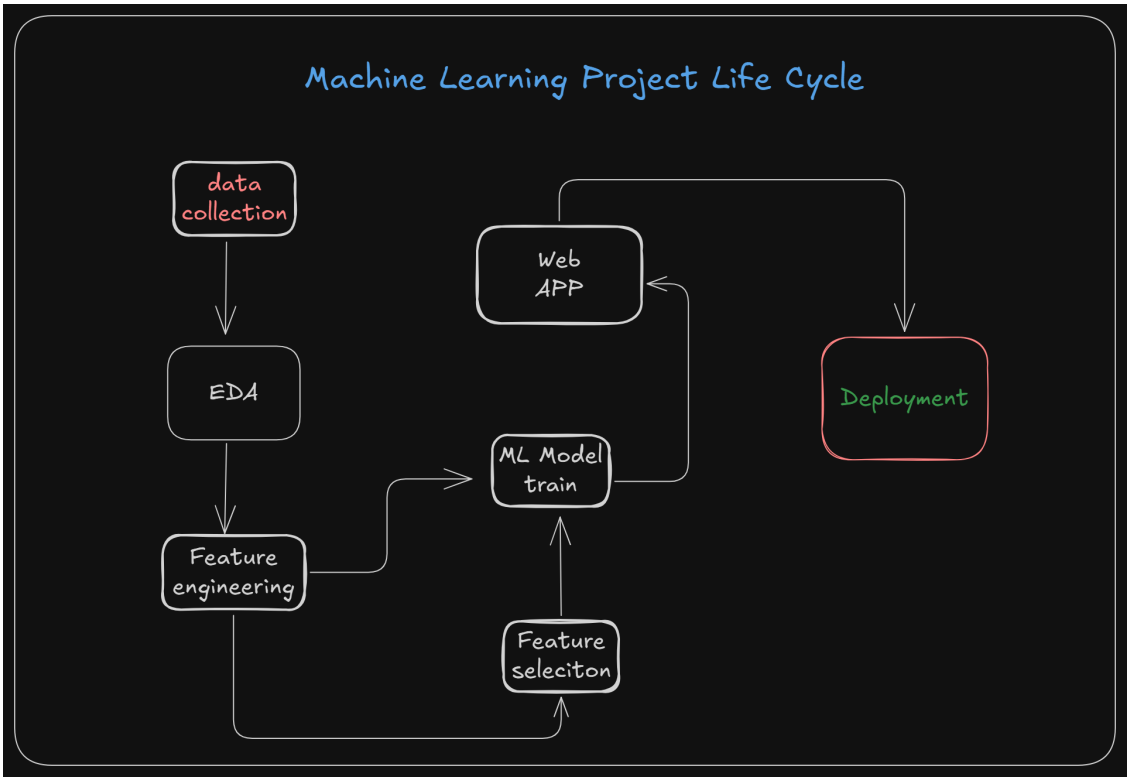
End-to-End-ML_Project

July 30, 2025

1 End to End Machine Learning Project Using Regression Models and Deployment



[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

[ ]:
```
df = pd.read_csv('../Dataset/Algerian_forest_fires_dataset_UPDATE.csv',␣
↪header=1)
```

[3]:
```
df.head()
```

[3]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes
0	01	06	2012	29	57	18	0	65.7	3.4	7.6	1.3	3.4	0.5	not fire
1	02	06	2012	29	61	13	1.3	64.4	4.1	7.6	1	3.9	0.4	not fire
2	03	06	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	not fire
3	04	06	2012	25	89	13	2.5	28.6	1.3	6.9	0	1.7	0	not fire
4	05	06	2012	27	77	16	0	64.8	3	14.2	1.2	3.9	0.5	not fire

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 246 entries, 0 to 245
Data columns (total 14 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   day          246 non-null    object
 1   month        245 non-null    object
 2   year         245 non-null    object
 3   Temperature  245 non-null    object
 4    RH          245 non-null    object
 5    Ws          245 non-null    object
 6   Rain         245 non-null    object
 7   FFMC         245 non-null    object
 8   DMC          245 non-null    object
 9   DC           245 non-null    object
 10  ISI          245 non-null    object
 11  BUI          245 non-null    object
 12  FWI          245 non-null    object
 13  Classes      244 non-null    object
dtypes: object(14)
memory usage: 27.0+ KB
```

1.0.1 Data Cleaning

[5]:
```
## Missing values
df[df.isnull().any(axis=1)]
```

[5]:
```
                                day month  year Temperature   RH   Ws Rain   \
122  Sidi-Bel Abbes Region Dataset   NaN   NaN         NaN  NaN  NaN   NaN
167                               14    07  2012          37   37   18   0.2

     FFMC   DMC      DC   ISI   BUI      FWI Classes
122   NaN   NaN     NaN   NaN   NaN      NaN       NaN
167  88.9  12.9  14.6 9  12.5  10.4  fire          NaN
```

[6]:
```
df.loc[:122, "Region"]=0
df.loc[122:, "Region"]=1

## This is creating a label from 1 to 122 as 0 and 122 onwards as 1
```

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 246 entries, 0 to 245
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   day          246 non-null    object
 1   month        245 non-null    object
 2   year         245 non-null    object
 3   Temperature  245 non-null    object
 4    RH          245 non-null    object
 5    Ws          245 non-null    object
 6   Rain         245 non-null    object
 7   FFMC         245 non-null    object
 8   DMC          245 non-null    object
 9   DC           245 non-null    object
 10  ISI          245 non-null    object
 11  BUI          245 non-null    object
 12  FWI          245 non-null    object
 13  Classes      244 non-null    object
 14  Region       246 non-null    float64
dtypes: float64(1), object(14)
memory usage: 29.0+ KB
```

[8]:
```
df.head()
```

[8]:
```
   day month  year Temperature  RH  Ws Rain   FFMC  DMC    DC  ISI  BUI  FWI  \
0  01    06  2012          29  57  18     0  65.7  3.4   7.6  1.3  3.4  0.5
1  02    06  2012          29  61  13   1.3  64.4  4.1   7.6    1  3.9  0.4
```

```
2  03    06  2012          26  82  22  13.1  47.1  2.5   7.1  0.3  2.7  0.1
3  04    06  2012          25  89  13   2.5  28.6  1.3   6.9    0  1.7    0
4  05    06  2012          27  77  16     0  64.8    3  14.2  1.2  3.9  0.5

     Classes    Region
0  not fire        0.0
1  not fire        0.0
2  not fire        0.0
3  not fire        0.0
4  not fire        0.0
```

[9]:
```
df = df.dropna().reset_index(drop=True)
```

[10]:
```
df.head()
```

[10]:
```
  day month  year Temperature  RH  Ws Rain   FFMC  DMC    DC  ISI  BUI  FWI  \
0  01    06  2012          29  57  18     0  65.7  3.4   7.6  1.3  3.4  0.5
1  02    06  2012          29  61  13   1.3  64.4  4.1   7.6    1  3.9  0.4
2  03    06  2012          26  82  22  13.1  47.1  2.5   7.1  0.3  2.7  0.1
3  04    06  2012          25  89  13   2.5  28.6  1.3   6.9    0  1.7    0
4  05    06  2012          27  77  16     0  64.8    3  14.2  1.2  3.9  0.5

     Classes    Region
0  not fire        0.0
1  not fire        0.0
2  not fire        0.0
3  not fire        0.0
4  not fire        0.0
```

[11]:
```
df.isnull().sum()
```

[11]:
```
day            0
month          0
year           0
Temperature    0
 RH            0
 Ws            0
Rain           0
FFMC           0
DMC            0
DC             0
ISI            0
BUI            0
FWI            0
Classes        0
Region         0
dtype: int64
```

[12]:
```
df.iloc[[122]]
```

[12]:
```
     day  month  year  Temperature   RH   Ws  Rain   FFMC  DMC  DC  ISI  BUI  \
122  day  month  year  Temperature   RH   Ws  Rain   FFMC  DMC  DC  ISI  BUI

     FWI  Classes    Region
122  FWI  Classes       1.0
```

[13]:
```
## Remove the 122th row
df = df.drop(122).reset_index(drop=True)
```

[14]:
```
df.iloc[[122]]
```

[14]:
```
    day month  year Temperature  RH  Ws Rain   FFMC  DMC   DC  ISI  BUI  FWI  \
122  01    06  2012          32  71  12   0.7  57.1  2.5  8.2  0.6  2.8  0.2

       Classes    Region
122  not fire        1.0
```

[15]:
```
df.columns
```

[15]:
```
Index(['day', 'month', 'year', 'Temperature', ' RH', ' Ws', 'Rain ', 'FFMC',
       'DMC', 'DC', 'ISI', 'BUI', 'FWI', 'Classes  ', 'Region'],
      dtype='object')
```

[16]:
```
## Fix spaces in columns names

df.columns = df.columns.str.strip()
df.columns
```

[16]:
```
Index(['day', 'month', 'year', 'Temperature', 'RH', 'Ws', 'Rain', 'FFMC',
       'DMC', 'DC', 'ISI', 'BUI', 'FWI', 'Classes', 'Region'],
      dtype='object')
```

1.0.2 Changes the required columns as integer data type

[17]:
```
df[['day', 'month', 'year', 'Temperature', 'RH', 'Ws']] = df[['day', 'month',␣
 ↪'year', 'Temperature', 'RH', 'Ws']].astype(int)
```

[18]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243 entries, 0 to 242
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   day          243 non-null    int32
 1   month        243 non-null    int32
```

```
 2   year         243 non-null    int32
 3   Temperature  243 non-null    int32
 4   RH           243 non-null    int32
 5   Ws           243 non-null    int32
 6   Rain         243 non-null    object
 7   FFMC         243 non-null    object
 8   DMC          243 non-null    object
 9   DC           243 non-null    object
 10  ISI          243 non-null    object
 11  BUI          243 non-null    object
 12  FWI          243 non-null    object
 13  Classes      243 non-null    object
 14  Region       243 non-null    float64
dtypes: float64(1), int32(6), object(8)
memory usage: 22.9+ KB
```

1.0.3 Changing the other columns to float data types

[19]:
```
object = [features for features in df.columns if df[features].dtypes=='O']
object
```

[19]:
```
['Rain', 'FFMC', 'DMC', 'DC', 'ISI', 'BUI', 'FWI', 'Classes']
```

[20]:
```
for i in object:
        if i!='Classes':
                df[i] = df[i].astype(float)
```

[21]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243 entries, 0 to 242
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   day          243 non-null    int32
 1   month        243 non-null    int32
 2   year         243 non-null    int32
 3   Temperature  243 non-null    int32
 4   RH           243 non-null    int32
 5   Ws           243 non-null    int32
 6   Rain         243 non-null    float64
 7   FFMC         243 non-null    float64
 8   DMC          243 non-null    float64
 9   DC           243 non-null    float64
 10  ISI          243 non-null    float64
 11  BUI          243 non-null    float64
 12  FWI          243 non-null    float64
 13  Classes      243 non-null    object
```

```
 14  Region       243 non-null    float64
dtypes: float64(8), int32(6), object(1)
memory usage: 22.9+ KB
```

[22]:
```
df.describe()
```

[22]:

	day	month	year	Temperature	RH	Ws
count	243.000000	243.000000	243.0	243.000000	243.000000	243.000000
mean	15.761317	7.502058	2012.0	32.152263	62.041152	15.493827
std	8.842552	1.114793	0.0	3.628039	14.828160	2.811385
min	1.000000	6.000000	2012.0	22.000000	21.000000	6.000000
25%	8.000000	7.000000	2012.0	30.000000	52.500000	14.000000
50%	16.000000	8.000000	2012.0	32.000000	63.000000	15.000000
75%	23.000000	8.000000	2012.0	35.000000	73.500000	17.000000
max	31.000000	9.000000	2012.0	42.000000	90.000000	29.000000

	Rain	FFMC	DMC	DC	ISI	BUI
count	243.000000	243.000000	243.000000	243.000000	243.000000	243.000000
mean	0.762963	77.842387	14.680658	49.430864	4.742387	16.690535
std	2.003207	14.349641	12.393040	47.665606	4.154234	14.228421
min	0.000000	28.600000	0.700000	6.900000	0.000000	1.100000
25%	0.000000	71.850000	5.800000	12.350000	1.400000	6.000000
50%	0.000000	83.300000	11.300000	33.100000	3.500000	12.400000
75%	0.500000	88.300000	20.800000	69.100000	7.250000	22.650000
max	16.800000	96.000000	65.900000	220.400000	19.000000	68.000000

	FWI	Region
count	243.000000	243.000000
mean	7.035391	0.497942
std	7.440568	0.501028
min	0.000000	0.000000
25%	0.700000	0.000000
50%	4.200000	0.000000
75%	11.450000	1.000000
max	31.100000	1.000000

1.0.4 Saving the cleaned dataset

[ ]:
```
df.to_csv('../Dataset/Algerian_forest_fires_dataset_cleaned.csv', index=False)
```

1.1 Exploratory Data Analysis

[24]:
```
df_copy = df
```

[25]:
```
df_copy.head()
```

[25]:
```
   day  month  year  Temperature  RH  Ws  Rain  FFMC  DMC    DC  ISI  BUI  \
0    1      6  2012           29  57  18   0.0  65.7  3.4   7.6  1.3  3.4
1    2      6  2012           29  61  13   1.3  64.4  4.1   7.6  1.0  3.9
2    3      6  2012           26  82  22  13.1  47.1  2.5   7.1  0.3  2.7
3    4      6  2012           25  89  13   2.5  28.6  1.3   6.9  0.0  1.7
4    5      6  2012           27  77  16   0.0  64.8  3.0  14.2  1.2  3.9

   FWI      Classes  Region
0  0.5  not fire        0.0
1  0.4  not fire        0.0
2  0.1  not fire        0.0
3  0.0  not fire        0.0
4  0.5  not fire        0.0
```

[26]:
```
df_copy = df.drop(['day', 'month', 'year'], axis=1)
df_copy.head()
```

[26]:
```
   Temperature  RH  Ws  Rain  FFMC  DMC    DC  ISI  BUI  FWI      Classes  \
0           29  57  18   0.0  65.7  3.4   7.6  1.3  3.4  0.5  not fire
1           29  61  13   1.3  64.4  4.1   7.6  1.0  3.9  0.4  not fire
2           26  82  22  13.1  47.1  2.5   7.1  0.3  2.7  0.1  not fire
3           25  89  13   2.5  28.6  1.3   6.9  0.0  1.7  0.0  not fire
4           27  77  16   0.0  64.8  3.0  14.2  1.2  3.9  0.5  not fire

   Region
0     0.0
1     0.0
2     0.0
3     0.0
4     0.0
```

[27]:
```
df_copy['Classes'].value_counts()
```

[27]:
```
Classes
fire            131
not fire        101
fire              4
fire              2
not fire          2
not fire          1
not fire          1
not fire          1
Name: count, dtype: int64
```

[28]:
```
### Encoding the categories in classes
```

```
df_copy['Classes'] = np.where(df_copy['Classes'].str.contains('not fire'), 0,␣
↪1) ## here 1 denotes the else part
```

[29]:
```
df_copy.head()
```

[29]:

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	Region
0	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	0.5	0	0.0
1	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	0.4	0	0.0
2	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	0	0.0
3	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	0.0	0	0.0
4	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0.5	0	0.0

[30]:
```
df_copy['Classes'].value_counts()
```

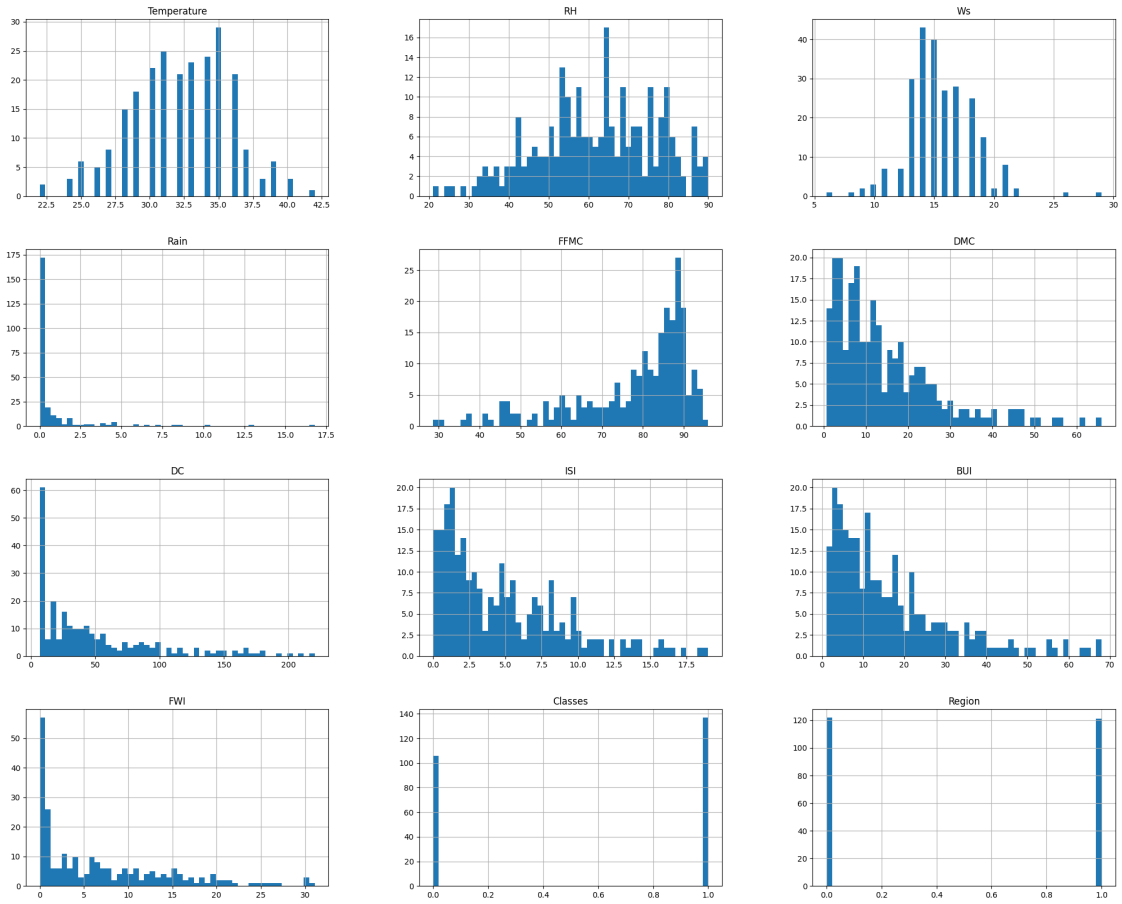
[30]:
```
Classes
1    137
0    106
Name: count, dtype: int64
```

1.1.1 Visualization of dataset

[31]:
```
print(plt.style.available)
## these are the styles are available for the use
```

```
['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-
nogrid', 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight',
'ggplot', 'grayscale', 'petroff10', 'seaborn-v0_8', 'seaborn-v0_8-bright',
'seaborn-v0_8-colorblind', 'seaborn-v0_8-dark', 'seaborn-v0_8-dark-palette',
'seaborn-v0_8-darkgrid', 'seaborn-v0_8-deep', 'seaborn-v0_8-muted',
'seaborn-v0_8-notebook', 'seaborn-v0_8-paper', 'seaborn-v0_8-pastel',
'seaborn-v0_8-poster', 'seaborn-v0_8-talk', 'seaborn-v0_8-ticks',
'seaborn-v0_8-white', 'seaborn-v0_8-whitegrid', 'tableau-colorblind10']
```

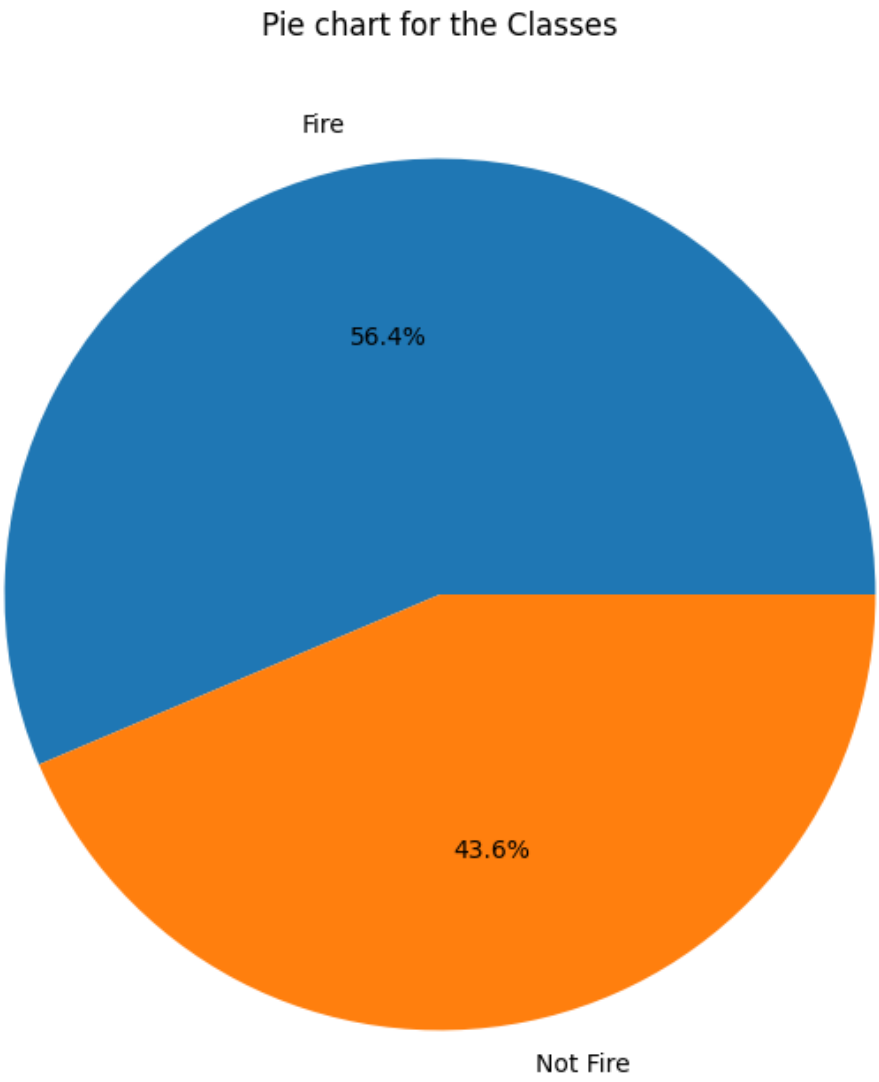
[32]:
```
plt.style.use('default')
df_copy.hist(bins=50, figsize=(25,20))
plt.show()
```



[33]:
```
## Percentage for pie chart
percentage = df_copy['Classes'].value_counts(normalize=True)*100 ## normalize␣
 ↪gives you the percentage of the specific data in my whole dataset
percentage
```

[33]:
```
Classes
1    56.378601
0    43.621399
Name: proportion, dtype: float64
```

[34]:
```
## Pie Chart of the data

classlabels = ['Fire', 'Not Fire']
plt.figure(figsize=(10,8))
plt.pie(percentage, labels=classlabels, autopct='%1.1f%%')
plt.title('Pie chart for the Classes')
plt.show()
```



1.1.2 Correlations between the dataset

[35]:
```
df_copy.corr()
```

[35]:

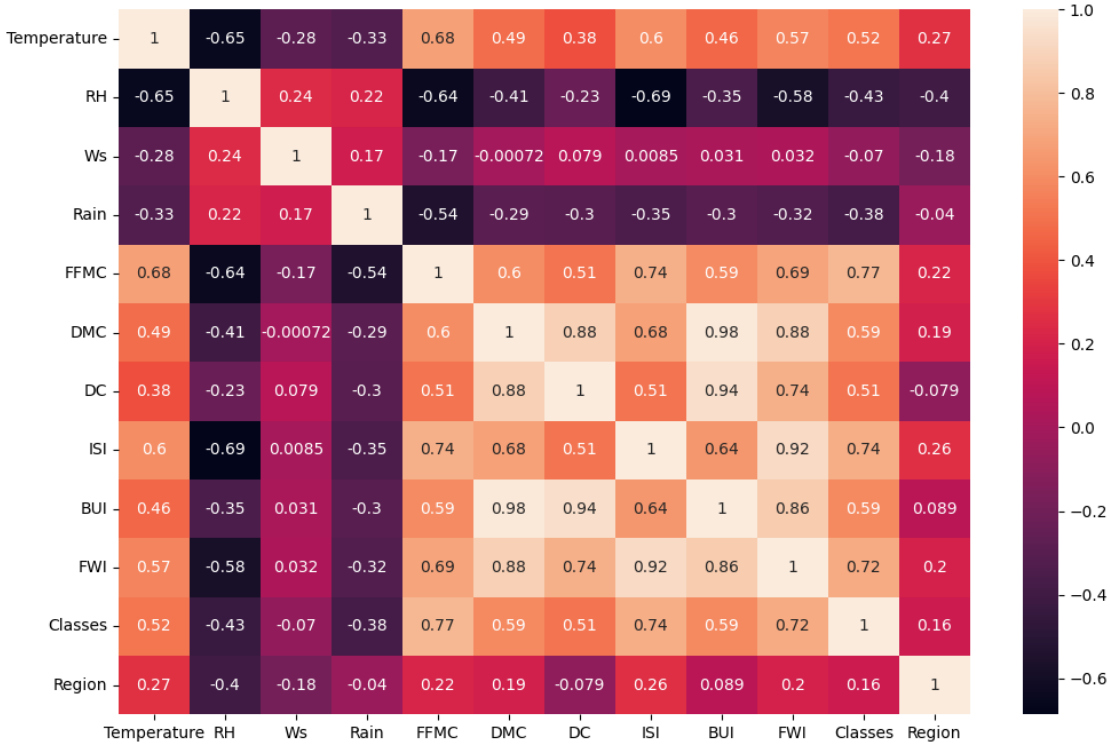
	Temperature	RH	Ws	Rain	FFMC	DMC	\
Temperature	1.000000	-0.651400	-0.284510	-0.326492	0.676568	0.485687	
RH	-0.651400	1.000000	0.244048	0.222356	-0.644873	-0.408519	
Ws	-0.284510	0.244048	1.000000	0.171506	-0.166548	-0.000721	
Rain	-0.326492	0.222356	0.171506	1.000000	-0.543906	-0.288773	
FFMC	0.676568	-0.644873	-0.166548	-0.543906	1.000000	0.603608	
DMC	0.485687	-0.408519	-0.000721	-0.288773	0.603608	1.000000	
DC	0.376284	-0.226941	0.079135	-0.298023	0.507397	0.875925	

```
ISI             0.603871 -0.686667  0.008532 -0.347484  0.740007  0.680454
BUI             0.459789 -0.353841  0.031438 -0.299852  0.592011  0.982248
FWI             0.566670 -0.580957  0.032368 -0.324422  0.691132  0.875864
Classes         0.516015 -0.432161 -0.069964 -0.379097  0.769492  0.585658
Region          0.269555 -0.402682 -0.181160 -0.040013  0.222241  0.192089
```

```
                   DC       ISI       BUI       FWI   Classes    Region
Temperature  0.376284  0.603871  0.459789  0.566670  0.516015  0.269555
RH          -0.226941 -0.686667 -0.353841 -0.580957 -0.432161 -0.402682
Ws           0.079135  0.008532  0.031438  0.032368 -0.069964 -0.181160
Rain        -0.298023 -0.347484 -0.299852 -0.324422 -0.379097 -0.040013
FFMC         0.507397  0.740007  0.592011  0.691132  0.769492  0.222241
DMC          0.875925  0.680454  0.982248  0.875864  0.585658  0.192089
DC           1.000000  0.508643  0.941988  0.739521  0.511123 -0.078734
ISI          0.508643  1.000000  0.644093  0.922895  0.735197  0.263197
BUI          0.941988  0.644093  1.000000  0.857973  0.586639  0.089408
FWI          0.739521  0.922895  0.857973  1.000000  0.719216  0.197102
Classes      0.511123  0.735197  0.586639  0.719216  1.000000  0.162347
Region      -0.078734  0.263197  0.089408  0.197102  0.162347  1.000000
```

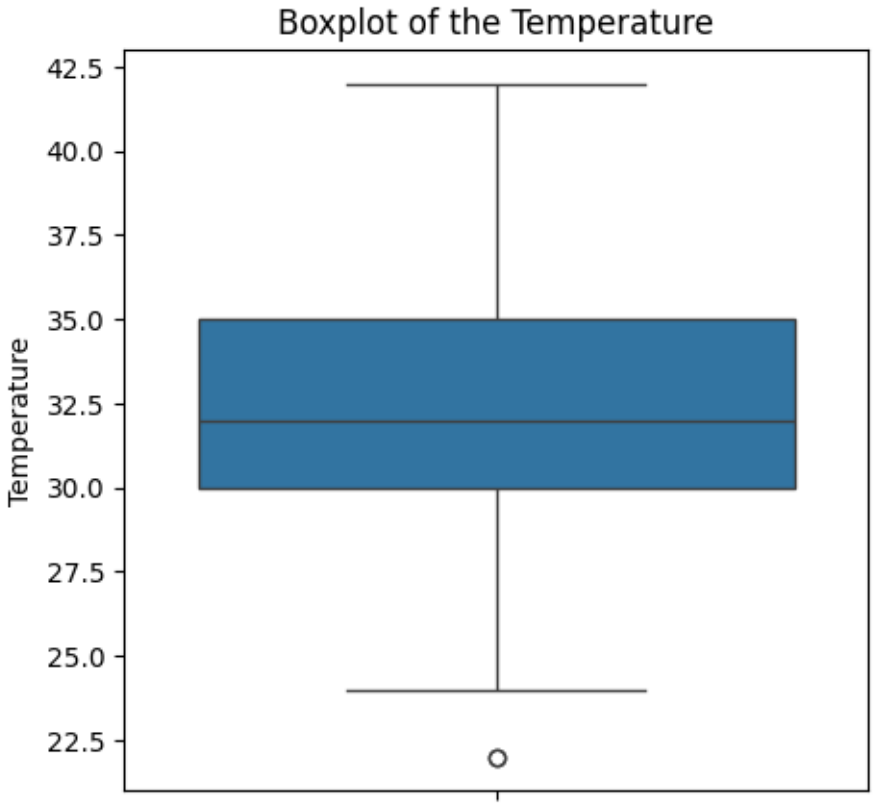
[36]:
```
plt.figure(figsize=(12,8))
sns.heatmap(df_copy.corr(), annot=True)
```

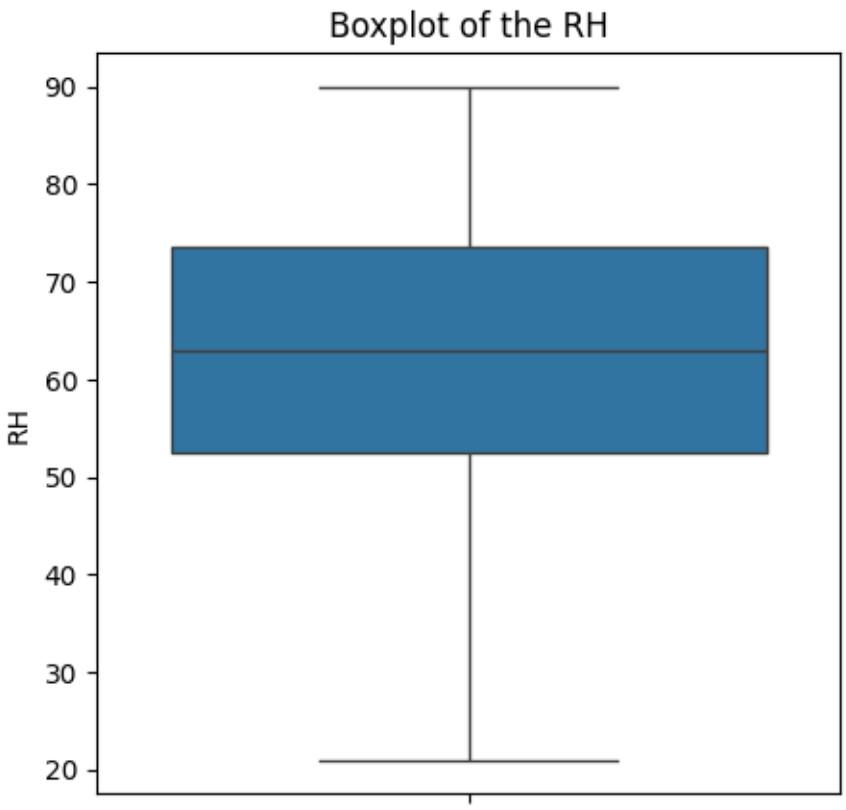
[36]: <Axes: >

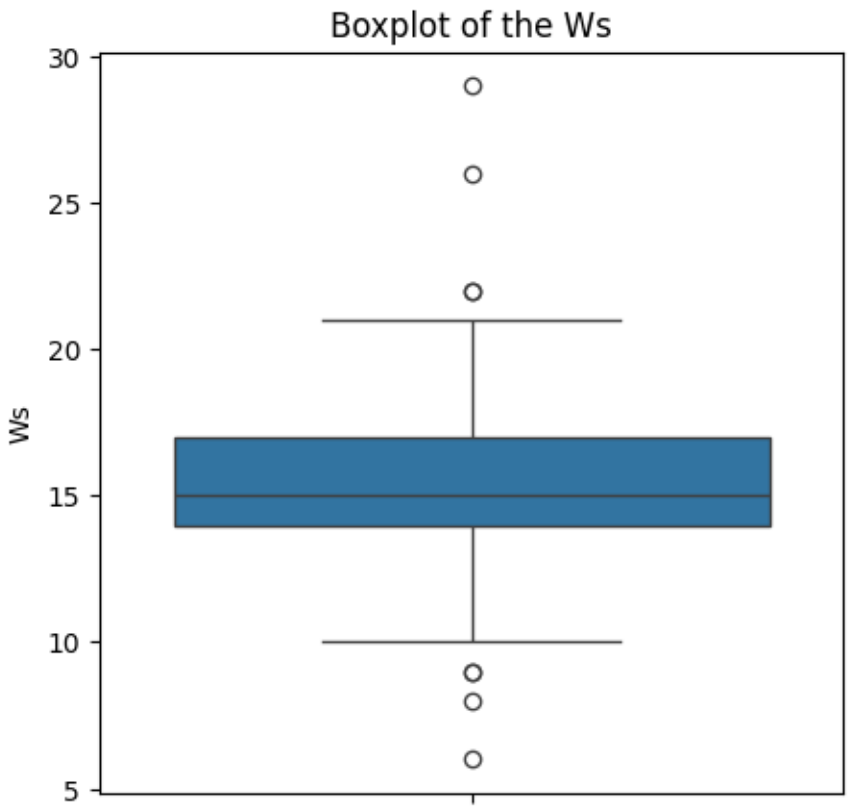


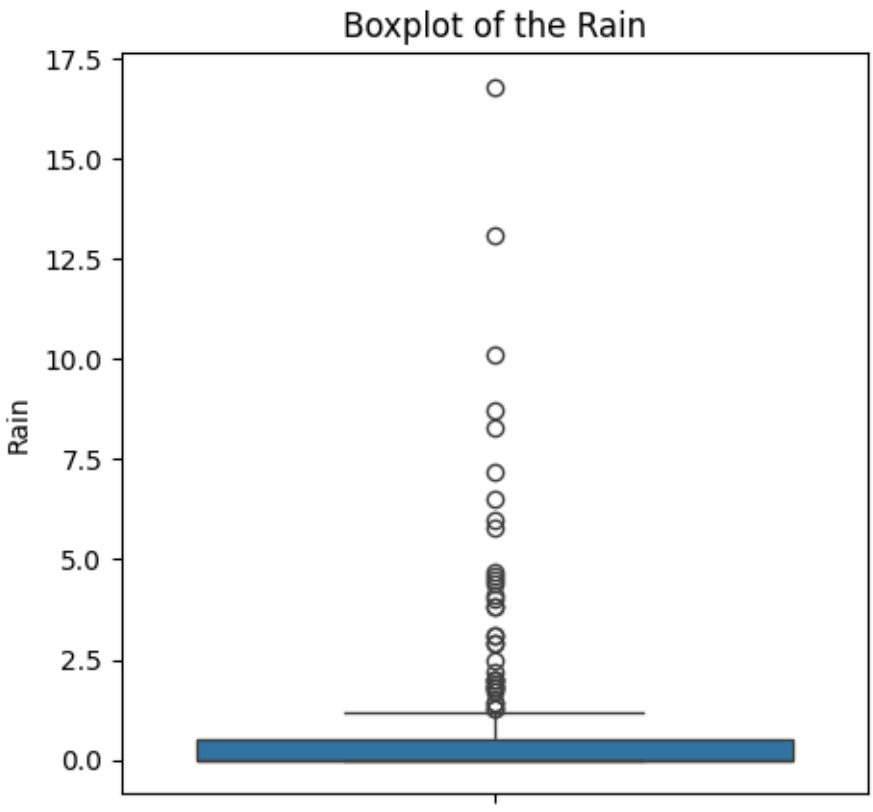
1.1.3 Analysis of the Outliers with respect to the features

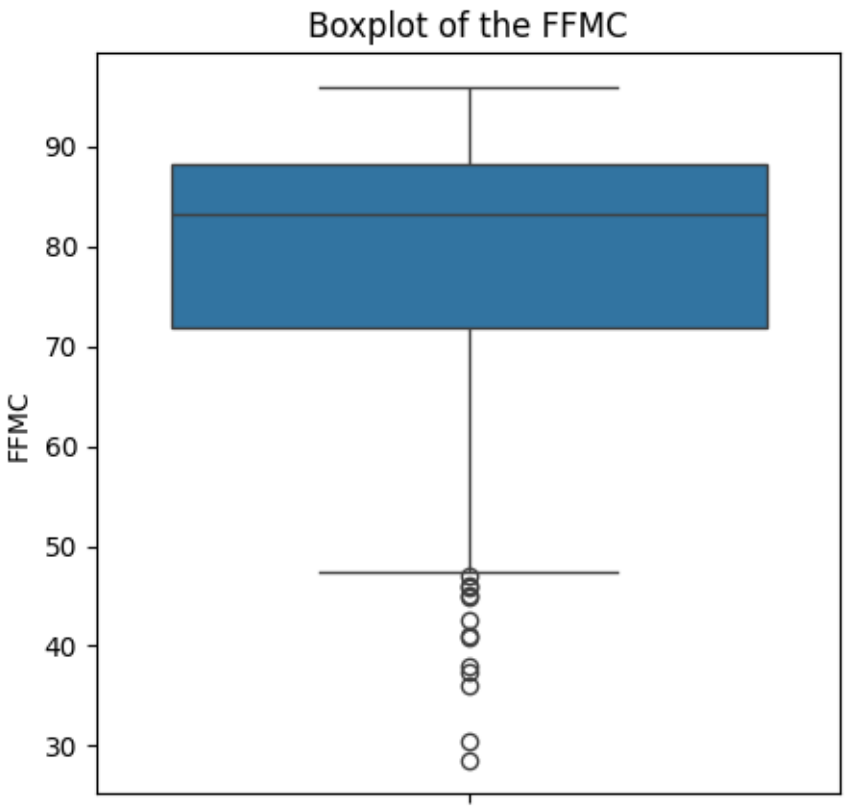
[37]:
```
for items in df_copy.columns[:-2]: # here[:-2] is not taking the last 2 columns
        plt.figure(figsize=(5,5))
        sns.boxplot(data=df_copy, y=items)
        plt.title(f"Boxplot of the {items}")
        plt.show()
```

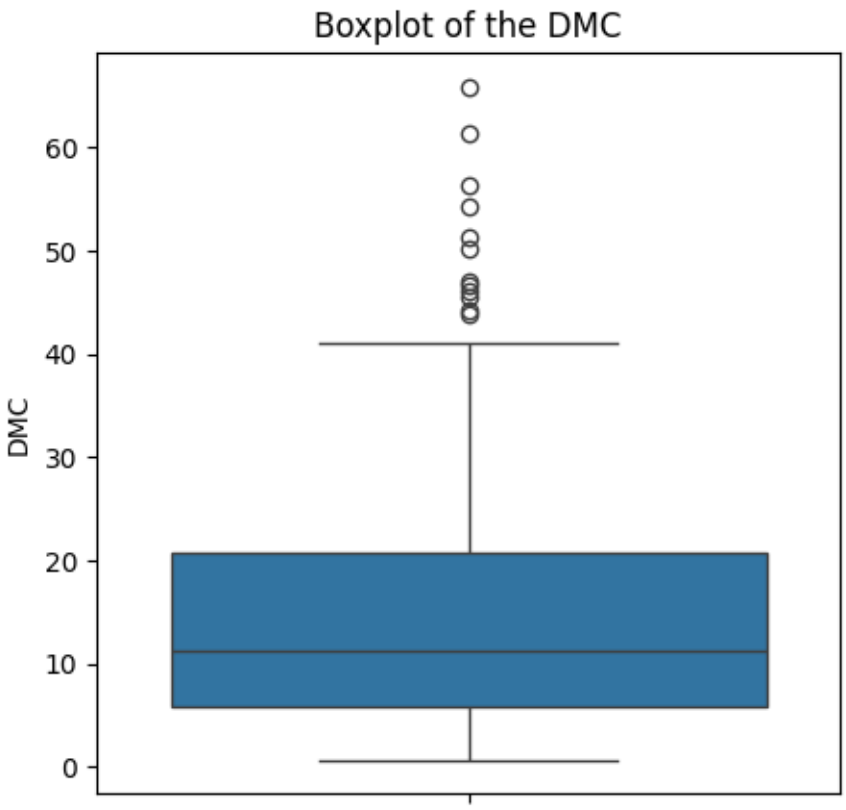


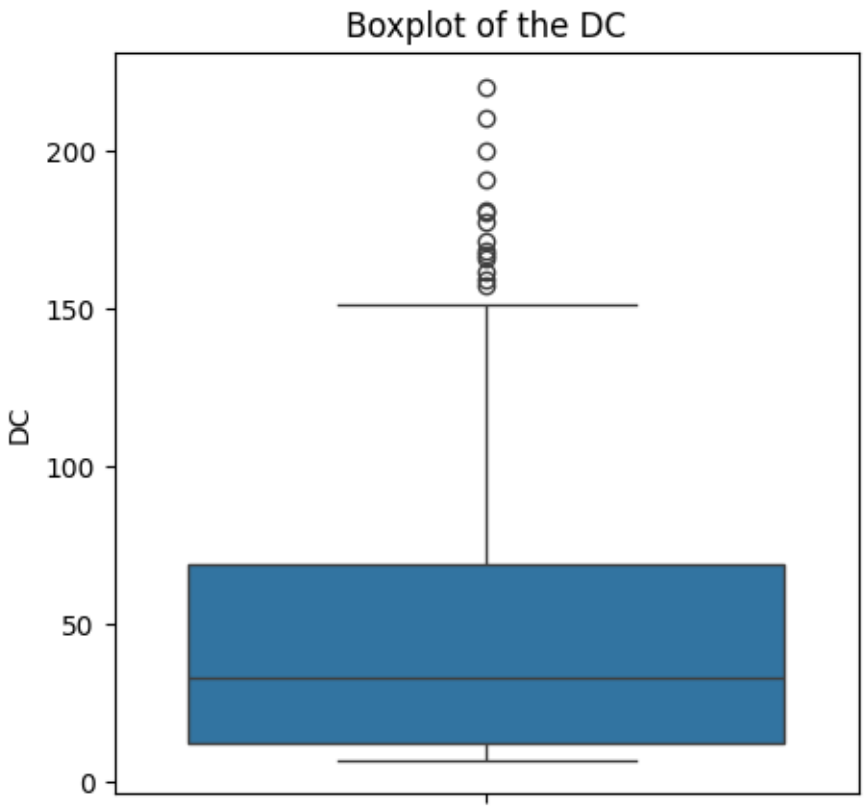

Boxplot of the RH
RH
90
80
70
60
50
40
30
20

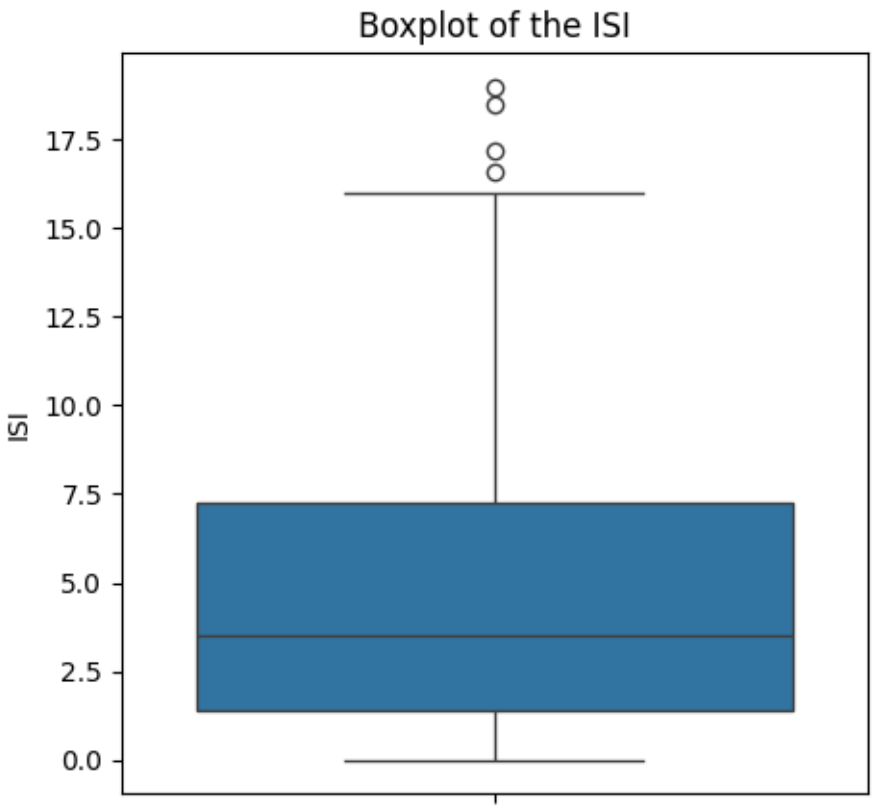

Boxplot of the Ws
Ws
30
25
20
15
10
5

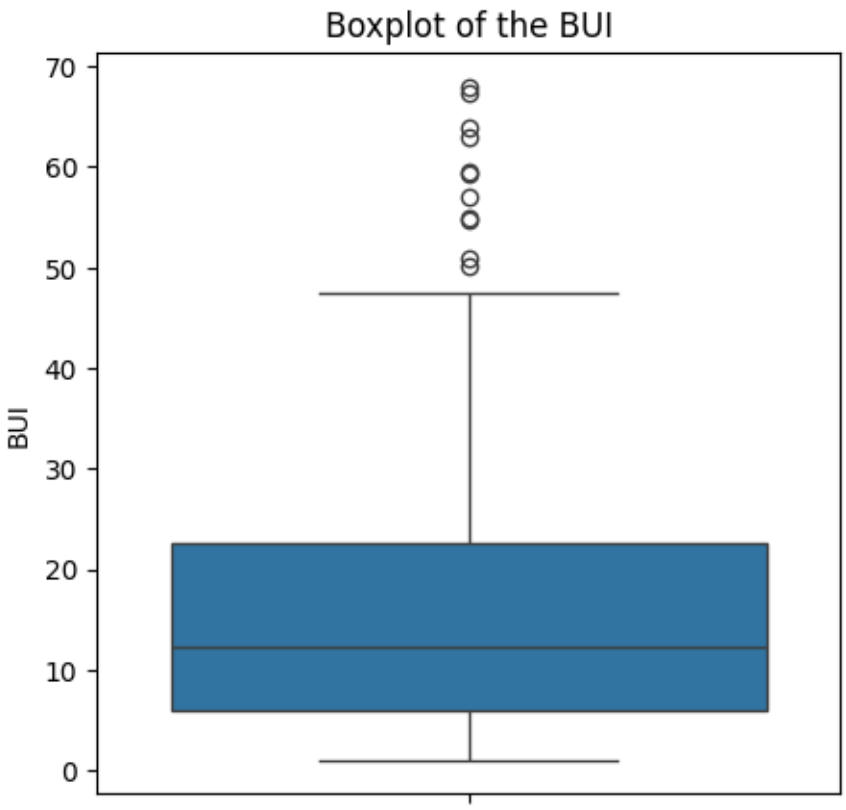

Boxplot of the Rain
Rain
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0

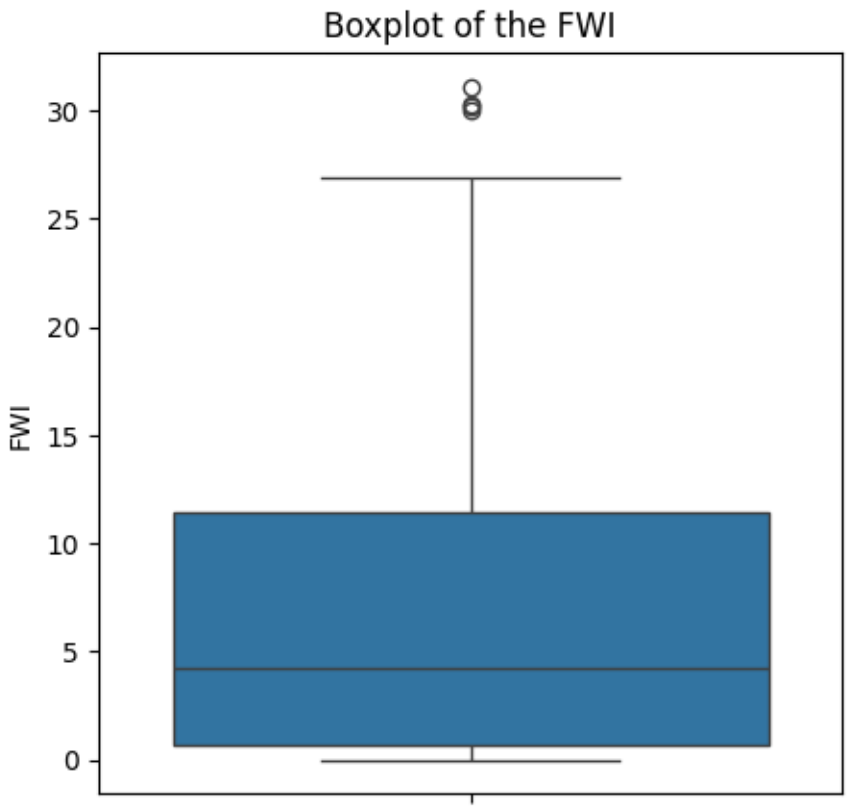

Boxplot of the FFMC
FFMC
90
80
70
60
50
40
30


Boxplot of the DMC
DMC
60
50
40
30
20
10
0

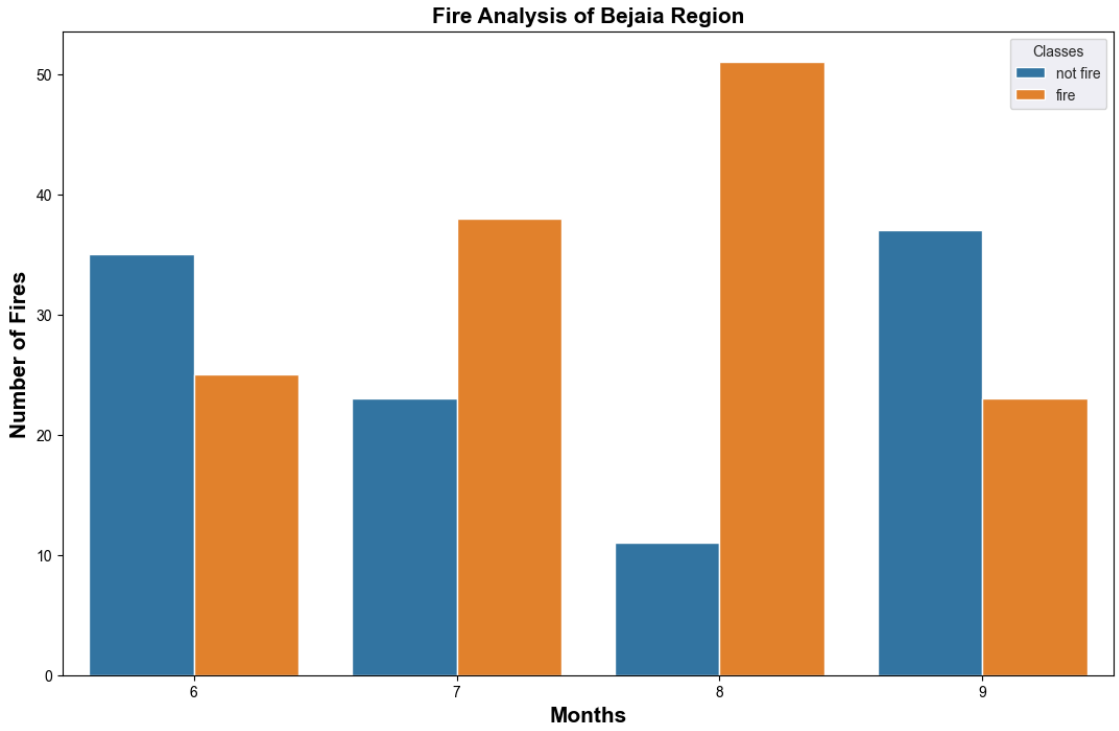

Boxplot of the DC
DC
200
150
100
50
0


Boxplot of the ISI
ISI
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0


Boxplot of the BUI
BUI
70
60
50
40
30
20
10
0



[38]:
```
## Monthly fire analysis
df['Classes'] = np.where(df['Classes'].str.contains('not fire'), 'not fire',␣
 ↪'fire')
plt.subplots(figsize=(13,8))
sns.set_style('darkgrid')
sns.countplot(x='month', hue='Classes', data=df)
plt.title('Fire Analysis of Bejaia Region', weight = 'bold', size = 15)
plt.ylabel('Number of Fires', weight = 'bold', size = 15)
plt.xlabel('Months', weight = 'bold', size = 15)
plt.show()
```



2 Feature Selection

[ ]:
```
df = pd.read_csv('..\\Dataset\Algerian_forest_fires_dataset_cleaned.csv')
```

[40]:
```
df.head()
```

[40]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	\
0	1	6	2012	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	
1	2	6	2012	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	
2	3	6	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	
3	4	6	2012	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	
4	5	6	2012	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	

	FWI	Classes	Region
0	0.5	not fire	0.0
1	0.4	not fire	0.0
2	0.1	not fire	0.0
3	0.0	not fire	0.0
4	0.5	not fire	0.0

[41]:
```
df.drop(['day', 'month', 'year'], axis=1, inplace=True)
```

```
df['Classes'] = np.where(df['Classes'].str.contains('not fire'), 0, 1) ## here 1 denotes the else part
```

[42]:
```
df.head()
```

[42]:

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	Region
0	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	0.5	0	0.0
1	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	0.4	0	0.0
2	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	0	0.0
3	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	0.0	0	0.0
4	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0.5	0	0.0

[43]:
```
df['Classes'].value_counts()
```

[43]:
```
Classes
1    137
0    106
Name: count, dtype: int64
```

[44]:
```
## Independent and dependent features
X =df.drop('FWI', axis=1)
y = df['FWI']
```

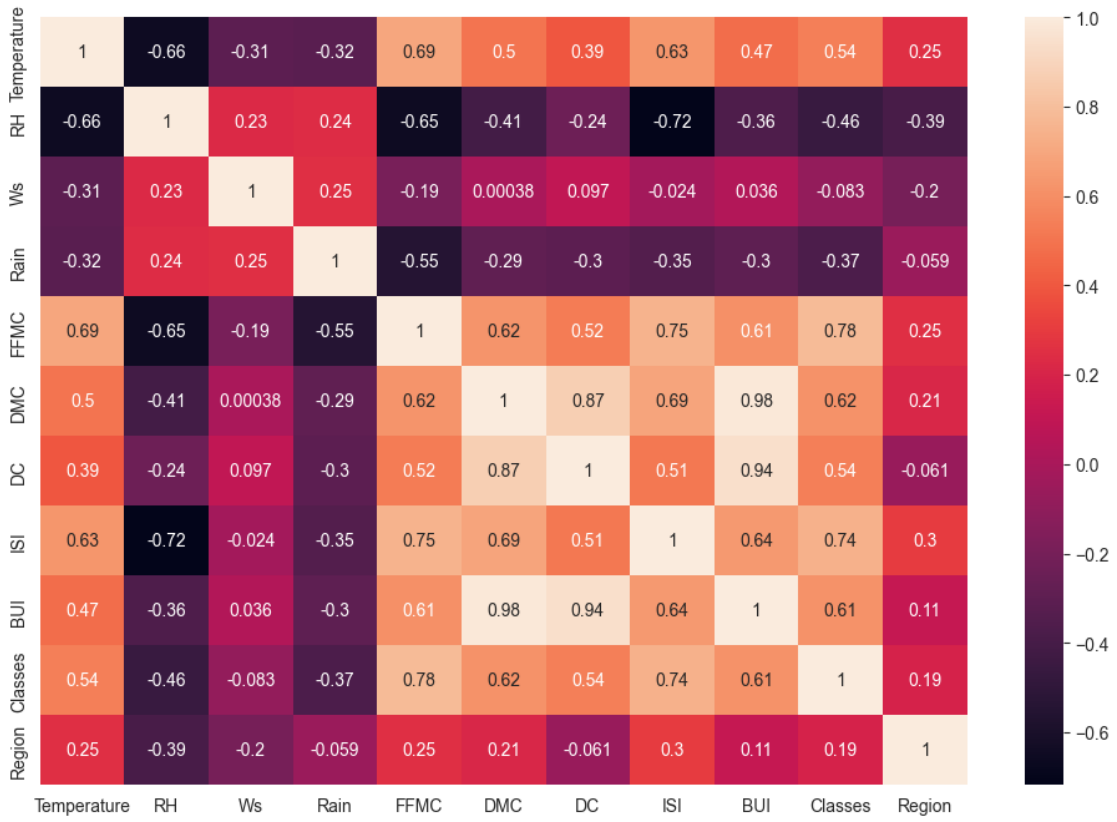
[45]:
```
X_train, X_test, y_train, y_test =train_test_split(X, y, test_size=0.25, random_state=42)
```

[46]:
```
X_train.shape, X_test.shape
```

[46]:
```
((182, 11), (61, 11))
```

2.0.1 Correlation of training data

[47]:
```
plt.figure(figsize=(12,8))
corr=X_train.corr()
sns.heatmap(corr, annot=True)
plt.show()
```



2.0.2 Function for Finding the best correlated data

[48]:
```
def correlation(dataset, threshold):
        col_corr = set()
        corr_matrix =dataset.corr()
        for i in range(len(corr_matrix.columns)):
                for j in range(i):
                        if abs(corr_matrix.iloc[i,j])>threshold:
                                colname = corr_matrix.columns[i]
                                col_corr.add(colname)
        return col_corr
```

[49]:
```
corr_features= correlation(X_train, 0.85) ## i am taking the 85% related data
```

[50]:
```
## Drop features when correlation is more than 0.85

X_train.drop(corr_features, axis = 1, inplace = True)
X_test.drop(corr_features, axis = 1, inplace = True)
```

[51]:
```
X_test.shape, X_train.shape
```

[51]: ((61, 9), (182, 9))

2.0.3 Feature Scaling Or Standardization

[52]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

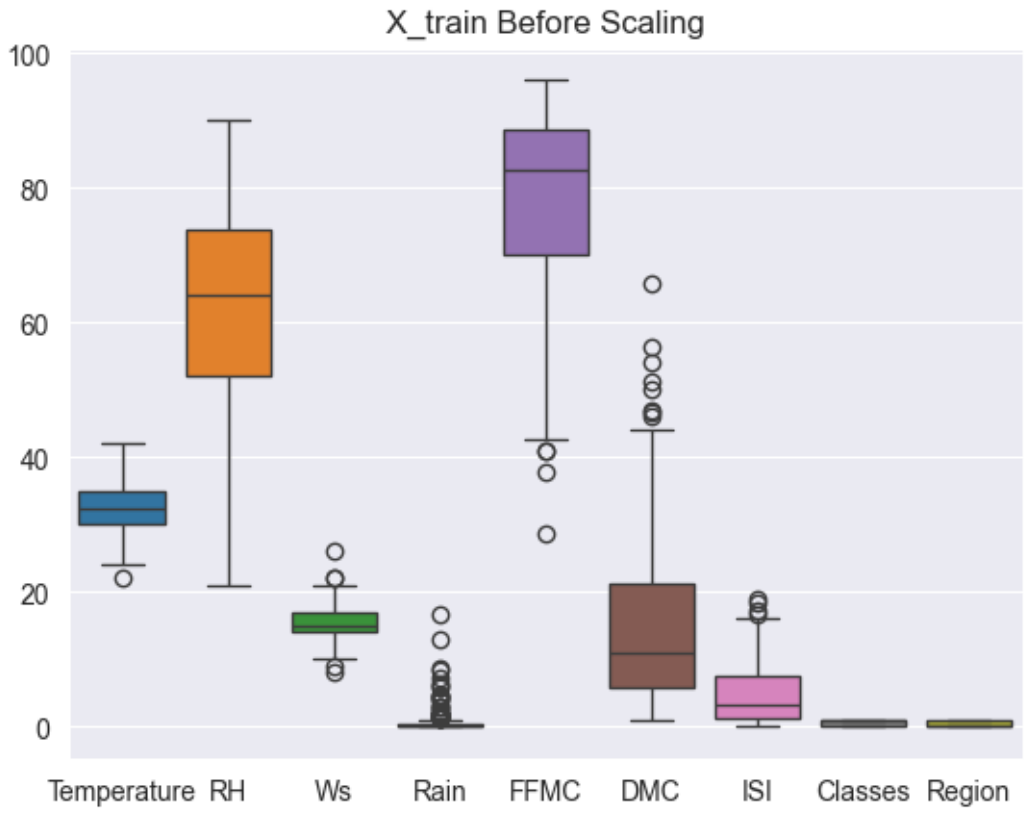
[53]:
```
X_train_scaled
```

[53]:
```
array([[-0.84284248,  0.78307967,  1.29972026, …, -0.62963326,
        -1.10431526, -0.98907071],
       [-0.30175842,  0.64950844, -0.59874754, …, -0.93058524,
        -1.10431526,  1.01105006],
       [ 2.13311985, -2.08870172, -0.21905398, …,  2.7271388 ,
         0.90553851,  1.01105006],
       …,
       [-1.9250106 ,  0.9166509 ,  0.54033314, …, -1.06948615,
        -1.10431526, -0.98907071],
       [ 0.50986767, -0.21870454,  0.16063958, …,  0.5973248 ,
         0.90553851,  1.01105006],
       [-0.57230045,  0.98343651,  2.05910739, …, -0.86113478,
        -1.10431526, -0.98907071]])
```

2.0.4 Box plot to understand effect of Standard Scaler

[54]:
```
sns.boxplot(data = X_train)
plt.title('X_train Before Scaling')
```

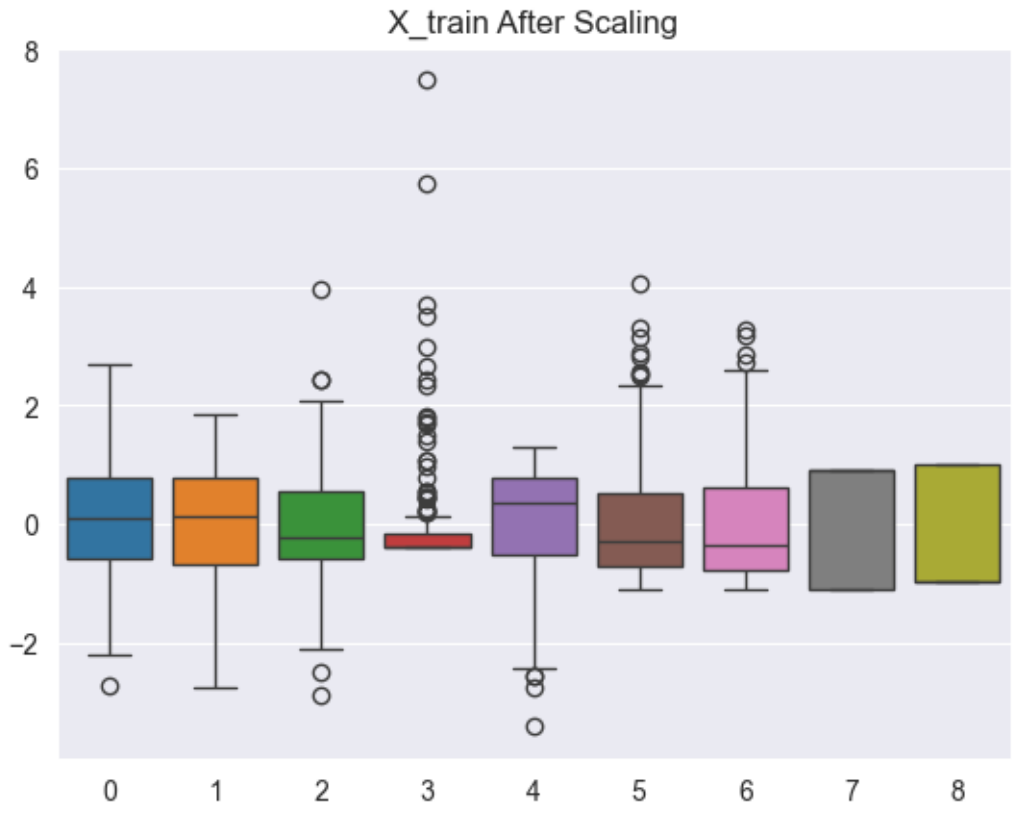
[54]:
```
Text(0.5, 1.0, 'X_train Before Scaling')
```



[55]:
```
## Optional for the subplots
# plt.subplots(figsize=(12,15))
# plt.subplot(2, 1, 1)
# sns.boxplot(data = X_train)
# plt.title('X_train Before Scaling')
# plt.subplot(2, 1, 2)
# sns.boxplot(data = X_train_scaled)
# plt.title('X_train After Scaling')
```

[56]:
```
sns.boxplot(data = X_train_scaled)
plt.title('X_train After Scaling')
```

[56]:
```
Text(0.5, 1.0, 'X_train After Scaling')
```



3 Linear Regression Model

[57]:
```
lin_reg = LinearRegression()
lin_reg.fit(X_train_scaled, y_train)
y_lin_pred = lin_reg.predict(X_test_scaled)
```

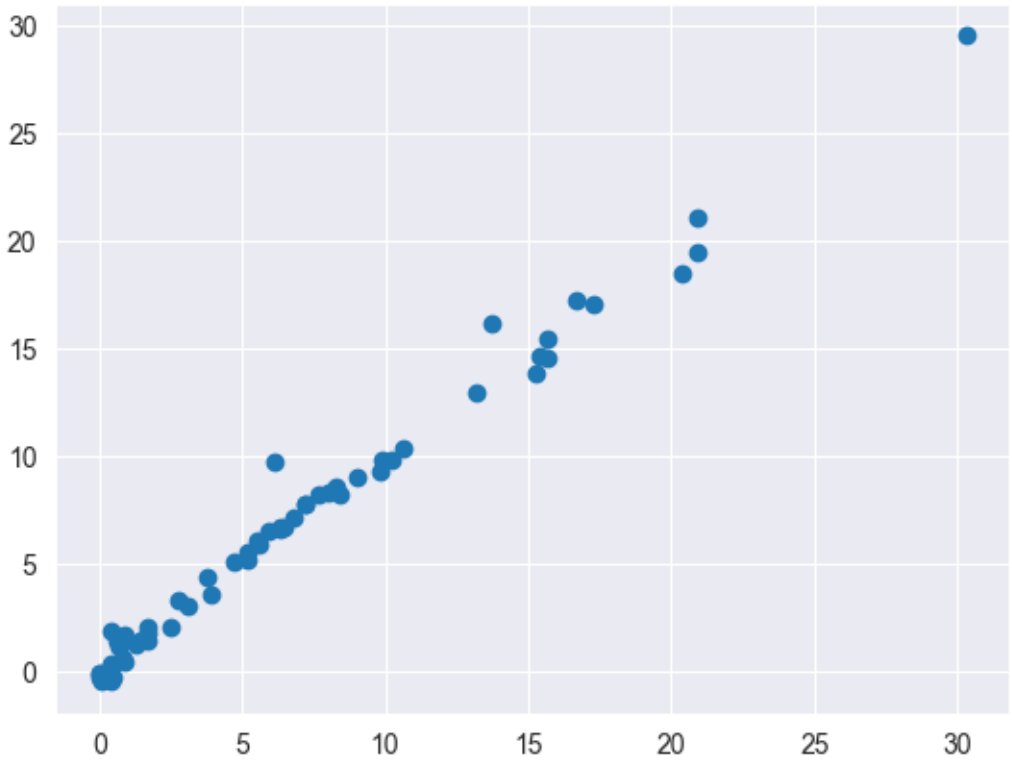
[58]:
```
mae = mean_absolute_error(y_test, y_lin_pred)
score = r2_score(y_test, y_lin_pred)
```

[59]:
```
print('mean_absolute_error: ', mae)
print('r2_score: ', score)
```

```
mean_absolute_error:  0.5468236465249978
r2_score:  0.9847657384266951
```

[60]:
```
plt.scatter(y_test, y_lin_pred)
```

[60]: <matplotlib.collections.PathCollection at 0x1668e8874f0>



4 Lasso Regression Model

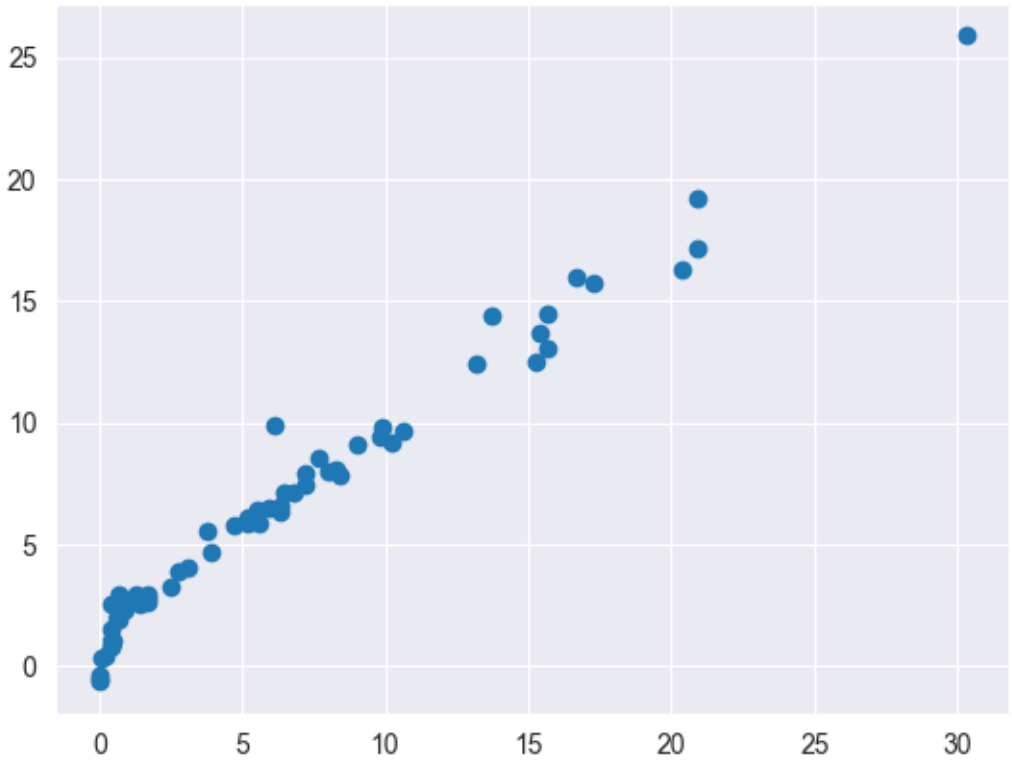
[61]:
```
from sklearn.linear_model import Lasso

lasso=Lasso()
lasso.fit(X_train_scaled, y_train)
y_laso_pred=lasso.predict(X_test_scaled)
mae = mean_absolute_error(y_test, y_laso_pred)
score = r2_score(y_test, y_laso_pred)
```

[62]:
```
print("mean_absolute_error: ", mae)
print('r2_score: ', score)
```

```
mean_absolute_error:  1.133175994914409
r2_score:  0.9492020263112388
```

[63]:
```
plt.scatter(y_test, y_laso_pred)
```

[63]: <matplotlib.collections.PathCollection at 0x1668e859de0>



5 Ridge Regression Model

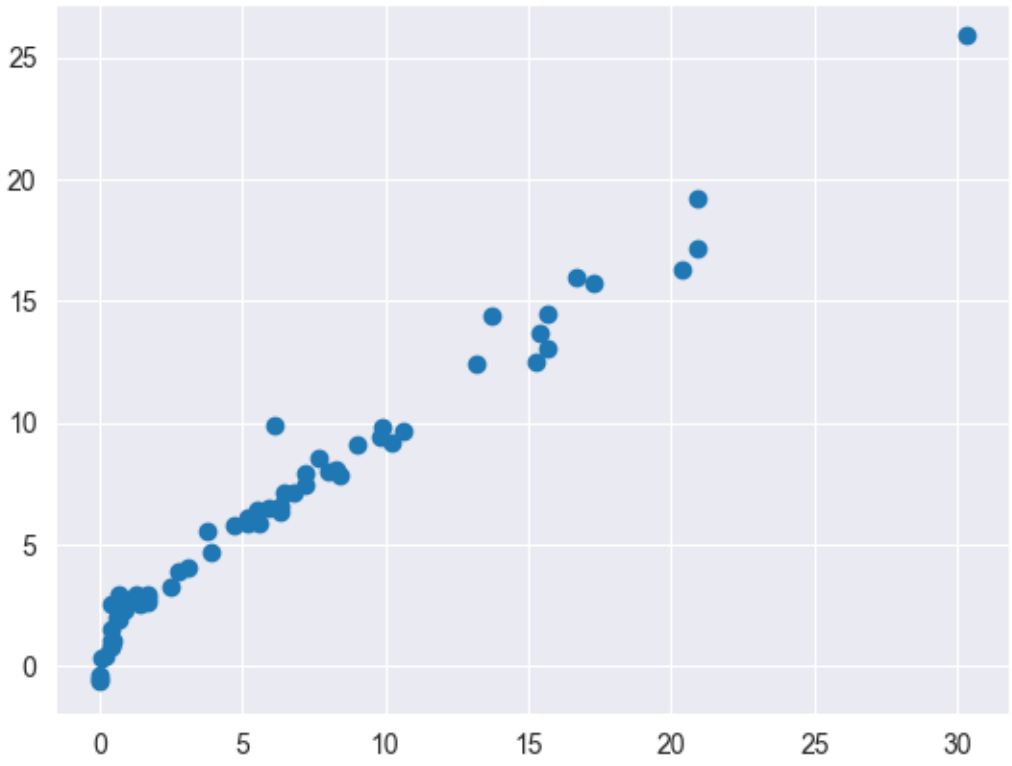
[64]:
```
from sklearn.linear_model import Ridge
ridge = Ridge()
ridge.fit(X_train_scaled, y_train)
y_ridge_pred = ridge.predict(X_test_scaled)
mae = mean_absolute_error(y_test, y_ridge_pred)
score = r2_score(y_test, y_ridge_pred)
```

[65]:
```
print("mean_absolute_error: ", mae)
print('r2_score: ', score)
```

```
mean_absolute_error:  0.5642305340105715
r2_score:  0.9842993364555512
```

[66]:
```
plt.scatter(y_test, y_laso_pred)
```

[66]: <matplotlib.collections.PathCollection at 0x16690d66b30>



6 ElasticNet Regression Model

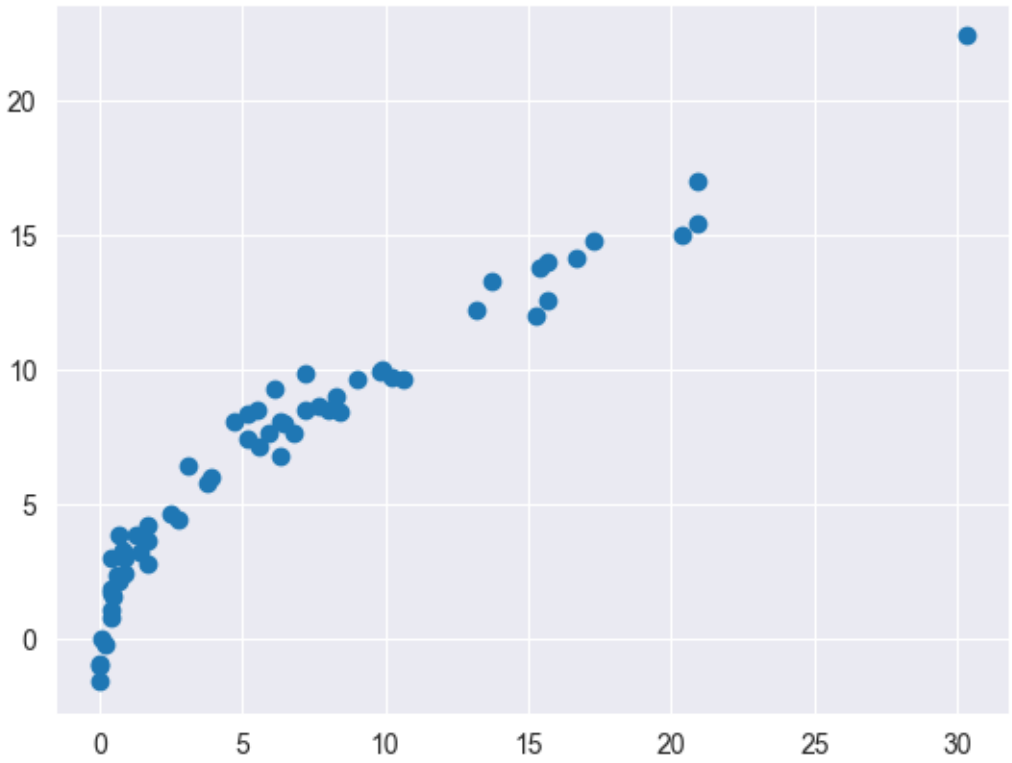
[67]:
```
from sklearn.linear_model import ElasticNet
elastic = ElasticNet()
elastic.fit(X_train_scaled, y_train)
y_elastic_pred = elastic.predict(X_test_scaled)
mae = mean_absolute_error(y_test, y_elastic_pred)
score = r2_score(y_test, y_elastic_pred)
```

[68]:
```
print('mean_absolute_error: ', mae)
print('r2_score: ', score)
```

```
mean_absolute_error:  1.8822353634896005
r2_score:  0.8753460589519703
```

[69]:
```
plt.scatter(y_test, y_elastic_pred)
```

[69]: <matplotlib.collections.PathCollection at 0x166923dbdc0>



7 Hyper-Parameter Tuning

7.1 Cross Validation Lasso

[70]:
```
from sklearn.linear_model import LassoCV
lassocv = LassoCV(cv=5)
lassocv.fit(X_train_scaled, y_train)
```

[70]:
```
LassoCV(cv=5)
```

[71]:
```
lassocv.predict(X_test_scaled)
```

[71]:
```
array([ 8.17490595,  7.68312478, -0.25676525,  4.72643402,  6.78715772,
        1.77624325,  2.23148094,  7.64057821,  1.99176323,  3.39941035,
        0.62808928,  9.95945488,  9.36168319, 16.98503659, 18.28488762,
        1.61644108,  1.62751276, -0.6415713 ,  7.28510526,  3.10926518,
        1.95541903,  0.18069335,  6.47563129,  0.14318503, 20.99597009,
        5.11755206,  5.86208849,  9.75914403, -0.77037467,  9.91838577,
        6.72277075, -0.31776007, 10.31109643, 14.4365551 ,  1.71022677,
        0.83439752,  2.03414915,  5.97488529, -0.6263644 , -0.56200288,
        6.47253729,  2.07971408,  8.46741557, -0.8464481 , 15.40443856,
        8.32941189,  8.48782486,  1.44030355, 13.02752812,  1.20911545,
```

```
        29.08623849,  5.49737681, 17.15937199, 19.28890096, 13.71102991,
        16.05355549,  0.99056448,  9.0873725 ,  3.84455993, 14.43991192,
         5.23034139])
```

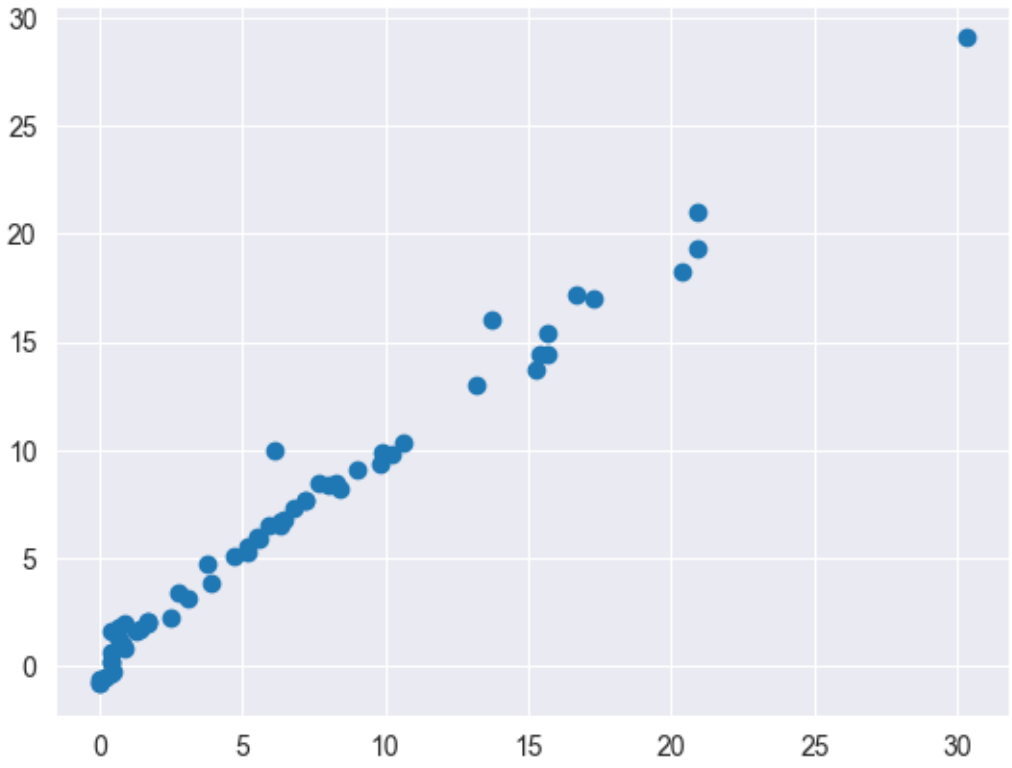
[72]:
```
### How many alphas it is trying to the mse path
lassocv.alphas_
```

[72]:
```
array([7.05853002, 6.58280872, 6.13914944, 5.72539132, 5.33951911,
       4.97965339, 4.64404142, 4.33104857, 4.03915039, 3.76692517,
       3.51304702, 3.27627941, 3.05546914, 2.84954075, 2.65749124,
       2.47838523, 2.31135036, 2.15557308, 2.01029467, 1.87480753,
       1.74845178, 1.63061198, 1.52071419, 1.41822315, 1.32263965,
       1.23349817, 1.15036452, 1.0728338 , 1.00052839, 0.93309613,
       0.87020857, 0.81155943, 0.75686304, 0.705853  , 0.65828087,
       0.61391494, 0.57253913, 0.53395191, 0.49796534, 0.46440414,
       0.43310486, 0.40391504, 0.37669252, 0.3513047 , 0.32762794,
       0.30554691, 0.28495408, 0.26574912, 0.24783852, 0.23113504,
       0.21555731, 0.20102947, 0.18748075, 0.17484518, 0.1630612 ,
       0.15207142, 0.14182231, 0.13226397, 0.12334982, 0.11503645,
       0.10728338, 0.10005284, 0.09330961, 0.08702086, 0.08115594,
       0.0756863 , 0.0705853 , 0.06582809, 0.06139149, 0.05725391,
       0.05339519, 0.04979653, 0.04644041, 0.04331049, 0.0403915 ,
       0.03766925, 0.03513047, 0.03276279, 0.03055469, 0.02849541,
       0.02657491, 0.02478385, 0.0231135 , 0.02155573, 0.02010295,
       0.01874808, 0.01748452, 0.01630612, 0.01520714, 0.01418223,
       0.0132264 , 0.01233498, 0.01150365, 0.01072834, 0.01000528,
       0.00933096, 0.00870209, 0.00811559, 0.00756863, 0.00705853])
```

[73]:
```
# lassocv.mse_path_ ## it gives you 500 values
```

[74]:
```
y_pred = lassocv.predict(X_test_scaled)
plt.scatter(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
score = r2_score(y_test, y_pred)
print("mean_absolute_error: ", mae)
print('r2_score: ', score)
```

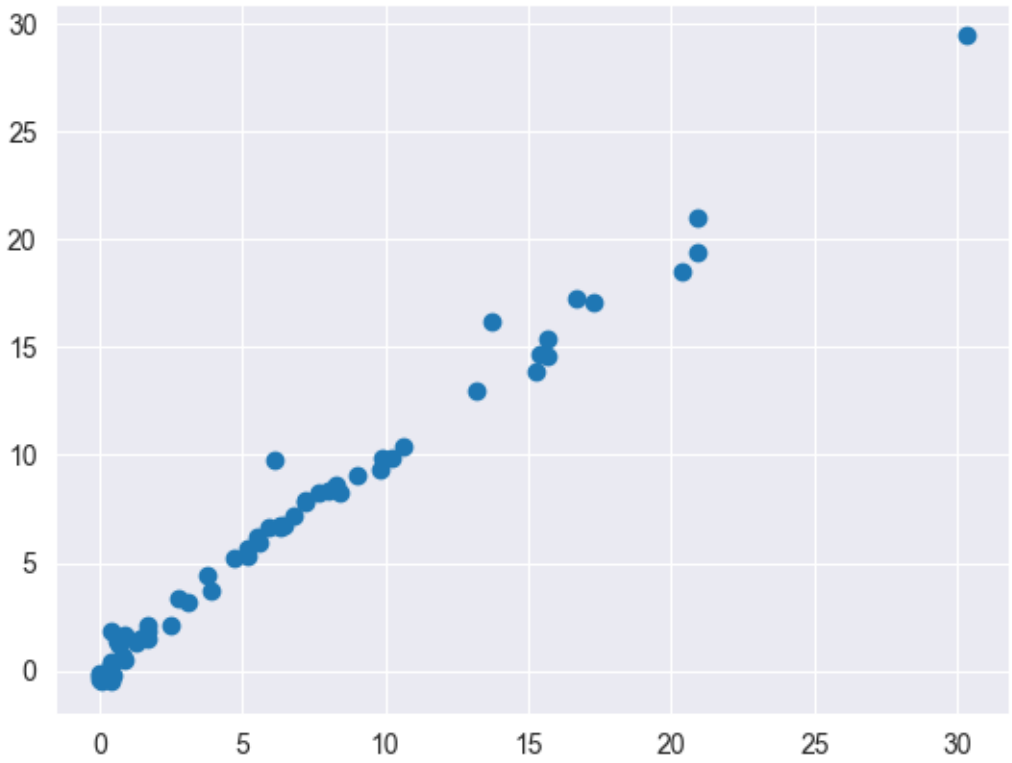
```
mean_absolute_error:  0.619970115826343
r2_score:  0.9820946715928275
```



7.2 Cross Validation Ridge

[75]:
```
from sklearn.linear_model import RidgeCV
ridgecv = RidgeCV(cv=5)
ridgecv.fit(X_train_scaled, y_train)
y_pred = ridgecv.predict(X_test_scaled)
plt.scatter(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
score = r2_score(y_test, y_pred)
print("mean_absolute_error: ", mae)
print('r2_score: ', score)
```

```
mean_absolute_error:  0.5642305340105715
r2_score:  0.9842993364555512
```



[76]:
```
ridgecv.alphas
```

[76]:
```
(0.1, 1.0, 10.0)
```

[77]:
```
ridgecv.get_params()
```

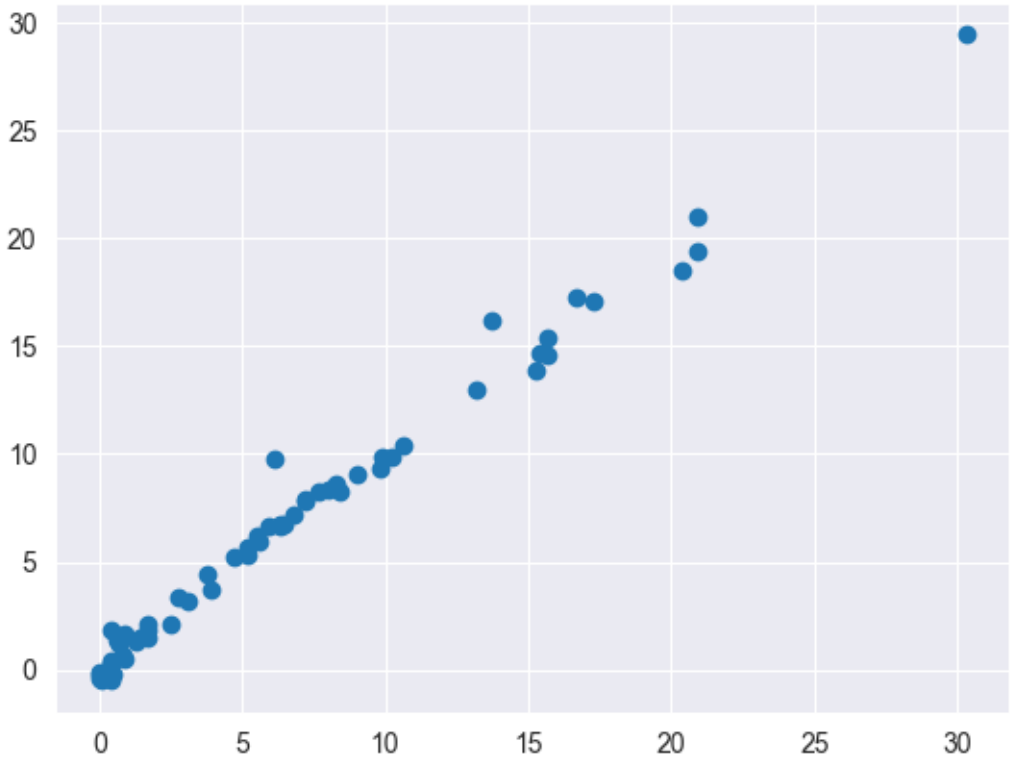
[77]:
```
{'alpha_per_target': False,
 'alphas': (0.1, 1.0, 10.0),
 'cv': 5,
 'fit_intercept': True,
 'gcv_mode': None,
 'scoring': None,
 'store_cv_results': None,
 'store_cv_values': 'deprecated'}
```

7.3 Cross Validation ElasticNet

[78]:
```
from sklearn.linear_model import ElasticNetCV
elasticcv = RidgeCV(cv=5)
elasticcv.fit(X_train_scaled, y_train)
y_pred = elasticcv.predict(X_test_scaled)
plt.scatter(y_test, y_pred)
```

```
mae = mean_absolute_error(y_test, y_pred)
score = r2_score(y_test, y_pred)
print("mean_absolute_error: ", mae)
print('r2_score: ', score)
```

```
mean_absolute_error:  0.5642305340105715
r2_score:  0.9842993364555512
```



[79]:
```
elasticcv.get_params()
```

[79]:
```
{'alpha_per_target': False,
 'alphas': (0.1, 1.0, 10.0),
 'cv': 5,
 'fit_intercept': True,
 'gcv_mode': None,
 'scoring': None,
 'store_cv_results': None,
 'store_cv_values': 'deprecated'}
```

7.4 Selecting the best model of the future predictions

[80]:
```
import pickle
pickle.dump(scaler, open('scaler.pkl', 'wb'))
pickle.dump(ridge, open('ridge.pkl', 'wb'))
```