About Dataset

The dataset includes **244 instances** that regroup a data of two regions of **Algeria**, namely the Bejaia region located in the northeast of Algeria and the Sidi Bel-abbes region located in the northwest of Algeria.

122 instances for each region.

The period from June 2012 to September 2012. The dataset includes 11 attributes and 1 output attribute (class) The 244 instances have been classified into fire (138 classes) and not fire (106 classes) classes.

Additional Variable Information

1. Date : (DD/MM/YYYY) Day, month ('june' to 'september'), year (2012) Weather data observations
2. Temp : temperature noon (temperature max) in Celsius degrees: 22 to 42
3. RH : Relative Humidity in %: 21 to 90
4. Ws :Wind speed in km/h: 6 to 29
5. Rain: total day in mm: 0 to 16.8 FWI Components
6. Fine Fuel Moisture Code (FFMC) index from the FWI system: 28.6 to 92.5
7. Duff Moisture Code (DMC) index from the FWI system: 1.1 to 65.9
8. Drought Code (DC) index from the FWI system: 7 to 220.4
9. Initial Spread Index (ISI) index from the FWI system: 0 to 18.5
10. Buildup Index (BUI) index from the FWI system: 1.1 to 68
11. Fire Weather Index (FWI) Index: 0 to 31.1
12. Classes: two classes, namely Fire and Not Fire

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
dataset = pd.read_csv("/content/Algerian_forest_fires_dataset_UPDATE.csv", header=1)
```

```
dataset.head()
```

	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

Next steps: Generate code with dataset | View recommended plots | New interactive sheet

```
dataset.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
```

Data Cleaning

```
## missing values
dataset.isnull().sum()
```

	0
day	0
month	1
year	1
Temperature	1
RH	1
Ws	1
Rain	1
FFMC	1
DMC	1
DC	1
ISI	1
BUI	1
FWI	1
Classes	2

dtype: int64

```
dataset.isnull().sum().sum()
```

```
14
```

```
dataset[dataset.isnull().any(axis =1)]
```

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes
122	Sidi-Bel Abbes Region Dataset	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
167		14	07	2012	37	37	18	0.2	88.9	12.9	14.6 9	12.5	10.4	fire	NaN

Add a new Column with region

```
dataset.loc[:122,["Region"]] = 0
dataset.loc[122:,["Region"]] = 1
dataset.head()
```

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	Region
0	01	06	2012	29	57	18	0	65.7	3.4	7.6	1.3	3.4	0.5	not fire	0.0
1	02	06	2012	29	61	13	1.3	64.4	4.1	7.6	1	3.9	0.4	not fire	0.0
2	03	06	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	not fire	0.0
3	04	06	2012	25	89	13	2.5	28.6	1.3	6.9	0	1.7	0	not fire	0.0
4	05	06	2012	27	77	16	0	64.8	3	14.2	1.2	3.9	0.5	not fire	0.0

Next steps: Generate code with `dataset` | View recommended plots | New interactive sheet

```
dataset.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
```

```
dataset["Region"] = dataset["Region"].astype(int)
```

```
dataset.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    int64  
dtypes: int64(1), object(14)
memory usage: 29.0+ KB
```

```
df = dataset
```

```
df.head(-1)
```

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	Region
0	01	06	2012	29	57	18	0	65.7	3.4	7.6	1.3	3.4	0.5	not fire	0
1	02	06	2012	29	61	13	1.3	64.4	4.1	7.6	1	3.9	0.4	not fire	0
2	03	06	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	not fire	0
3	04	06	2012	25	89	13	2.5	28.6	1.3	6.9	0	1.7	0	not fire	0
4	05	06	2012	27	77	16	0	64.8	3	14.2	1.2	3.9	0.5	not fire	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
240	25	09	2012	28	70	15	0	79.9	13.8	36.1	2.4	14.1	3	not fire	1
241	26	09	2012	30	65	14	0	85.4	16	44.5	4.5	16.9	6.5	fire	1
242	27	09	2012	28	87	15	4.4	41.1	6.5	8	0.1	6.2	0	not fire	1
243	28	09	2012	27	87	29	0.5	45.9	3.5	7.9	0.4	3.4	0.2	not fire	1
244	29	09	2012	24	54	18	0.1	79.7	4.3	15.2	1.7	5.1	0.7	not fire	1

245 rows × 15 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.isnull().sum()
```

	0
day	0
month	1
year	1
Temperature	1
RH	1
Ws	1
Rain	1
FFMC	1
DMC	1
DC	1
ISI	1
BUI	1
FWI	1
Classes	2
Region	0

dtype: int64

Remove NUll Values

```
df = df.dropna().reset_index(drop=True)
df.head()
```

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

Next steps: Generate code with df | View recommended plots | New interactive sheet

Double-click (or enter) to edit

```
df.isnull().sum()
```

	0
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

```
df.iloc[[122]]
```

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	Region
122	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	1

```
df = df.drop(index=122).reset_index(drop=True)
```

```
df.iloc[[122]]
```

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	Region
122	01	06	2012	32	71	12	0.7	57.1	2.5	8.2	0.6	2.8	0.2	not fire	1

```
df.columns
```

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

fix spaces in column names

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

```
df.columns
```

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

```
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    object
 1   month        243 non-null    object
 2   year         243 non-null    object
 3   Temperature  243 non-null    object
 4   RH           243 non-null    object
```

```
 5   Ws           243 non-null    object
 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    int64
dtypes: int64(1), object(14)
memory usage: 28.6+ KB
```

Change the required columns as integer datatype

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

```
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    int64
 1   month        243 non-null    int64
 2   year         243 non-null    int64
 3   Temperature  243 non-null    int64
 4   RH           243 non-null    int64
 5   Ws           243 non-null    int64
 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    int64
dtypes: int64(7), object(8)
memory usage: 28.6+ KB
```

Changing the other columns to float datatype

```
objects = [features for features in df.columns if df[features].dtype == 'O']
# shows which features have O as an object class
```

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

```
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    int64
 1   month        243 non-null    int64
 2   year         243 non-null    int64
 3   Temperature  243 non-null    int64
 4   RH           243 non-null    int64
 5   Ws           243 non-null    int64
 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    int64
dtypes: float64(7), int64(7), object(1)
memory usage: 28.6+ KB
```

```
df.describe()
```

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	
count	243.000000	243.000000	243.0	243.000000	243.000000	243.000000	243.000000	243.000000	243.000000	243.000000	243.000000	243.0(
mean	15.761317	7.502058	2012.0	32.152263	62.041152	15.493827	0.762963	77.842387	14.680658	49.430864	4.742387	16.6
std	8.842552	1.114793	0.0	3.628039	14.828160	2.811385	2.003207	14.349641	12.393040	47.665606	4.154234	14.2
min	1.000000	6.000000	2012.0	22.000000	21.000000	6.000000	0.000000	28.600000	0.700000	6.900000	0.000000	1.1(
25%	8.000000	7.000000	2012.0	30.000000	52.500000	14.000000	0.000000	71.850000	5.800000	12.350000	1.400000	6.0(
50%	16.000000	8.000000	2012.0	32.000000	63.000000	15.000000	0.000000	83.300000	11.300000	33.100000	3.500000	12.4(
75%	23.000000	8.000000	2012.0	35.000000	73.500000	17.000000	0.500000	88.300000	20.800000	69.100000	7.250000	22.6
max	31.000000	9.000000	2012.0	42.000000	90.000000	29.000000	16.800000	96.000000	65.900000	220.400000	19.000000	68.0(

```
df.head()
```

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	Region
0	1	6	2012	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	0.5	not fire	0
1	2	6	2012	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	0.4	not fire	0
2	3	6	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	not fire	0
3	4	6	2012	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	0.0	not fire	0
4	5	6	2012	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0.5	not fire	0

Next steps: Generate code with df | View recommended plots | New interactive sheet

Lets Save the Clean dataset

```
df.to_csv('Clean_Algerian_forest_fires_cleaned_dataset.csv', index= False)
```

Exploratory Data Analysis

```
df_copy = df
```

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

	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	not fire	0
1	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	0.4	not fire	0
2	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	not fire	0
3	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	0.0	not fire	0
4	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0.5	not fire	0
...	...	...	...	...	...	...	...	...	...	...	...	...
238	30	65	14	0.0	85.4	16.0	44.5	4.5	16.9	6.5	fire	1
239	28	87	15	4.4	41.1	6.5	8.0	0.1	6.2	0.0	not fire	1
240	27	87	29	0.5	45.9	3.5	7.9	0.4	3.4	0.2	not fire	1
241	24	54	18	0.1	79.7	4.3	15.2	1.7	5.1	0.7	not fire	1
242	24	64	15	0.2	67.3	3.8	16.5	1.2	4.8	0.5	not fire	1

243 rows × 12 columns

Encoding of the categories in classes

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

Classes	count
fire	131
not fire	101
fire	4
fire	2
not fire	2
not fire	1
not fire	1
not fire	1

dtype: int64

```
df_copy['Classes'] = np.where(df_copy['Classes'].str.contains( 'not fire'),0, 1)
```

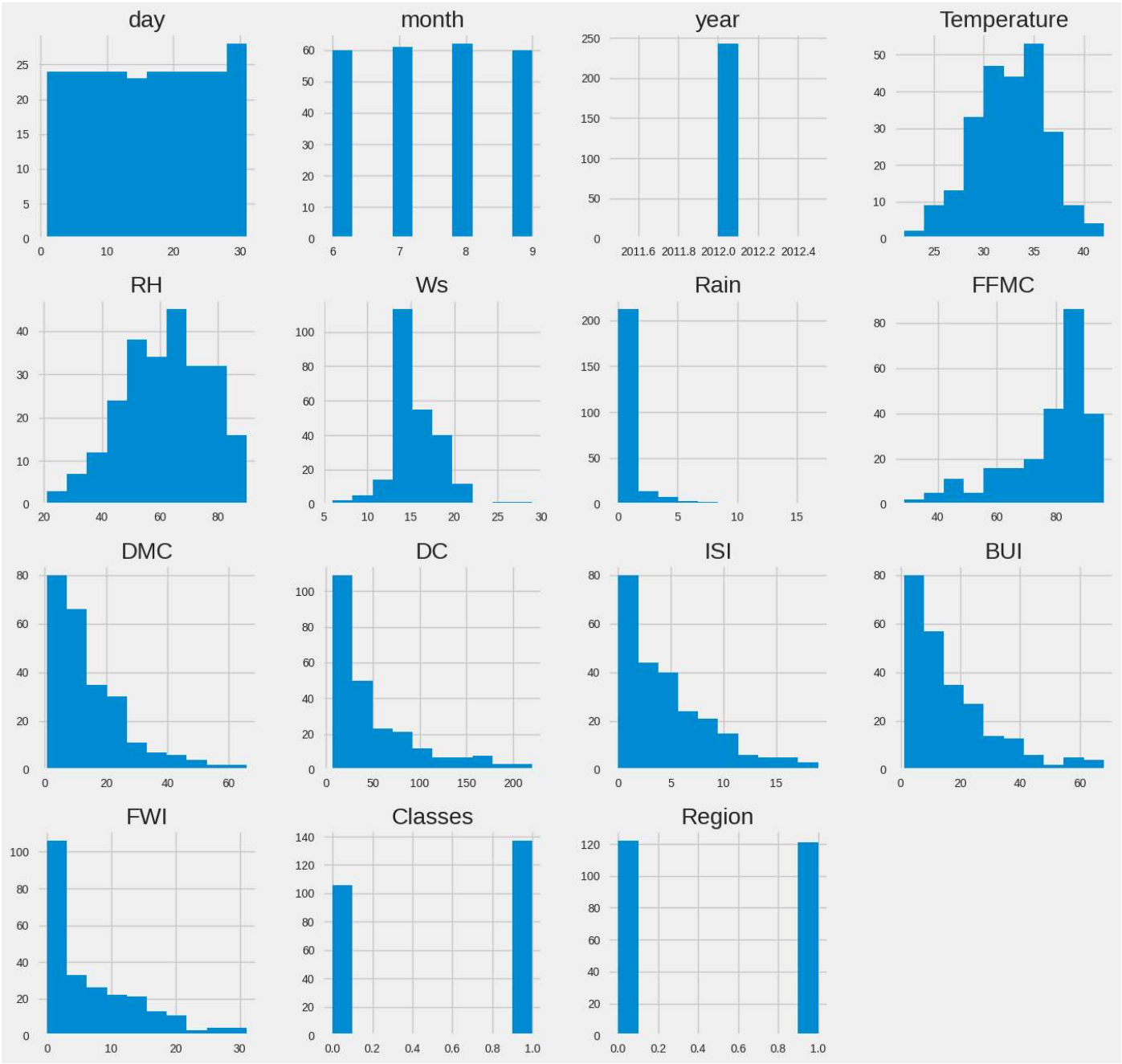
```
df_copy.head()
df.drop('classes', axis =1, inplace= True)
```

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

Classes	count
1	137
0	106

dtype: int64

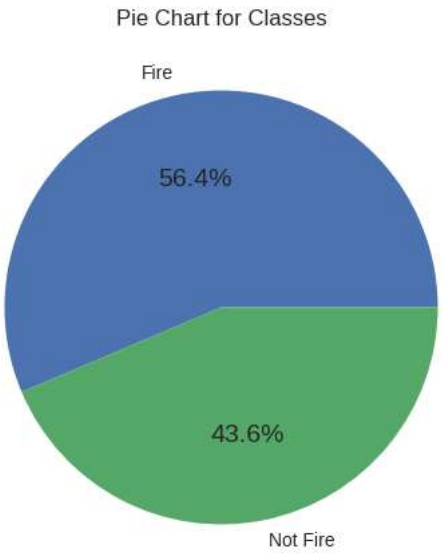
```
## Plot density plot for all features
plt.style.use('fivethirtyeight')
df_copy.hist(figsize=(15,15))
plt.show()
```




```
## Percentage for Pie Chart
percentage = df_copy['Classes'].value_counts(normalize = True)*100
```

```
# Plotting piechart
classlabels = ['Fire', 'Not Fire']
plt.figure(figsize = (5,5))
plt.pie(percentage, labels=classlabels, autopct = '%1.1f%%' )
```

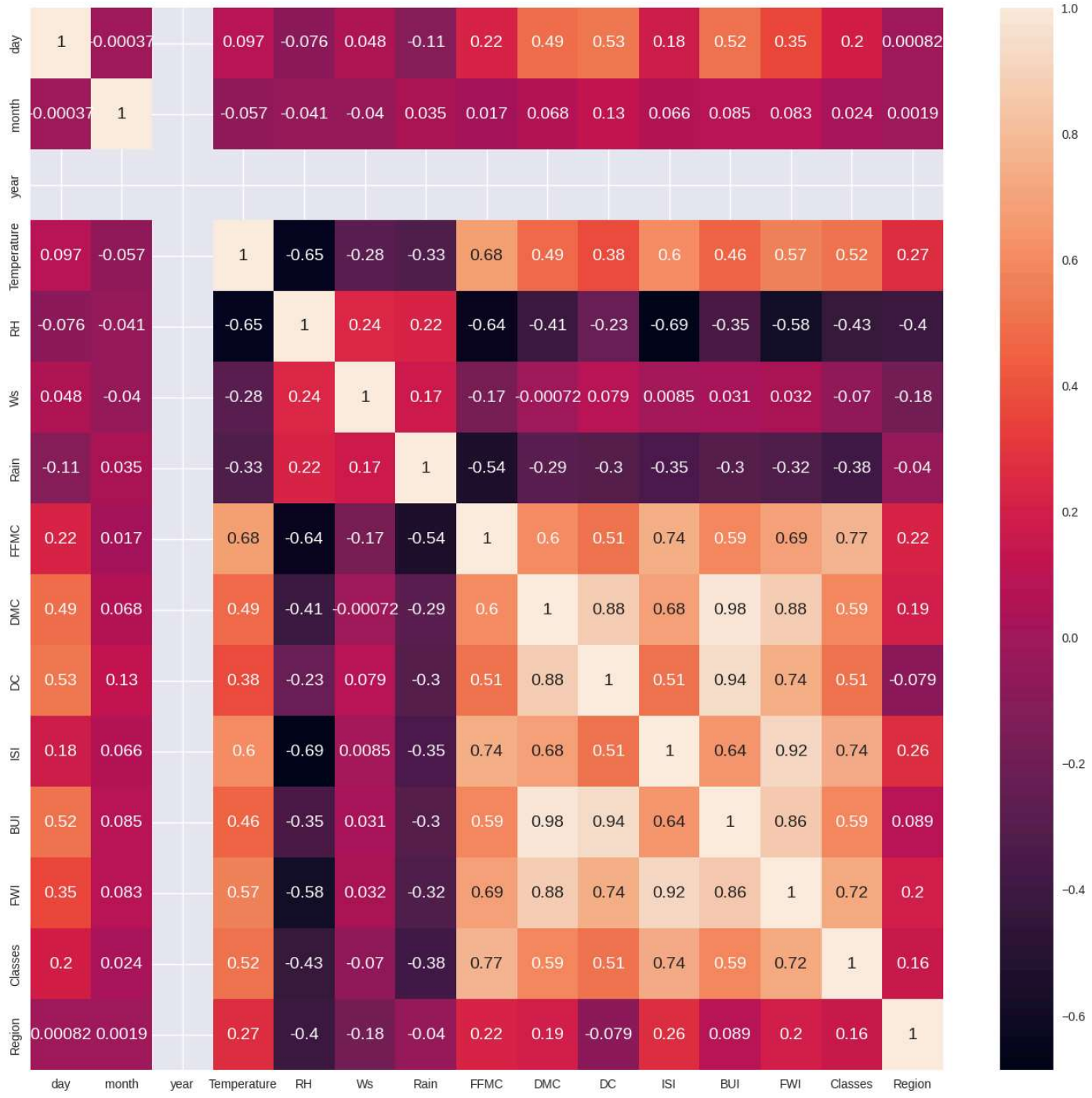
```
plt.title('Pie Chart for Classes')
plt.show()
```



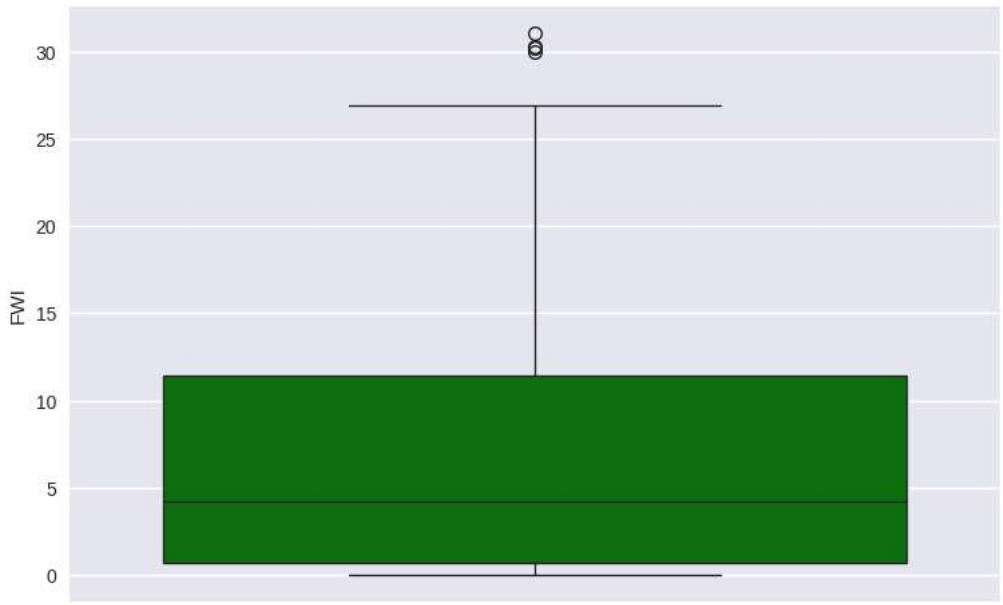

```
df.corr()
```

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	
day	1.000000	-0.000369	NaN	0.097227	-0.076034	0.047812	-0.112523	0.224956	0.491514	0.527952	0.180543	0.517117	0.3
month	-0.000369	1.000000	NaN	-0.056781	-0.041252	-0.039880	0.034822	0.017030	0.067943	0.126511	0.065608	0.085073	0.0
year	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
Temperature	0.097227	-0.056781	NaN	1.000000	-0.651400	-0.284510	-0.326492	0.676568	0.485687	0.376284	0.603871	0.459789	0.5
RH	-0.076034	-0.041252	NaN	-0.651400	1.000000	0.244048	0.222356	-0.644873	-0.408519	-0.226941	-0.686667	-0.353841	-0.5
Ws	0.047812	-0.039880	NaN	-0.284510	0.244048	1.000000	0.171506	-0.166548	-0.000721	0.079135	0.008532	0.031438	0.0
Rain	-0.112523	0.034822	NaN	-0.326492	0.222356	0.171506	1.000000	-0.543906	-0.288773	-0.298023	-0.347484	-0.299852	-0.3
FFMC	0.224956	0.017030	NaN	0.676568	-0.644873	-0.166548	-0.543906	1.000000	0.603608	0.507397	0.740007	0.592011	0.6
DMC	0.491514	0.067943	NaN	0.485687	-0.408519	-0.000721	-0.288773	0.603608	1.000000	0.875925	0.680454	0.982248	0.8
DC	0.527952	0.126511	NaN	0.376284	-0.226941	0.079135	-0.298023	0.507397	0.875925	1.000000	0.508643	0.941988	0.7
ISI	0.180543	0.065608	NaN	0.603871	-0.686667	0.008532	-0.347484	0.740007	0.680454	0.508643	1.000000	0.644093	0.9
BUI	0.517117	0.085073	NaN	0.459789	-0.353841	0.031438	-0.299852	0.592011	0.982248	0.941988	0.644093	1.000000	0.8
FWI	0.350781	0.082639	NaN	0.566670	-0.580957	0.032368	-0.324422	0.691132	0.875864	0.739521	0.922895	0.857973	1.0
Classes	0.202840	0.024004	NaN	0.516015	-0.432161	-0.069964	-0.379097	0.769492	0.585658	0.511123	0.735197	0.586639	0.7
Region	0.000821	0.001857	NaN	0.269555	0.402682	0.181160	0.040013	0.222241	0.192089	0.078734	0.263197	0.089408	0.1

```
## Correlation
plt.figure(figsize = (15,15))
sns.heatmap(df_copy.corr(), annot = True)
plt.show()
```




```
## Box Plots
sns.boxplot(df_copy['FWI'], color = 'green')
plt.show()
```



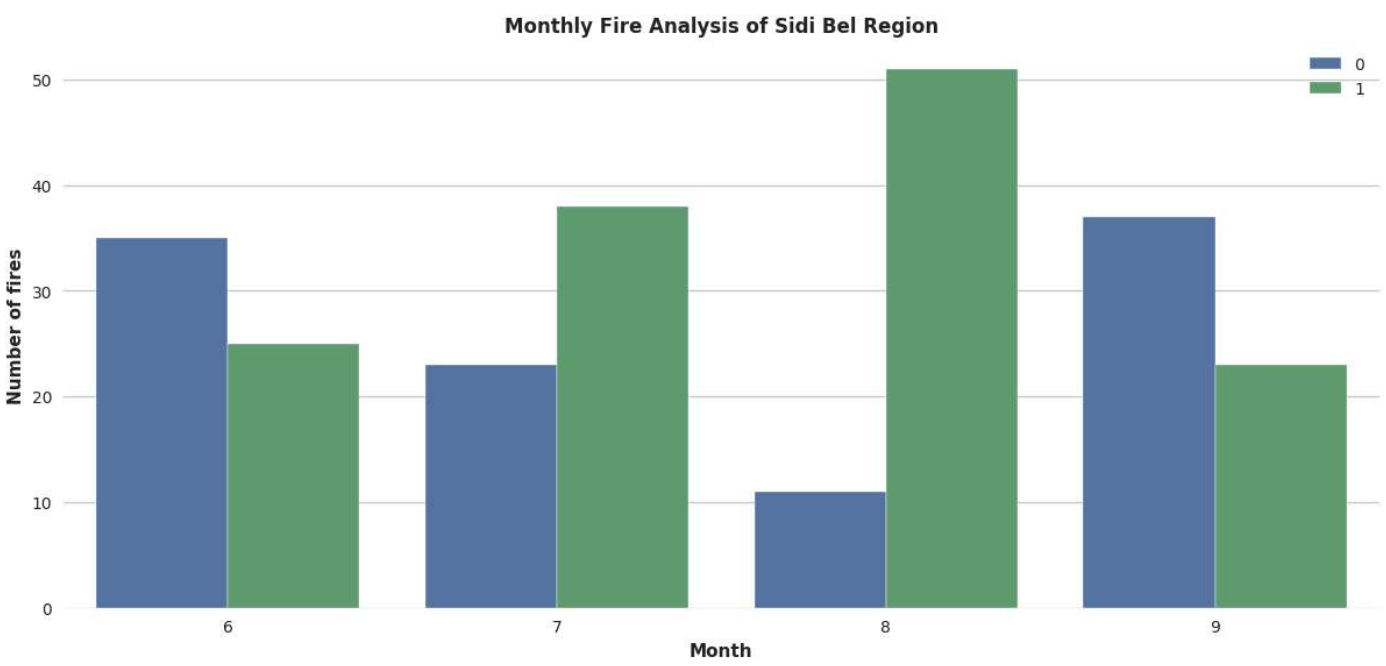

```
df.head()
```

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	Region
0	1	6	2012	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	0.5	0	0
1	2	6	2012	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	0.4	0	0
2	3	6	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	0	0
3	4	6	2012	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	0.0	0	0
4	5	6	2012	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0.5	0	0

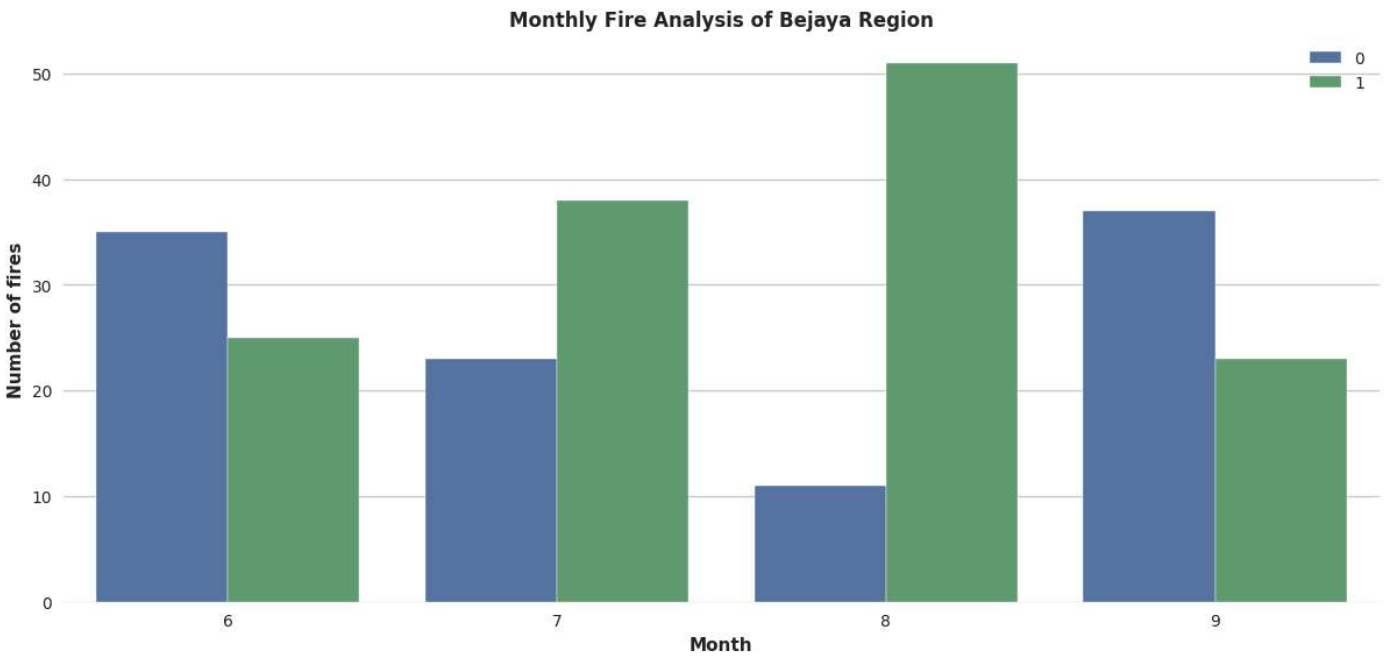
Next steps: Generate code with df | View recommended plots | New interactive sheet

Monthly Fire Analysis

```
df.temp = [df.loc[df['Region']]== 1]
plt.subplots(figsize=(13,6))
sns.set_style('whitegrid')
sns.countplot(x='month', data=df, hue='Classes')
plt.legend(loc='upper right')
plt.ylabel('Number of fires', weight = 'bold')
plt.xlabel('Month', weight = 'bold')
plt.title('Monthly Fire Analysis of Sidi Bel Region', weight = 'bold')
plt.show()
```




```
df.temp = [df.loc[df['Region']]== 0]
plt.subplots(figsize=(13,6))
sns.set_style('whitegrid')
sns.countplot(x='month', data=df, hue='Classes')
plt.legend(loc='upper right')
plt.ylabel('Number of fires', weight = 'bold')
plt.xlabel('Month', weight = 'bold')
plt.title('Monthly Fire Analysis of Bejaya Region', weight = 'bold')
plt.show()
```



Feature Selection

```
df.head()
```

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	Region
0	1	6	2012	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	0.5	0	0
1	2	6	2012	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	0.4	0	0
2	3	6	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	0	0
3	4	6	2012	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	0.0	0	0
4	5	6	2012	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0.5	0	0

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.columns
```

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

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

```
df.head()
```

	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
1	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	0.4	0	0
2	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	0	0
3	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	0.0	0	0
4	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0.5	0	0

Next steps: Generate code with df | View recommended plots | New interactive sheet

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

Classes	count
1	137
0	106

dtype: int64

Independent and dependent features

```
X = df.drop('FWI', axis =1)
y = df['FWI']
```

```
X.head()
```

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

Next steps: Generate code with X | View recommended plots | New interactive sheet

```
y
```

	FWI
0	0.5
1	0.4
2	0.1
3	0.0
4	0.5
...	...
238	6.5
239	0.0
240	0.2
241	0.7
242	0.5

243 rows × 1 columns

dtype: float64

Train Test Split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 42)
```

```
X_train.shape, X_test.shape
```

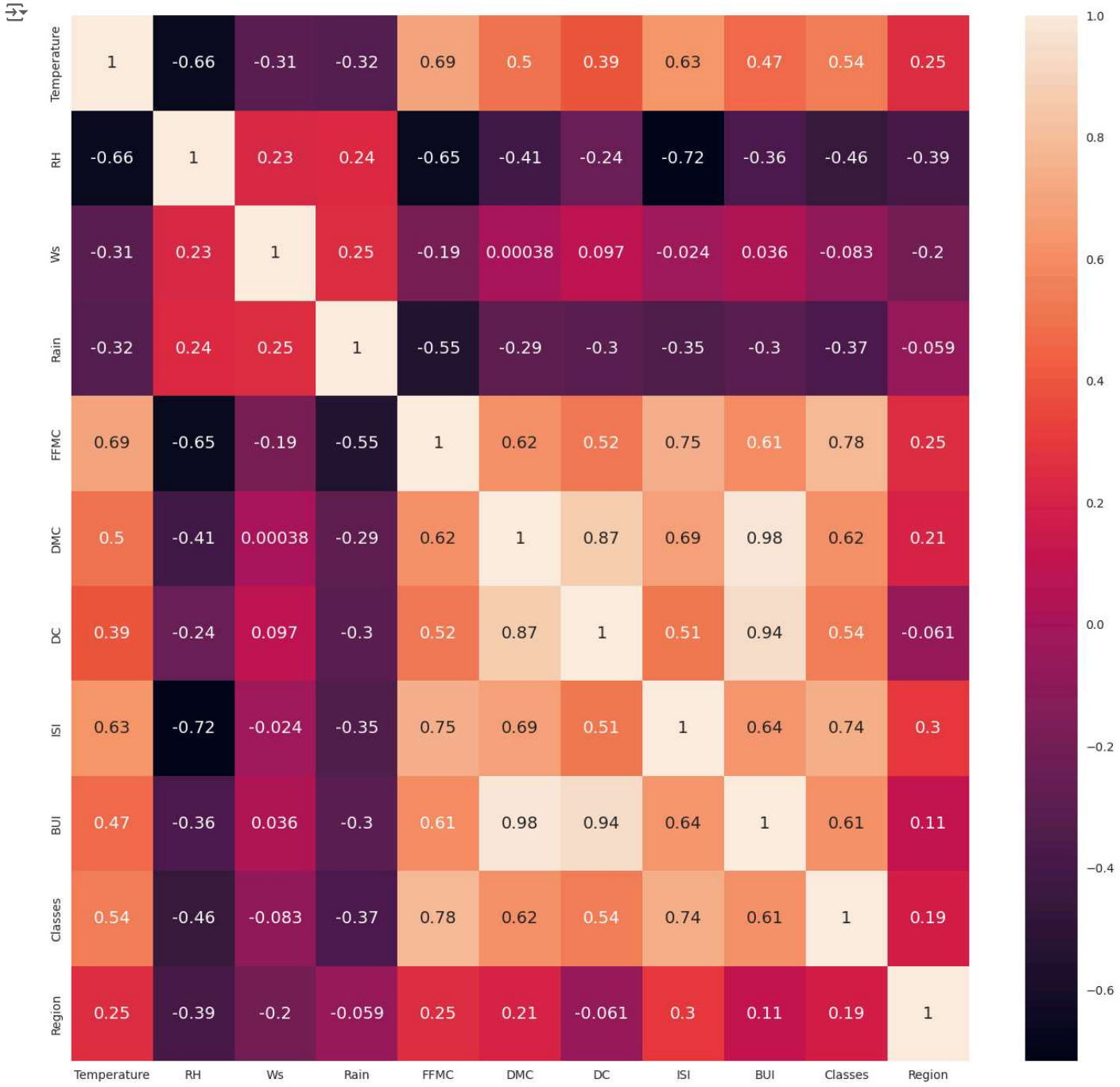
```
((182, 11), (61, 11))
```

Feature Selection based on correlation

```
X_train.corr()
```

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	Classes	Region
Temperature	1.000000	-0.656095	-0.305977	-0.317512	0.694768	0.498173	0.390684	0.629848	0.473609	0.542141	0.254549
RH	-0.656095	1.000000	0.225736	0.241656	-0.653023	-0.414601	-0.236078	-0.717804	-0.362317	-0.456876	-0.394665
Ws	-0.305977	0.225736	1.000000	0.251932	-0.190076	0.000379	0.096576	-0.023558	0.035633	-0.082570	-0.199969
Rain	-0.317512	0.241656	0.251932	1.000000	-0.545491	-0.289754	-0.302341	-0.345707	-0.300964	-0.369357	-0.059022
FFMC	0.694768	-0.653023	-0.190076	-0.545491	1.000000	0.620807	0.524101	0.750799	0.607210	0.781259	0.249514
DMC	0.498173	-0.414601	0.000379	-0.289754	0.620807	1.000000	0.868647	0.685656	0.983175	0.617273	0.212582
DC	0.390684	-0.236078	0.096576	-0.302341	0.524101	0.868647	1.000000	0.513701	0.942414	0.543581	-0.060838
ISI	0.629848	-0.717804	-0.023558	-0.345707	0.750799	0.685656	0.513701	1.000000	0.643818	0.742977	0.296441
BUI	0.473609	-0.362317	0.035633	-0.300964	0.607210	0.983175	0.942414	0.643818	1.000000	0.612239	0.114897
Classes	0.542141	-0.456876	-0.082570	-0.369357	0.781259	0.617273	0.543581	0.742977	0.612239	1.000000	0.188837
Region	0.254549	-0.394665	-0.199969	-0.059022	0.249514	0.212582	-0.060838	0.296441	0.114897	0.188837	1.000000

```
## Check for multicolinearity
plt.figure(figsize = (15,15))
sns.heatmap(X_train.corr(), annot = True)
plt.show()
```




```
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
# Remvove the feature that is highly correlated
```

```
# THRESHOLD -- Domain expertise
corr_features = correlation(X_train, 0.85)
```

```
## Drop features when the correlation is more than 0.85
X_train.drop(corr_features, axis =1, inplace= True)
X_test.drop(corr_features, axis =1, inplace= True)
```

```
X_train.shape, X_test.shape
```

```
((182, 9), (61, 9))
```

Feature Scaling or Standardization

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
X_train_scaled
```

```
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]])
```

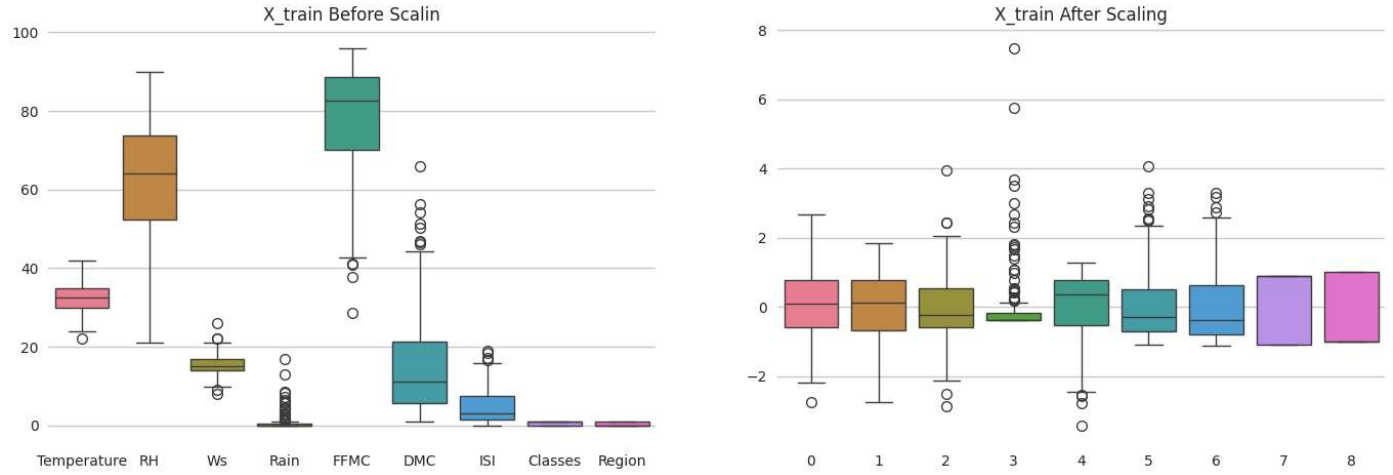
```
X_test_scaled
```

```
array([[-3.01758418e-01,  1.15223531e-01, -2.19053977e-01,
        -3.84060174e-01,  6.33218240e-01, -4.25075679e-02,
         2.03772218e-01,  9.05538514e-01, -9.89070710e-01],
       [ 2.39325642e-01, -5.52632606e-01, -9.78441098e-01,
        -3.84060174e-01,  7.37980727e-01, -3.83352062e-01,
         3.65823283e-01,  9.05538514e-01, -9.89070710e-01],
       [-1.11338451e+00, -2.85490151e-01,  9.20026704e-01,
         6.45241658e-01, -9.73139891e-01, -9.14435344e-01,
        -8.37984627e-01, -1.10431526e+00,  1.01105006e+00],
       [ 5.09867672e-01, -2.85490151e-01, -9.78441098e-01,
        -2.90487280e-01,  1.30358303e-01,  3.14190159e-01,
        -6.29633258e-01, -1.10431526e+00,  1.01105006e+00],
       [-5.72300448e-01,  1.82009145e-01, -5.98747538e-01,
        -3.84060174e-01,  5.42424085e-01,  1.00171523e-01,
        -7.40296073e-02,  9.05538514e-01,  1.01105006e+00],
       [ 1.86257782e+00,  1.15223531e-01, -2.49721534e+00,
         1.77377189e-01, -2.67739147e-01, -2.40672972e-01,
        -8.61134779e-01, -1.10431526e+00,  1.01105006e+00],
       [-1.11338451e+00,  8.49865282e-01,  1.60639583e-01,
        -3.84060174e-01,  3.18930780e-01, -8.19315950e-01,
        -3.51831432e-01,  9.05538514e-01,  1.01105006e+00],
       [ 1.32149376e+00, -1.75477365e+00, -9.78441098e-01,
        -1.03341493e-01,  5.98297411e-01,  2.50777229e-01,
        -4.57915097e-03,  9.05538514e-01,  1.01105006e+00],
       [-1.11338451e+00,  1.11700774e+00, -1.35813466e+00,
        -3.84060174e-01, -3.09644141e-01, -4.15058527e-01,
        -8.14834475e-01, -1.10431526e+00, -9.89070710e-01],
       [-3.01758418e-01, -8.19775060e-01,  1.29972026e+00,
        -1.03341493e-01,  1.16274848e-02, -3.27865749e-01,
        -5.60182801e-01, -1.10431526e+00, -9.89070710e-01],
       [-1.11338451e+00,  9.16650896e-01,  2.05910739e+00,
        -3.84060174e-01, -3.51549136e-01, -6.13223930e-01,
        -9.53735387e-01, -1.10431526e+00, -9.89070710e-01],
       [ 7.80409702e-01, -4.19061378e-01, -5.98747538e-01,
        -1.96914386e-01,  1.09405806e-01,  1.76476091e+00,
        -6.29633258e-01, -1.10431526e+00,  1.01105006e+00],
```

```
[-3.12163881e-02, -7.52989447e-01, -9.78441098e-01,
 -3.84060174e-01,  7.72901556e-01,  1.00171523e-01,
  4.81574043e-01,  9.05538514e-01,  1.01105006e+00],
[ 1.05095173e+00, -8.19775060e-01,  1.60639583e-01,
 -3.84060174e-01,  8.56711545e-01,  1.42391642e+00,
  1.08347800e+00,  9.05538514e-01,  1.01105006e+00],
[ 1.05095173e+00, -4.85846992e-01,  9.20026704e-01,
 -3.84060174e-01,  8.00838219e-01,  1.48732935e+00,
  1.17607861e+00,  9.05538514e-01, -9.89070710e-01],
[ 5.09867672e-01, -7.52989447e-01,  1.60639583e-01,
  1.39382481e+00, -9.32501256e-03, -5.33957769e-01,
 -6.52783410e-01, -1.10431526e+00,  1.01105006e+00],
[-1.65446857e+00, -8.86560674e-01, -3.63629602e+00,
  5.51668764e-01, -1.14075987e+00, -2.24819739e-01,
 -9.76885539e-01, -1.10431526e+00,  1.01105006e+00],
[-8.42842478e-01,  1.78486387e+00, -9.78441098e-01,
 -5.65550457e-02, -2.90076965e+00, -1.03333459e+00,
 -1.11578645e+00, -1.10431526e+00, -9.89070710e-01],
[-3.12163881e-02,  8.49865282e-01, -5.98747538e-01,
 -3.84060174e-01,  6.12265743e-01, -1.37626962e-01,
  8.80214574e-02,  9.05538514e-01, -9.89070710e-01],
```

```
## Box plots to understand effect of Standard Scalar
plt.subplots(figsize=(15,5))
plt.subplot(1,2,1)
sns.boxplot(data = X_train)
plt.title('X_train Before Scalin')
plt.subplot(1,2,2)
sns.boxplot(data = X_train_scaled)
plt.title('X_train After Scaling')
plt.show()
```

```
<ipython-input-201-74822d31e19d>:3: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 and will be r
  plt.subplot(1,2,1)
```



Model Training

```
Start coding or generate with AI.
```

Linear Regression

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
```

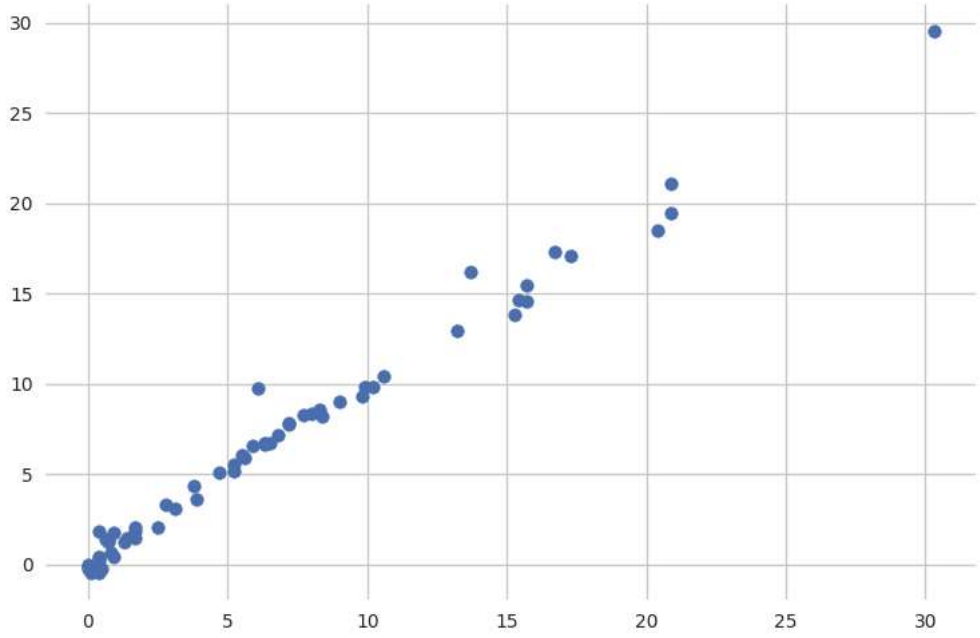
```
linereg = LinearRegression()
linereg.fit(X_train_scaled, y_train)
```

```
y_pred = lineareg.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error: 0.6742766873791607
R-squared: 0.9847657384266951
```

```
plt.scatter(y_test, y_pred)
```

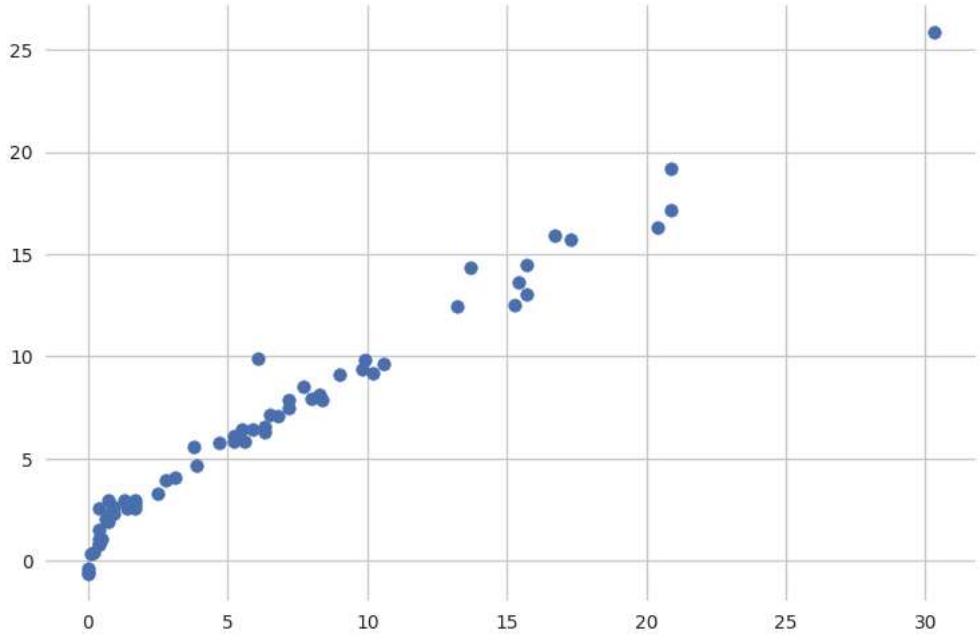
```
<matplotlib.collections.PathCollection at 0x7de246d78b50>
```



Lasso Regression

```
from sklearn.linear_model import Lasso
lassoreg = Lasso()
lassoreg.fit(X_train_scaled, y_train)
y_pred = lassoreg.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
plt.scatter(y_test, y_pred)
```

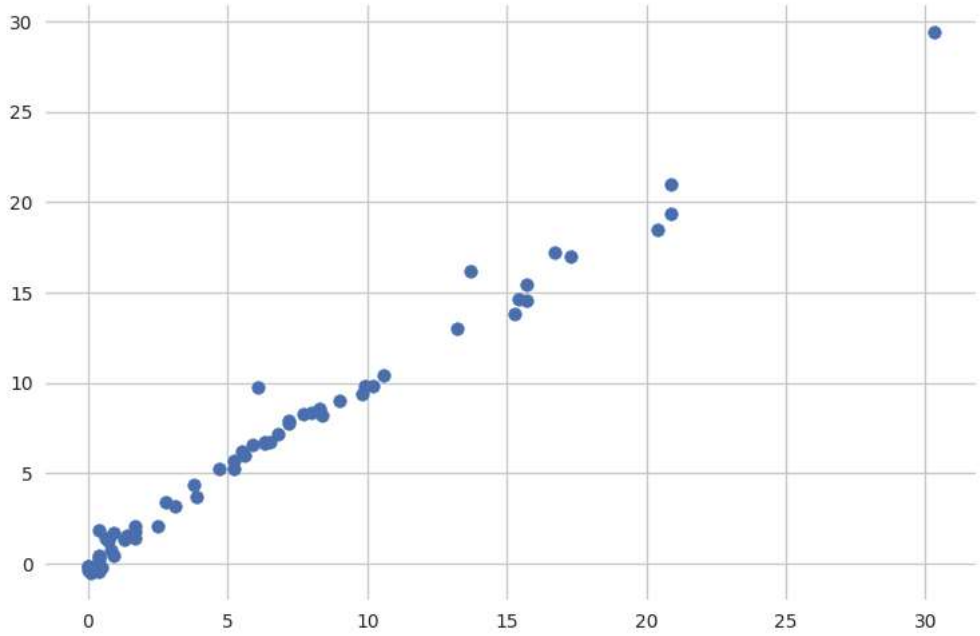
```
Mean Squared Error: 2.2483458918974772
R-squared: 0.9492020263112388
<matplotlib.collections.PathCollection at 0x7de24b812ef0>
```



Ridge Regression

```
from sklearn.linear_model import Ridge
ridgereg = Ridge()
ridgereg.fit(X_train_scaled, y_train)
y_pred = ridgereg.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
plt.scatter(y_test, y_pred)
```

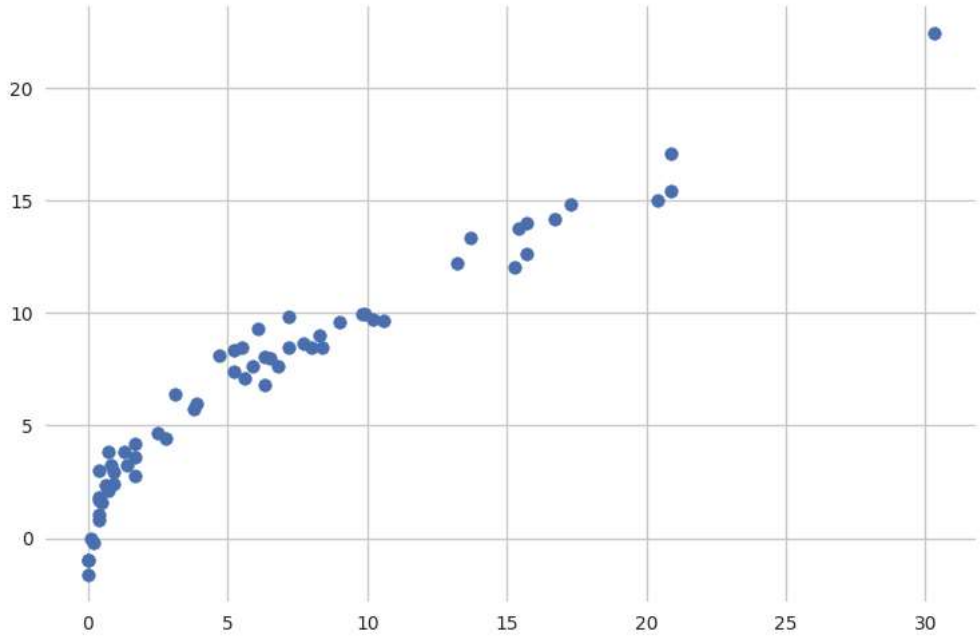
```
Mean Squared Error: 0.6949198918152074
R-squared: 0.9842993364555513
<matplotlib.collections.PathCollection at 0x7de246edd2a0>
```



Elastic Net Regression

```
from sklearn.linear_model import ElasticNet
elasticnetreg = ElasticNet()
elasticnetreg.fit(X_train_scaled, y_train)
y_pred = elasticnetreg.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
plt.scatter(y_test, y_pred)
```

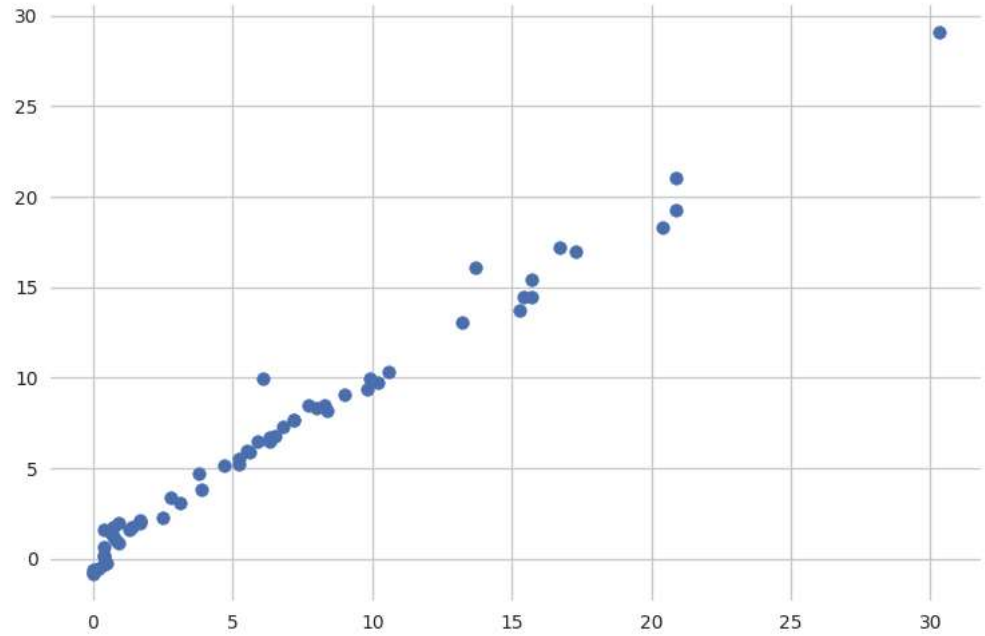
```
Mean Squared Error: 5.5172511010252245
R-squared: 0.8753460589519703
<matplotlib.collections.PathCollection at 0x7de24b27f220>
```



Cross Validations

```
from sklearn.linear_model import LassoCV
# Cross Validation
# Iterative fitting along with regularisation
# by default 5 folds else we can set by cv = n
lassocv = LassoCV(cv =5)
lassocv.fit(X_train_scaled, y_train)
y_pred = lassocv.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
plt.scatter(y_test, y_pred)
```

```
Mean Squared Error: 0.792499555474362
R-squared: 0.9820946715928275
<matplotlib.collections.PathCollection at 0x7de248893ac0>
```




```
lassocv.alphas_
```

```
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])
```

```
lassocv.mse_path_
```