Preparing the dataset

Importing some libraries and packages

In [547…

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedKFold
```

Importing the dataset

In [548…

```
dataset = pd.read_excel('/Users/dakshbhuva/Downloads/ENB2012_data.xlsx')
dataset.head()
```

Out[548…

	X1	X2	X3	X4	X5	X6	X7	X8	Y2	Unnamed: 9	Unnamed: 10
0	0.98	514.5	294.0	110.25	7.0	2	0.0	0	21.33	NaN	NaN
1	0.98	514.5	294.0	110.25	7.0	3	0.0	0	21.33	NaN	NaN
2	0.98	514.5	294.0	110.25	7.0	4	0.0	0	21.33	NaN	NaN
3	0.98	514.5	294.0	110.25	7.0	5	0.0	0	21.33	NaN	NaN
4	0.90	563.5	318.5	122.50	7.0	2	0.0	0	28.28	NaN	NaN

Extracting the required columns

In [549…

```
cols_to_use = ['X1','X2','X3','X4','X5','X6','X7','X8','Y2',]
dataset = dataset[cols_to_use]
dataset.head()
```

Out[549…

	X1	X2	X3	X4	X5	X6	X7	X8	Y2
0	0.98	514.5	294.0	110.25	7.0	2	0.0	0	21.33
1	0.98	514.5	294.0	110.25	7.0	3	0.0	0	21.33
2	0.98	514.5	294.0	110.25	7.0	4	0.0	0	21.33
3	0.98	514.5	294.0	110.25	7.0	5	0.0	0	21.33
4	0.90	563.5	318.5	122.50	7.0	2	0.0	0	28.28

In [550…

```
dataset.shape
```

Out[550… (768, 9)

In [551…

```
dvalues = dataset.values
X, y = dvalues[:, :-1], dvalues[:, -1]
print(X)
print(y)
```

```
[[9.800e-01 5.145e+02 2.940e+02 ... 2.000e+00 0.000e+00 0.000e+00]
 [9.800e-01 5.145e+02 2.940e+02 ... 3.000e+00 0.000e+00 0.000e+00]
 [9.800e-01 5.145e+02 2.940e+02 ... 4.000e+00 0.000e+00 0.000e+00]
 ...
 [6.200e-01 8.085e+02 3.675e+02 ... 3.000e+00 4.000e-01 5.000e+00]
 [6.200e-01 8.085e+02 3.675e+02 ... 4.000e+00 4.000e-01 5.000e+00]
 [6.200e-01 8.085e+02 3.675e+02 ... 5.000e+00 4.000e-01 5.000e+00]]
[21.33 21.33 21.33 21.33 28.28 25.38 25.16 29.6  27.3  21.97 23.49 27.87
 23.77 21.46 21.16 24.93 37.73 31.27 30.93 39.44 29.79 29.68 29.79 29.4
 10.9  11.19 10.94 11.17 11.27 11.72 11.29 11.67 11.74 12.05 11.73 11.93
 12.4  12.23 12.4  12.14 16.78 16.8  16.75 16.67 12.07 12.22 12.08 12.04
 26.47 26.37 26.44 26.29 32.92 29.87 29.58 34.33 30.89 25.6  27.03 31.73
 27.31 24.91 24.61 28.51 41.68 35.28 34.43 43.33 33.87 34.07 34.14 33.67
 13.43 13.71 13.48 13.7  13.8  14.28 13.87 14.27 14.28 14.61 14.3  14.45
 13.9  13.72 13.88 13.65 19.37 19.43 19.34 19.32 14.34 14.5  14.33 14.27
 25.95 25.63 26.13 25.89 32.54 29.44 29.36 34.2  30.91 25.63 27.36 31.9
 27.38 25.02 24.8  28.79 41.07 34.62 33.87 42.86 33.91 34.07 34.17 33.78
 13.39 13.72 13.57 13.79 13.67 14.11 13.8  14.21 13.2  13.54 13.32 13.51
 14.86 14.75 15.   14.74 19.23 19.34 19.32 19.3  14.37 14.57 14.27 14.24
 25.68 26.02 25.84 26.14 34.14 32.85 30.08 29.67 31.73 31.01 25.9  27.4
 28.68 27.54 25.35 24.93 43.12 41.22 35.1  34.29 33.85 34.11 34.48 34.5
 13.6  13.36 13.65 13.49 14.14 13.77 14.3  13.87 14.44 14.27 14.67 14.4
 13.46 13.7  13.59 13.83 19.14 19.18 19.37 19.29 14.09 14.23 14.14 13.89
 25.91 25.72 26.18 25.87 29.34 33.91 32.83 29.92 27.17 31.76 31.06 25.81
 24.61 28.61 27.57 25.16 34.25 43.3  41.86 35.29 34.11 33.62 33.89 34.05
 13.2  13.36 13.21 13.53 13.67 14.12 13.79 14.2  14.29 14.49 14.42 14.73
 14.86 14.67 15.   14.83 19.24 19.25 19.42 19.48 14.37 14.34 14.28 14.47
 25.64 25.98 25.88 26.18 29.82 29.52 34.45 33.01 25.82 27.33 32.04 31.28
 25.11 24.77 28.88 27.69 34.99 34.18 43.14 41.26 34.25 34.35 33.64 33.88
 13.65 13.44 13.72 13.5  14.18 13.75 14.26 13.89 14.55 14.28 14.46 14.39
 14.54 14.81 14.65 14.87 19.24 19.18 19.26 19.29 14.24 13.97 13.99 14.15
 29.79 29.79 29.28 29.49 36.12 33.17 32.71 37.58 33.98 28.61 30.12 34.73
 30.17 27.84 27.25 31.39 43.8  37.81 36.85 45.52 36.85 37.58 37.45 36.62
 15.19 15.5  15.28 15.5  15.42 15.85 15.44 15.81 15.21 15.63 15.48 15.78
 16.39 16.27 16.39 16.19 21.13 21.19 21.09 21.08 15.77 15.95 15.77 15.76
 29.62 29.69 30.18 30.02 35.56 32.64 32.77 37.72 33.37 27.89 29.9  34.52
 28.27 26.96 26.72 29.88 43.86 37.41 36.77 45.97 36.87 37.35 37.28 36.81
 14.73 15.1  15.18 15.44 14.91 15.4  14.94 15.32 15.52 15.85 15.66 15.99
 15.89 15.85 16.22 15.87 20.47 20.56 20.48 20.43 15.32 15.64 15.14 15.3
 29.43 29.78 30.1  30.19 36.35 35.1  32.83 32.46 33.52 32.93 28.38 29.82
 28.77 27.76 26.95 26.41 45.13 43.66 37.76 36.87 36.07 36.44 37.28 37.29
 14.49 13.79 14.72 14.76 14.92 14.74 15.57 14.94 14.92 14.38 15.44 15.17
 15.53 15.8  16.14 16.26 19.87 20.03 20.46 20.28 14.89 14.96 14.89 14.35
 29.61 29.59 30.19 30.12 32.12 37.12 36.16 33.16 29.45 34.19 33.93 28.31
 26.3  29.43 28.76 27.34 36.26 45.48 44.16 37.26 37.2  36.76 37.05 37.51
 14.92 15.24 15.03 15.35 14.67 15.09 15.2  15.64 15.37 15.73 15.83 16.13
 15.95 15.59 16.17 16.14 19.65 19.76 20.37 19.9  15.41 15.56 15.07 15.38
 29.53 29.77 30.   30.2  32.25 32.   37.19 35.62 28.02 29.43 34.15 33.47
```

```
26.53 26.08 29.31 28.14 37.54 36.66 45.28 43.73 36.93 37.01 35.73 36.15
14.48 14.58 14.81 14.03 15.27 14.71 15.23 14.97 15.14 14.97 15.22 14.6
15.83 16.03 15.8  16.06 20.13 20.01 20.19 20.29 15.19 14.61 14.61 14.75
33.37 33.34 32.83 33.04 39.28 36.38 35.92 40.99 35.99 30.66 31.7  36.73
31.71 29.13 28.99 33.54 45.29 39.07 38.35 46.94 39.55 40.85 40.63 39.48
16.94 17.25 17.03 17.25 17.1  17.51 17.12 17.47 16.5  17.   16.87 17.2
18.14 18.03 18.14 17.95 22.72 22.73 22.72 22.53 17.2  17.21 17.15 17.2
32.96 33.13 33.94 33.78 38.35 35.39 34.94 40.66 35.48 30.53 32.28 36.86
30.34 27.93 28.95 32.92 45.59 39.41 38.84 48.03 39.48 40.4  40.47 39.7
16.43 16.93 16.99 17.03 16.77 17.37 17.27 17.51 16.44 17.01 17.23 17.22
17.85 17.89 18.36 18.15 21.72 22.07 22.09 21.93 17.36 17.38 16.86 16.99
32.78 33.24 33.86 34.   37.26 35.04 33.82 33.31 35.22 34.7  30.11 31.6
32.43 30.65 29.77 29.64 46.44 44.18 38.81 38.23 38.17 38.48 39.66 40.1
16.08 15.39 16.57 16.6  16.11 15.47 16.7  16.1  16.35 15.84 16.99 17.02
17.04 17.63 18.1  18.22 20.78 20.72 21.54 21.53 16.9  17.14 16.56 16.
32.95 33.06 33.95 33.88 33.98 39.92 39.22 36.1  31.53 36.2  36.21 31.
28.2  32.35 31.14 28.43 38.33 47.59 46.23 39.56 40.36 39.67 39.85 40.77
16.61 16.74 16.9  17.32 16.85 17.2  17.23 17.74 16.81 16.88 16.9  17.39
17.86 17.82 18.36 18.24 21.68 21.54 22.25 22.49 17.1  16.79 16.58 16.79
32.88 33.23 33.76 34.01 33.94 33.14 38.79 37.27 29.69 31.2  36.26 35.71
29.93 29.56 33.84 32.54 38.56 37.7  47.01 44.87 39.37 39.8  37.79 38.18
16.69 16.62 16.94 16.7  15.59 14.58 15.33 15.31 16.63 15.87 16.54 16.74
17.64 17.79 17.55 18.06 20.82 20.21 20.71 21.4  16.88 17.11 16.61 16.03]
```

Spliting the dataset into Training Data and Testing Data

In [552…

```python
from sklearn.model_selection import train_test_split
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size = 0.3, ra
```

Linear Regression

In [553…

```python
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(train_X, train_y)
```

Out[553… `LinearRegression()`

In [554…

```python
pred_y = reg.predict(test_X)
```

Accuracy score and mean squared error (MSE)

In [555…

```python
reg.score(train_X, train_y)
```

Out[555… `0.8960068984743137`

In [556…

```
reg.score(test_X, test_y)
```

Out[556… 0.8679934073082682

In [557…

```
from sklearn.metrics import mean_squared_error

print("MSE", mean_squared_error(test_y, pred_y))
```

MSE 11.932799400373

5-fold cross validation based on mean squared error (MSE)

In [558…

```
cv = RepeatedKFold(n_splits=5, n_repeats=1, random_state=1)
scores = cross_val_score(reg, X, y, scoring='neg_mean_squared_error', cv=cv,
```

In [559…

```
scores = np.absolute(scores)
print(scores)
print('Mean MSE: %.3f' % (np.mean(scores)))
```

```
[13.33425648  9.25534554 11.44645645  8.08679584  9.12829731]
Mean MSE: 10.250
```

Lasso Regression

In [560…

```
from sklearn.linear_model import Lasso


lasso_reg = Lasso(alpha=0.1, max_iter=100, tol=0.1)
lasso_reg.fit(train_X, train_y)
```

Out[560… Lasso(alpha=0.1, max_iter=100, tol=0.1)

In [561…

```
pred_lasso_y = lasso_reg.predict(test_X)
```

Accuracy score and mean squared error (MSE)

In [562…

```
lasso_reg.score(train_X, train_y)
```

Out[562… 0.8496597478733995

In [563…

```
lasso_reg.score(test_X, test_y)
```

Out[563… 0.8210153461763534

In [564…

```
print("MSE", mean_squared_error(test_y, pred_lasso_y))
```

```
MSE 16.17940381818941
```

5-fold cross validation based on mean squared error (MSE)

In [565…

```
cv = RepeatedKFold(n_splits=5, n_repeats=1, random_state=1)
scores = cross_val_score(lasso_reg, X, y, scoring='neg_mean_squared_error', c
```

In [566…

```
scores = np.absolute(scores)
print(scores)
print('Mean MSE: %.3f' % (np.mean(scores)))
```

```
[17.5974518  12.90823341 17.22102097 12.83917627 12.00915707]
Mean MSE: 14.515
```

Ridge Regression

In [567…

```
from sklearn.linear_model import Ridge

ridge_reg = Ridge(alpha=0.1, max_iter=100, tol=0.1)
ridge_reg.fit(train_X, train_y)
```

Out[567… Ridge(alpha=0.1, max_iter=100, tol=0.1)

In [568…

```
pred_ridge_y = ridge_reg.predict(test_X)
```

Accuracy score and mean squared error (MSE)

In [569…

```
ridge_reg.score(train_X, train_y)
```

Out[569… 0.8936293256602481

In [570…

```
ridge_reg.score(test_X, test_y)
```

Out[570… 0.8651293361344408

In [571…
```
print("MSE", mean_squared_error(test_y, pred_ridge_y))
```

```
MSE 12.191698490856195
```

5-fold cross validation based on mean squared error (MSE)

In [572…
```
cv = RepeatedKFold(n_splits=5, n_repeats=1, random_state=1)
scores = cross_val_score(ridge_reg, X, y, scoring='neg_mean_squared_error', c
```

In [573…
```
scores = np.absolute(scores)
print(scores)
print('Mean MSE: %.3f' % (np.mean(scores)))
```

```
[13.51010438  9.40918598 11.72885574  8.3552932   9.32791736]
Mean MSE: 10.466
```

Elastic Net Regression

In [574…
```
from sklearn.linear_model import ElasticNet

elastic_reg = ElasticNet(alpha=0.1, max_iter=100, tol=0.1)
elastic_reg.fit(train_X, train_y)
```

Out[574… ElasticNet(alpha=0.1, max_iter=100, tol=0.1)

In [575…
```
pred_elastic_y = elastic_reg.predict(test_X)
```

Accuracy score and mean squared error (MSE)

In [576…
```
elastic_reg.score(train_X, train_y)
```

Out[576… 0.8170593174850932

In [577…
```
elastic_reg.score(test_X, test_y)
```

Out[577… 0.7897475170056485

In [578…

```
print("MSE", mean_squared_error(test_y, pred_elastic_y))
```

```
MSE 19.005874266150027
```

5-fold cross validation based on mean squared error (MSE)

In [579…

```
cv = RepeatedKFold(n_splits=5, n_repeats=1, random_state=1)
scores = cross_val_score(elastic_reg, X, y, scoring='neg_mean_squared_error',
```

In [580…

```
scores = np.absolute(scores)
print(scores)
print('Mean MSE: %.3f' % (np.mean(scores)))
```

```
[20.36707312 15.40995067 20.44575913 14.71116777 13.54677376]
Mean MSE: 16.896
```