,
117.66598 ,  94.483635, 106.90122 , 109.380806,  77.8979  ,
110.48612 ,  99.153244, 106.39893 ,  92.8728  , 110.28912 ,
106.150925,  89.683914,  96.81835 , 113.44693 ,  92.144226,
112.29617 , 110.0286  ,  77.56775 ,  98.827934,  94.28715 ,
 94.502426, 112.69377 ,  95.98716 ,  92.82443 ,  96.27501 ,
101.14896 , 117.0008  ,  98.720726, 111.56153 ,  95.22018 ,
 78.52486 ,  76.41741 ,  92.91781 ,  93.70483 ,  99.49896 ,
 97.43497 ,  94.26783 ,  94.942024, 101.07056 , 109.74039 ,
108.654076,  96.18987 ,  95.200615, 119.776634, 100.27581 ,
 95.49978 ,  97.22035 ,  98.62275 , 105.72902 , 100.9576  ,
 79.04528 , 110.7134  ,  95.23454 ,  91.00254 ,  77.95871 ,
 94.86608 ,  94.99487 ,  96.03508 , 118.02039 , 111.526344,
112.29617 ,  97.256134,  95.01948 ,  93.41141 , 103.36951 ,
104.18445 ,  96.896515,  95.35874 ,  94.872635,  96.00513 ,
115.98028 , 102.260735,  95.9224  ,  94.30681 , 102.25603 ,
 77.79164 , 106.77711 ,  95.88031 , 114.40499 , 103.20938 ,
108.20958 , 112.97095 ,  95.18289 , 107.92462 , 115.25919 ,
102.41061 ,  91.42268 ,  92.00468 ,  94.884964, 106.50975 ,
 89.99088 , 102.317215, 102.83775 , 100.55083 , 109.43237 ,
112.13198 , 102.18665 , 111.15156 , 110.54858 , 110.51614 ,
108.41888 ,  97.193275, 109.236275, 106.04332 , 109.738045,
 93.35795 ,  90.220764, 104.24451 ,  96.55311 ,  91.364296,
 97.652916, 117.58473 , 110.93535 ,  96.25541 , 115.45399 ,
 96.6087  , 110.66947 , 109.37108 , 110.26683 , 104.435936,
104.25182 , 102.09044 ,  95.75879 , 102.815216, 106.18065 ,
106.04324 , 111.63316 ,  79.169205,  99.66941 , 100.29105 ,
109.46506 ,  96.05021 ,  77.79634 ,  94.60032 ,  90.78525 ,
101.01803 ,  93.24653 ,  96.891045,  95.031975,  95.36828 ,
 91.86589 , 103.101364,  76.97899 , 102.31815 ,  89.95941 ,
 97.63841 ,  78.85132 , 100.46968 , 108.37641 , 110.075516,
112.736115,  94.00512 ,  98.18743 , 107.706116, 100.613365,
110.639626,  94.2699  ,  77.89916 , 120.53361 , 100.38619 ,
 98.372536, 110.180405, 110.04202 , 107.360855, 100.24    ,
101.03174 ,  90.41143 ,  95.386375,  98.19896 , 125.484566,
104.25479 , 110.88251 ,  92.48205 , 111.16505 , 102.53629 ,
104.19979 ,  79.01128 , 111.37309 ,  96.10103 , 114.16982 ,
114.76498 , 119.28924 ,  93.88485 , 103.046234, 101.05331 ,
109.84993 , 117.671326,  90.99441 ,  93.62031 ,  76.00516 ,
119.69326 , 123.34458 , 105.272026,  95.84506 , 114.44972 ,
 93.19055 , 104.08929 ,  93.741875, 104.44882 ,  92.82443 ,
108.038216, 149.55583 , 111.896706, 108.58816 , 104.122345,
104.53548 ,  93.58274 ,  79.384445,  94.25859 ,  91.22594 ,
 78.097725, 103.26982 , 104.51197 , 104.18075 , 100.87105 ,
 96.15687 , 102.04494 , 103.5467  , 107.784386, 103.89648 ,
 96.034065,  98.8276  , 107.40328 ,  93.12604 ,  95.31345 ,
104.32937 ,  96.53481 ,  94.91513 ,  77.86689 ,  97.073044,
112.29377 , 102.77217 , 107.87226 , 107.04358 , 107.74389 ,
 94.80413 ,  96.241394, 100.19786 ,  98.75547 ,  94.22975 ,
114.65699 ,  91.98309 , 106.072426,  98.880684, 100.09432 ,
```

```
       109.10479 , 104.77586 , 108.98334 , 104.33127 , 125.30266 ,
        92.48353 ,  95.378075,  93.463066,  95.95413 , 104.727036,
       113.63795 , 101.17951 ,  96.44008 ,  94.076485,  79.01128 ,
        94.263214, 112.10622 ,  97.81413 , 103.73991 , 106.45638 ,
        96.673294,  95.54144 ,  99.07312 ,  77.99097 ,  94.10802 ,
       102.11674 ,  98.73931 , 104.36535 , 107.05322 , 109.76488 ,
       105.88774 ,  93.70928 ,  95.48396 ,  98.82992 , 106.802086,
        93.70483 , 108.478195, 112.88941 ,  93.75348 ,  78.604675,
        93.51407 ,  96.14093 , 102.71265 , 102.542496,  93.67723 ,
        93.79802 ,  95.614716,  97.9388  ,  98.48574 , 100.27282 ,
        96.17427 , 102.83002 ,  80.85536 ,  93.87166 , 103.07159 ,
       106.86675 ,  99.13204 ,  93.29401 , 114.302284, 110.48612 ,
       100.11883 ,  96.509415, 117.72692 ,  94.78217 , 110.48575 ,
       106.6565  ,  97.11362 ,  96.838295,  95.74276 ,  95.29114 ,
        89.38183 ,  99.518425,  94.29663 , 107.15875 , 102.772964,
        97.88271 , 109.76114 ,  76.79471 ,  94.006645,  97.22305 ,
       104.02748 , 102.314026,  94.20501 , 118.21795 ,  95.23863 ,
       102.90499 ,  97.02768 ,  99.17879 ,  97.66618 , 112.75941 ,
        93.20259 ,  92.57567 ,  95.19576 , 102.63008 , 110.03748 ,
        93.00837 ,  92.55716 , 101.304344,  98.45057 ,  95.833015,
       109.96669 , 101.88462 ,  96.037704,  94.59356 , 106.8573  ,
        95.89016 , 116.052666,  90.6794  , 103.369156, 101.30014 ,
       116.186325, 104.44807 ,  92.93815 ,  97.68635 ,  77.79779 ,
        93.71588 , 102.18825 ,  80.30446 ,  93.66172 ,  94.6532  ,
        97.14279 , 106.08888 ,  95.60365 , 107.802505, 114.55388 ,
        95.178116, 103.983955, 104.89429 ,  93.7005  ,  95.408104,
        93.03886 , 107.21901 ], dtype=float32)
```

In [54]:

```
mse_score = mean_squared_error(y_val, predictions_val)
print("MSE is:",mse_score)
```

```
MSE is: 80.85414063359927
```

In [55]:

```
from sklearn.metrics import r2_score
print("R2 score is : ",r2_score(y_val, predictions_val) )
```

```
R2 score is :  0.4805385673885587
```

In [ ]:

```

```