In [117…

```
import pandas as pd
import numpy as np
```

In [118…

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

In [119…

```
# Check for null values in training as well as testing data
```

In [120…

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

Out[120…

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

In [121…

```
df_test.isnull().sum()
```

Out[121…

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

In [122…

```
df_train.head()
```

Out[122…

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X382
0	0	130.81	k	v	at	a	d	u	j	o	...	0	0	1	0	0	0	0
1	6	88.53	k	t	av	e	d	y	l	o	...	1	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
3	9	80.62	az	t	n	f	d	x	l	e	...	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

5 rows × 378 columns

In [123…

```
df_train.dtypes.value_counts()
```

Out[123…

```
int64      369
object       8
float64      1
dtype: int64
```

In [124…

```
df_test.dtypes.value_counts()
```

Out[124…

```
int64     369
object      8
dtype: int64
```

In [125…

```
df_test.head()
```

Out[125…

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

5 rows × 377 columns

In [126…

```
# Drop the ID column from the tet data and ID and y from the training data
# No null values are found in the data
# Check for the columns with zero variance and remove those columns
# Before that seperate the dependent and independent variable
y_train = df_train["y"]
df_id = df_train["ID"]
df_train = df_train.drop(["ID", "y"], axis = 1)
df_test = df_test.drop("ID", axis = 1)
```

In [127…

```
df_train.head()
```

Out[127…

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	...	X375	X376	X377	X378	X379	X380	X382
0	k	v	at	a	d	u	j	o	0	0	...	0	0	1	0	0	0	0
1	k	t	av	e	d	y	l	o	0	0	...	1	0	0	0	0	0	0
2	az	w	n	c	d	x	j	x	0	0	...	0	0	0	0	0	0	1
3	az	t	n	f	d	x	l	e	0	0	...	0	0	0	0	0	0	0
4	az	v	n	f	d	h	d	n	0	0	...	0	0	0	0	0	0	0

5 rows × 376 columns

In [128…

```
df_test.head()
```

Out[128…

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	...	X375	X376	X377	X378	X379	X380	X382

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	...	X375	X376	X377	X378	X379	X380	X382
0	az	v	n	f	d	t	a	w	0	0	...	0	0	0	1	0	0	0
1	t	b	ai	a	d	b	g	y	0	0	...	0	0	1	0	0	0	0
2	az	v	as	f	d	a	j	j	0	0	...	0	0	0	1	0	0	0
3	az	l	n	f	d	z	l	n	0	0	...	0	0	0	1	0	0	0
4	w	s	as	c	d	y	i	m	0	0	...	1	0	0	0	0	0	0

5 rows × 376 columns

In [129...

```
y_train.head()
```

Out[129...

```
0    130.81
1     88.53
2     76.26
3     80.62
4     78.02
Name: y, dtype: float64
```

In [130...

```
# Seperate the categorical and numerical data for the training and the testing data
# So that we can eaisly preprocess the data
df_num_train = df_train.select_dtypes(exclude = np.object)
df_cat_train = df_train.select_dtypes(include = np.object)
df_num_test = df_test.select_dtypes(exclude = np.object)
df_cat_test = df_test.select_dtypes(include = np.object)
```

```
<ipython-input-130-7c8aa6cfa58a>:3: DeprecationWarning: `np.object` is a deprecated
alias for the builtin `object`. To silence this warning, use `object` by itself. Doi
ng this will not modify any behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/r
elease/1.20.0-notes.html#deprecations
  df_num_train = df_train.select_dtypes(exclude = np.object)
<ipython-input-130-7c8aa6cfa58a>:4: DeprecationWarning: `np.object` is a deprecated
alias for the builtin `object`. To silence this warning, use `object` by itself. Doi
ng this will not modify any behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/r
elease/1.20.0-notes.html#deprecations
  df_cat_train = df_train.select_dtypes(include = np.object)
<ipython-input-130-7c8aa6cfa58a>:5: DeprecationWarning: `np.object` is a deprecated
alias for the builtin `object`. To silence this warning, use `object` by itself. Doi
ng this will not modify any behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/r
elease/1.20.0-notes.html#deprecations
  df_num_test = df_test.select_dtypes(exclude = np.object)
<ipython-input-130-7c8aa6cfa58a>:6: DeprecationWarning: `np.object` is a deprecated
alias for the builtin `object`. To silence this warning, use `object` by itself. Doi
ng this will not modify any behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/r
elease/1.20.0-notes.html#deprecations
  df_cat_test = df_test.select_dtypes(include = np.object)
```

In [131...

```
df_num_train.head()
```

Out[131...

	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	...	X375	X376	X377	X378	X379	X38
0	0	0	0	1	0	0	0	0	1	0	...	0	0	1	0	0	
1	0	0	0	0	0	0	0	0	1	0	...	1	0	0	0	0	

	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	...	X375	X376	X377	X378	X379	X38
2	0	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	

5 rows × 368 columns

In [132…

```
df_num_test.head()
```

Out[132…

	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	...	X375	X376	X377	X378	X379	X38
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	
1	0	0	0	0	0	0	0	0	0	1	...	0	0	1	0	0	
2	0	0	0	0	1	0	0	0	0	0	...	0	0	0	1	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	
4	0	0	0	0	1	0	0	0	0	0	...	1	0	0	0	0	

5 rows × 368 columns

In [133…

```
# Now search for the columns which has zero variance and remove them
# Because those columns are of no use for us to build a model
v = np.array(df_num_train.var())
col_names = list(df_num_train.columns)
for i in range(368):
    if v[i] ==0:
        df_num_train = df_num_train.drop(col_names[i],  axis =1)
        df_num_test = df_num_test.drop(col_names[i], axis = 1)
```

In [134…

```
## All the columns with zero variance is reoved from the taining and testing numeric
df_num_train.head()
```

Out[134…

	X10	X12	X13	X14	X15	X16	X17	X18	X19	X20	...	X375	X376	X377	X378	X379	X38
0	0	0	1	0	0	0	0	1	0	0	...	0	0	1	0	0	
1	0	0	0	0	0	0	0	1	0	0	...	1	0	0	0	0	
2	0	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	

5 rows × 356 columns

In [135…

```
df_num_test.head()
```

Out[135…

	X10	X12	X13	X14	X15	X16	X17	X18	X19	X20	...	X375	X376	X377	X378	X379	X38

	X10	X12	X13	X14	X15	X16	X17	X18	X19	X20	...	X375	X376	X377	X378	X379	X38
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	
1	0	0	0	0	0	0	0	0	1	0	...	0	0	1	0	0	
2	0	0	0	1	0	0	0	0	0	0	...	0	0	0	1	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	
4	0	0	0	1	0	0	0	0	0	0	...	1	0	0	0	0	

5 rows × 356 columns

In [136...

```
# Now apply the label encoder on the categorical columns in the training and testing
from sklearn.preprocessing import LabelEncoder
```

In [137...

```
le = LabelEncoder()
col_names = list(df_cat_train.columns)
for i in col_names:
    df_cat_train[i] = le.fit_transform(df_cat_train[i])
    df_cat_test[i] = le.fit_transform(df_cat_test[i])
```

```
<ipython-input-137-57cc79ed0218>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df_cat_train[i] = le.fit_transform(df_cat_train[i])
<ipython-input-137-57cc79ed0218>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df_cat_test[i] = le.fit_transform(df_cat_test[i])
```

In [138...

```
df_cat_train.head()
```

Out[138...

	X0	X1	X2	X3	X4	X5	X6	X8
0	32	23	17	0	3	24	9	14
1	32	21	19	4	3	28	11	14
2	20	24	34	2	3	27	9	23
3	20	21	34	5	3	27	11	4
4	20	23	34	5	3	12	3	13

In [139...

```
df_cat_test.head()
```

Out[139...

	X0	X1	X2	X3	X4	X5	X6	X8
0	21	23	34	5	3	26	0	22
1	42	3	8	0	3	9	6	24

	X0	X1	X2	X3	X4	X5	X6	X8
2	21	23	17	5	3	0	9	9
3	21	13	34	5	3	31	11	13
4	45	20	17	2	3	30	8	12

In [140…

```
df_cat_train.dtypes.value_counts()
```

Out[140…

```
int32    8
dtype: int64
```

In [141…

```
df_cat_test.dtypes.value_counts()
```

Out[141…

```
int32    8
dtype: int64
```

In [142…

```
# Now apply the MinMaxScaler technique on the categorical data sets
from sklearn.preprocessing import MinMaxScaler
```

In [143…

```
mn = MinMaxScaler()
df_cat_train_1 = mn.fit_transform(df_cat_train)
df_cat_test_1 = mn.fit_transform(df_cat_test)
df_cat_train_sc = pd.DataFrame(df_cat_train_1, index = df_cat_train.index, columns =
df_cat_test_sc = pd.DataFrame(df_cat_test_1, index = df_cat_test.index, columns = df
```

In [144…

```
## All the categorical data is chnaged into the numerical data
# Now concat the numerical and categorical data set of the training and testing data
df_final_train = pd.concat([df_num_train, df_cat_train_sc], axis = 1)
df_final_test = pd.concat([df_num_test, df_cat_test_sc], axis = 1)
```

In [145…

```
df_final_train.head()
```

Out[145…

	X10	X12	X13	X14	X15	X16	X17	X18	X19	X20	...	X384	X385	X0	X1	
0	0	0	1	0	0	0	0	1	0	0	...	0	0	0.695652	0.884615	0.39
1	0	0	0	0	0	0	0	1	0	0	...	0	0	0.695652	0.807692	0.44
2	0	0	0	0	0	0	1	0	0	0	...	0	0	0.434783	0.923077	0.79
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0.434783	0.807692	0.79
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0.434783	0.884615	0.79

5 rows × 364 columns

In [146…

```
df_final_test.head()
```

Out[146…

	X10	X12	X13	X14	X15	X16	X17	X18	X19	X20	...	X384	X385	X0	X1	X
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0.4375	0.884615	0.7727
1	0	0	0	0	0	0	0	0	1	0	...	0	0	0.8750	0.115385	0.1818

```
   X10 X12 X13 X14 X15 X16 X17 X18 X19 X20 ... X384 X385     X0        X1
2    0   0   0   1   0   0   0   0   0   0 ...    0    0 0.4375  0.884615  0.3863
3    0   0   0   0   0   0   0   0   0   0 ...    0    0 0.4375  0.500000  0.7727
4    0   0   0   1   0   0   0   0   0   0 ...    0    0 0.9375  0.769231  0.3863
```

5 rows × 364 columns

In [147…

```python
## Now apply the Dimensionality Reduction via using PCA
from sklearn.decomposition import PCA
```

In [148…

```python
n_comp = 12
pca = PCA(n_components = n_comp, random_state = 420)
df_train_pca = pca.fit_transform(df_final_train)
df_test_pca = pca.fit_transform(df_final_test)
```

In [149…

```python
## Number of columns are reduced to 12 for both training as well as the testing data
# But the data is in nd array format so covert that into data frame
df_train_pca2 = pd.DataFrame(df_train_pca, columns = ["PC1", "PC2", "PC3", "PC4", "P
df_test_pca2 = pd.DataFrame(df_test_pca, columns = ["PC1", "PC2", "PC3", "PC4", "PC5
```

In [150…

```python
df_train_pca2.head()
```

Out[150…

```
        PC1       PC2       PC3       PC4       PC5       PC6       PC7       PC8       PC9
0  0.682272  2.217390  1.233625  0.885738  1.401422  0.054223  0.654832 -0.937322  0.192252 -0
1 -0.279051  1.164201 -0.764263 -0.660639  0.237863  0.066804  1.237285 -0.530337 -0.108870  0
2 -1.018083  2.979512  0.558557  2.540751 -0.926714  3.282631 -0.940264  0.557082 -0.925952 -0
3 -0.658559  2.545045 -0.425408  2.997377 -1.681632  3.134975  0.074145  0.084039 -1.072854  0
4 -0.652313  2.370739 -0.583703  3.194208 -1.999394  3.167652 -0.143351  0.229232 -1.754659 -0
```

In [151…

```python
df_test_pca2.head()
```

Out[151…

```
        PC1       PC2       PC3       PC4       PC5       PC6       PC7       PC8       PC9
0 -0.425629  0.308747 -2.524555  2.641045 -1.995091  3.513185 -0.302433  0.095607 -1.952755 -
1  3.714128  1.677736 -0.003393 -0.604436  1.448695 -0.106079 -1.490445 -1.305825 -1.211747
2 -1.204703  0.513145 -0.911797  1.237162 -0.392674  3.213377 -0.737050  0.718334 -1.504446 -
3 -0.367696  0.181923 -2.419289  2.552553 -2.111593  3.544317 -0.253927  0.027803 -1.974823 -
4 -2.860633  0.470359  0.775156 -1.612700  0.796252  0.178710 -0.133292  0.098793 -0.079083 -
```

In [152…

```python
## Now all preprocessing steps are done
# Now apply the train_test_split to seperate the training and validation data
```

```
from sklearn.model_selection import train_test_split
```

In [153…

```
## Now define the dependent and independent variables
x = df_train_pca2
y = y_train
```

In [154…

```
X_train, X_valid, Y_train, Y_valid = train_test_split(x, y, test_size = 0.2, random_
```

In [155…

```
## Training and validation data is created
# Now train the model using XGBoost algorithm
import xgboost as xgb
```

In [156…

```
d_train = xgb.DMatrix(X_train, label = Y_train)
d_valid = xgb.DMatrix(X_valid, label = Y_valid)
d_test = xgb.DMatrix(df_test_pca2)
```

In [157…

```
params = {}
params["objective"] = "reg:linear"
params["eta"] = 0.02
params["max_depth"] = 4
```

In [158…

```
w = [(d_train, "train"), (d_valid, "valid")]
```

In [159…

```
## import the r2_score metric for the evaluation of the model
from sklearn.metrics import r2_score
```

In [160…

```
## create a function to
#generate the r2_score
def score(preds, dtrain):
    labels = dtrain.get_label()
    return"r2", r2_score(labels, preds)
```

In [161…

```
## Now lets train the model
trained = xgb.train(params, d_train, 1000, w, early_stopping_rounds = 50, feval = sc
```

```
[13:05:50] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/sr
c/objective/regression_obj.cu:171: reg:linear is now deprecated in favor of reg:squa
rederror.
[0]     train-rmse:99.14659     train-r2:-58.35085      valid-rmse:98.26758     vali
d-r2:-67.64398
[10]    train-rmse:81.25641     train-r2:-38.86453      valid-rmse:80.39062     vali
d-r2:-44.94020
[20]    train-rmse:66.67461     train-r2:-25.84061      valid-rmse:65.79161     vali
d-r2:-29.76968
[30]    train-rmse:54.79623     train-r2:-17.12896      valid-rmse:53.86914     vali
d-r2:-19.62824
[40]    train-rmse:45.13720     train-r2:-11.30101      valid-rmse:44.16470     vali
d-r2:-12.86541
[50]    train-rmse:37.29795     train-r2:-7.39927       valid-rmse:36.29005     vali
d-r2:-8.36175
[60]    train-rmse:30.94680     train-r2:-4.78233       valid-rmse:29.90724     vali
d-r2:-5.35821
[70]    train-rmse:25.83114     train-r2:-3.02864       valid-rmse:24.75159     vali
d-r2:-3.35500
```

```
[80]	train-rmse:21.73255	train-r2:-1.85163	valid-rmse:20.61720	vali
d-r2:-2.02163
[90]	train-rmse:18.47799	train-r2:-1.06149	valid-rmse:17.33865	vali
d-r2:-1.13704
[100]	train-rmse:15.91879	train-r2:-0.53000	valid-rmse:14.75732	vali
d-r2:-0.54809
[110]	train-rmse:13.93206	train-r2:-0.17193	valid-rmse:12.76408	vali
d-r2:-0.15814
[120]	train-rmse:12.40632	train-r2:0.07070	valid-rmse:11.25407	vali
d-r2:0.09967
[130]	train-rmse:11.25278	train-r2:0.23548	valid-rmse:10.13224	vali
d-r2:0.27022
[140]	train-rmse:10.39089	train-r2:0.34811	valid-rmse:9.30835	vali
d-r2:0.38407
[150]	train-rmse:9.75497	train-r2:0.42546	valid-rmse:8.72695	vali
d-r2:0.45861
[160]	train-rmse:9.29393	train-r2:0.47848	valid-rmse:8.32602	vali
d-r2:0.50721
[170]	train-rmse:8.97829	train-r2:0.51330	valid-rmse:8.06254	vali
d-r2:0.53791
[180]	train-rmse:8.75464	train-r2:0.53725	valid-rmse:7.89498	vali
d-r2:0.55692
[190]	train-rmse:8.58798	train-r2:0.55470	valid-rmse:7.78532	vali
d-r2:0.56914
[200]	train-rmse:8.46478	train-r2:0.56738	valid-rmse:7.72132	vali
d-r2:0.57620
[210]	train-rmse:8.36830	train-r2:0.57719	valid-rmse:7.68090	vali
d-r2:0.58062
[220]	train-rmse:8.29351	train-r2:0.58471	valid-rmse:7.66025	vali
d-r2:0.58287
[230]	train-rmse:8.22629	train-r2:0.59142	valid-rmse:7.64971	vali
d-r2:0.58402
[240]	train-rmse:8.17027	train-r2:0.59696	valid-rmse:7.64299	vali
d-r2:0.58475
[250]	train-rmse:8.12457	train-r2:0.60146	valid-rmse:7.64451	vali
d-r2:0.58459
[260]	train-rmse:8.08198	train-r2:0.60563	valid-rmse:7.65097	vali
d-r2:0.58388
[270]	train-rmse:8.05113	train-r2:0.60863	valid-rmse:7.65869	vali
d-r2:0.58304
[280]	train-rmse:8.00920	train-r2:0.61270	valid-rmse:7.66112	vali
d-r2:0.58278
[290]	train-rmse:7.96914	train-r2:0.61656	valid-rmse:7.66797	vali
d-r2:0.58203
[292]	train-rmse:7.96561	train-r2:0.61690	valid-rmse:7.66941	vali
d-r2:0.58188
```

In [162…

```
## Now predict the test values using xgboost
predict = trained.predict(d_test)
```

In [168…

```
## Now create a seperate data frame
# so that we can clearly see the results
res = pd.DataFrame()
res["ID"] = df_id
res["y"] = predict
res.head()
```

Out[168…

	ID	y
0	0	78.084824
1	6	92.252342
2	7	81.842590
3	9	78.531166

	ID	y
4	13	108.980766

In [172…

```
## Now save the result in a csv file
res.to_csv("Final_Output.csv", index = False)
```