Algerian Forest Fires Dataset

Data Set Information:

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 attribues and 1 output attribue (class) The 244 instances have been classified into fire(138 classes) and not fire (106 classes) classes.

Attribute 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**

Importing Libraries

In [1]:

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

Load the dataset

In [2]:

```
dataset=pd.read_csv('Algerian_forest_fires_dataset_UPDATE.csv' ,header=1)
```

Top 5 rows

In [3]:

```
dataset.head()
```

Out[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	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes
	05	06	2012	27	77	16	0	[illegible]	3	[illegible]	1.2	3.9	0.5	not fire

In [83]:

```
## Shape
df.shape
```

Out[83]:

```
(243, 12)
```

Check data types

In [84]:

```
df.dtypes
```

Out[84]:

```
Temperature      int64
RH               int64
Ws               int64
Rain           float64
FFMC           float64
DMC            float64
DC             float64
ISI            float64
BUI            float64
FWI            float64
Classes          int32
Region           int64
dtype: object
```

Summary of dataset

In [4]:

```
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

Check missing values

In [ ]:

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

The dataset is converted into two sets based on Region from 122th index, we can make a new column based on the Region

1 : "Bejaia Region Dataset"

2 : "Sidi-Bel Abbes Region Dataset"

Add new column with region

In [5]:

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

In [6]:

```
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
```

In [7]:

```
df.head()
```

Out[7]:

	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

In [9]:

```
df[['Region']]=df[['Region']].astype(int)
```

In [10]:

```
df.head()
```

Out[10]:

	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

In [11]:

```
df.tail()
```

Out[11]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	Region
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	30	09	2012	24	64	15	0.2	67.3	3.8	16.5	1.2	4.8	0.5	not fire	1

In [12]:

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

Out[12]:

```
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
```

In [13]:

```
## Removing the null values
df=df.dropna().reset_index(drop=True)
```

In [14]:

```
df.head()
```

Out[14]:

	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

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	Region
4	05	06	2012	27	77	16	0	64.8	3	14.2	1.2	3.9	0.5	not fire	0

In [15]:

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

Out[15]:

```
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
```

In [16]:

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

Out[16]:

	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

In [17]:

```
##remove the 122nd row
df=df.drop(122).reset_index(drop=True)
```

In [18]:

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

Out[18]:

	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

Check column names

In [21]:

```
df.columns
```

Out[21]:

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

In [22]:

```
## fix spaces in columns names
df.columns=df.columns.str.strip()
df.columns
```

Out[22]:

```
Index(['day', 'month', 'year', 'Temperature', 'RH', 'Ws', 'Rain', 'FFMC',
```

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

In [23]:

```
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    int32 
dtypes: int32(1), object(14)
memory usage: 27.7+ KB
```

Changes the required columns as integer data type

In [24]:

```
df.columns
```

Out[24]:

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

In [25]:

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

In [26]:

```
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    int32
```

```
 14  Region       243 non-null    int32
dtypes: int32(7), object(8)
memory usage: 22.0+ KB
```

In [27]:

```
df.head()
```

Out[27]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	Region
0	1	6	2012	29	57	18	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	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	1.7	0	not fire	0
4	5	6	2012	27	77	16	0	64.8	3	14.2	1.2	3.9	0.5	not fire	0

Changing the other columns to float data datatype

In [28]:

```
objects=[features for features in df.columns if df[features].dtypes=='O']
```

In [29]:

```
objects
```

Out[29]:

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

In [30]:

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

In [31]:

```
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    int32  
dtypes: float64(7), int32(7), object(1)
memory usage: 22.0+ KB
```

In [32]:

```
objects
```

Out[32]:

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

In [33]:

```
df.describe()
```

Out[33]:

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

In [34]:

```
df.head()
```

Out[34]:

	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

In [35]:

```
## Let ave the cleaned dataset
df.to_csv('Algerian_forest_fires_cleaned_dataset.csv',index=False)
```

Exploratory Data Analysis

In [36]:

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

In [37]:

```
df_copy.head()
```

Out[37]:

	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

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	Region
4	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0.5	not fire	0

In [38]:

```python
## categories in classes
df_copy['Classes'].value_counts()
```

Out[38]:

```
fire             131
not fire         101
fire               4
fire               2
not fire           2
not fire           1
not fire           1
not fire           1
Name: Classes, dtype: int64
```

In [39]:

```python
## Encoding of the categories in classes
df_copy['Classes']=np.where(df_copy['Classes'].str.contains('not fire'),0,1)
```

In [40]:

```python
df_copy.head()
```

Out[40]:

	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

In [41]:

```python
df_copy.tail()
```

Out[41]:

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	Region
238	30	65	14	0.0	85.4	16.0	44.5	4.5	16.9	6.5	1	1
239	28	87	15	4.4	41.1	6.5	8.0	0.1	6.2	0.0	0	1
240	27	87	29	0.5	45.9	3.5	7.9	0.4	3.4	0.2	0	1
241	24	54	18	0.1	79.7	4.3	15.2	1.7	5.1	0.7	0	1
242	24	64	15	0.2	67.3	3.8	16.5	1.2	4.8	0.5	0	1

In [42]:

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

Out[42]:

```
1    137
0    106
Name: Classes, dtype: int64
```

In [43]:

```python
## Plot desnity plot for all features
```

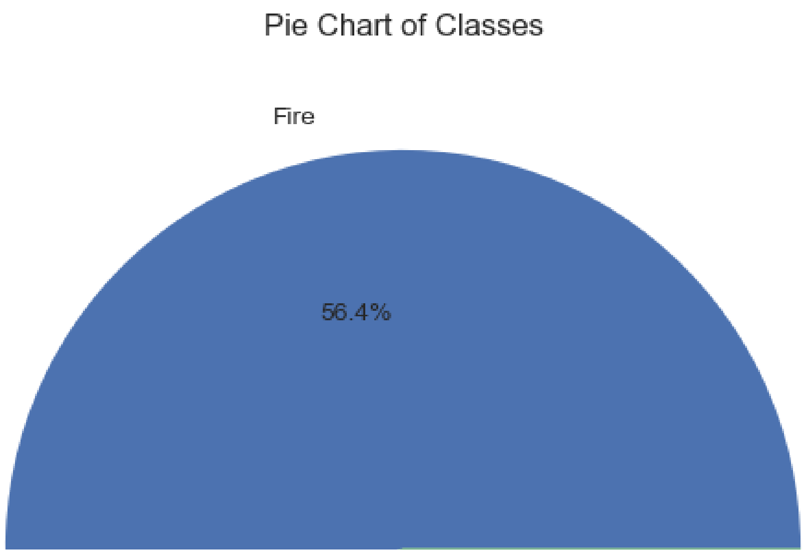
```
plt.style.use('seaborn')
df_copy.hist(bins=50,figsize=(20,15))
plt.show()
```

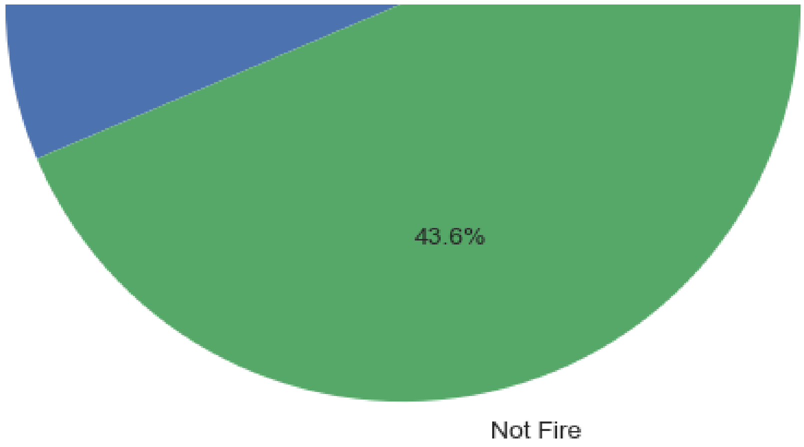
In [44]:

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

In [45]:

```
# plotting piechart
classlabels=["Fire","Not Fire"]
plt.figure(figsize=(12,7))
plt.pie(percentage,labels=classlabels,autopct='%1.1f%%')
plt.title("Pie Chart of Classes")
plt.show()
```





Correlation

In [46]:

```
df_copy.corr()
```

Out[46]:

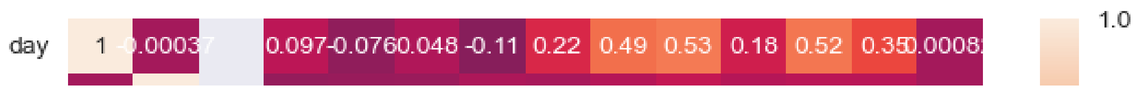
	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classe
Temperature	1.000000	-0.651400	-0.284510	-0.326492	0.676568	0.485687	0.376284	0.603871	0.459789	0.566670	0.51601
RH	-0.651400	1.000000	0.244048	0.222356	-0.644873	-0.408519	-0.226941	-0.686667	-0.353841	-0.580957	0.43216
Ws	-0.284510	0.244048	1.000000	0.171506	-0.166548	-0.000721	0.079135	0.008532	0.031438	0.032368	0.06996
Rain	-0.326492	0.222356	0.171506	1.000000	-0.543906	-0.288773	-0.298023	-0.347484	-0.299852	-0.324422	0.37909
FFMC	0.676568	-0.644873	-0.166548	-0.543906	1.000000	0.603608	0.507397	0.740007	0.592011	0.691132	0.76949
DMC	0.485687	-0.408519	-0.000721	-0.288773	0.603608	1.000000	0.875925	0.680454	0.982248	0.875864	0.58565
DC	0.376284	-0.226941	0.079135	-0.298023	0.507397	0.875925	1.000000	0.508643	0.941988	0.739521	0.51112
ISI	0.603871	-0.686667	0.008532	-0.347484	0.740007	0.680454	0.508643	1.000000	0.644093	0.922895	0.73519
BUI	0.459789	-0.353841	0.031438	-0.299852	0.592011	0.982248	0.941988	0.644093	1.000000	0.857973	0.58663
FWI	0.566670	-0.580957	0.032368	-0.324422	0.691132	0.875864	0.739521	0.922895	0.857973	1.000000	0.71921
Classes	0.516015	-0.432161	-0.069964	-0.379097	0.769492	0.585658	0.511123	0.735197	0.586639	0.719216	1.00000
Region	0.269555	-0.402682	-0.181160	-0.040013	0.222241	0.192089	-0.078734	0.263197	0.089408	0.197102	0.16234

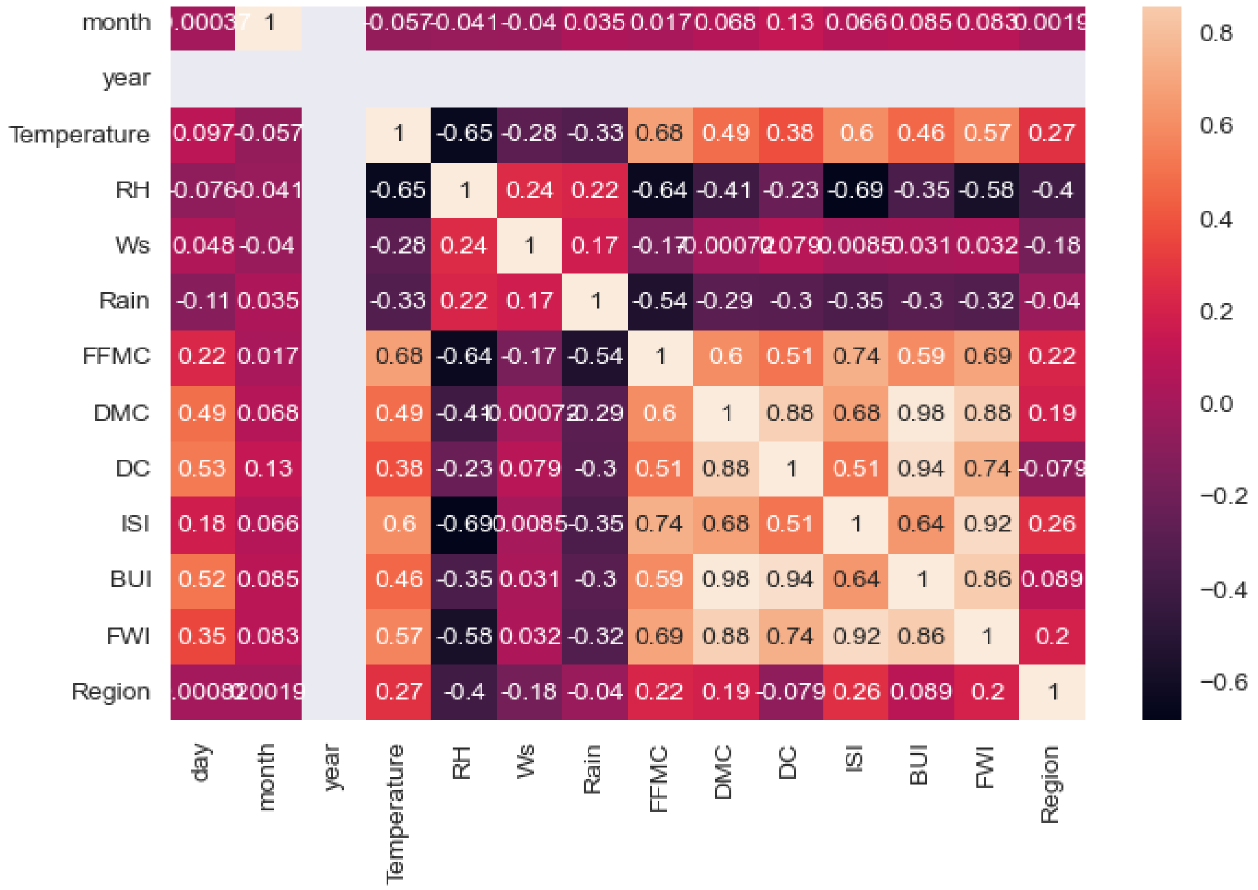
In [47]:

```
sns.heatmap(df.corr(),annot=True)
```

Out[47]:

<AxesSubplot:>



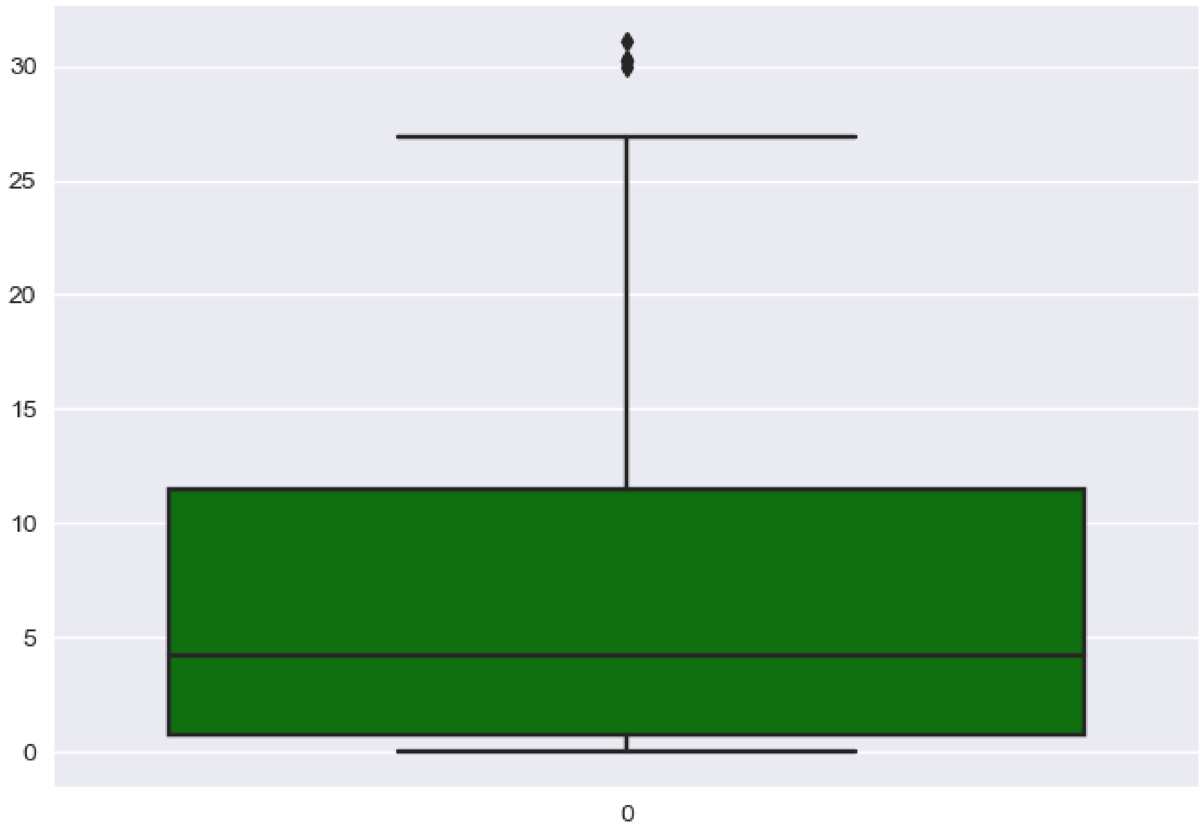


In [48]:

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

Out[48]:

```
<AxesSubplot:>
```



In [49]:

```
df.head()
```

Out[49]:

	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

In [50]:

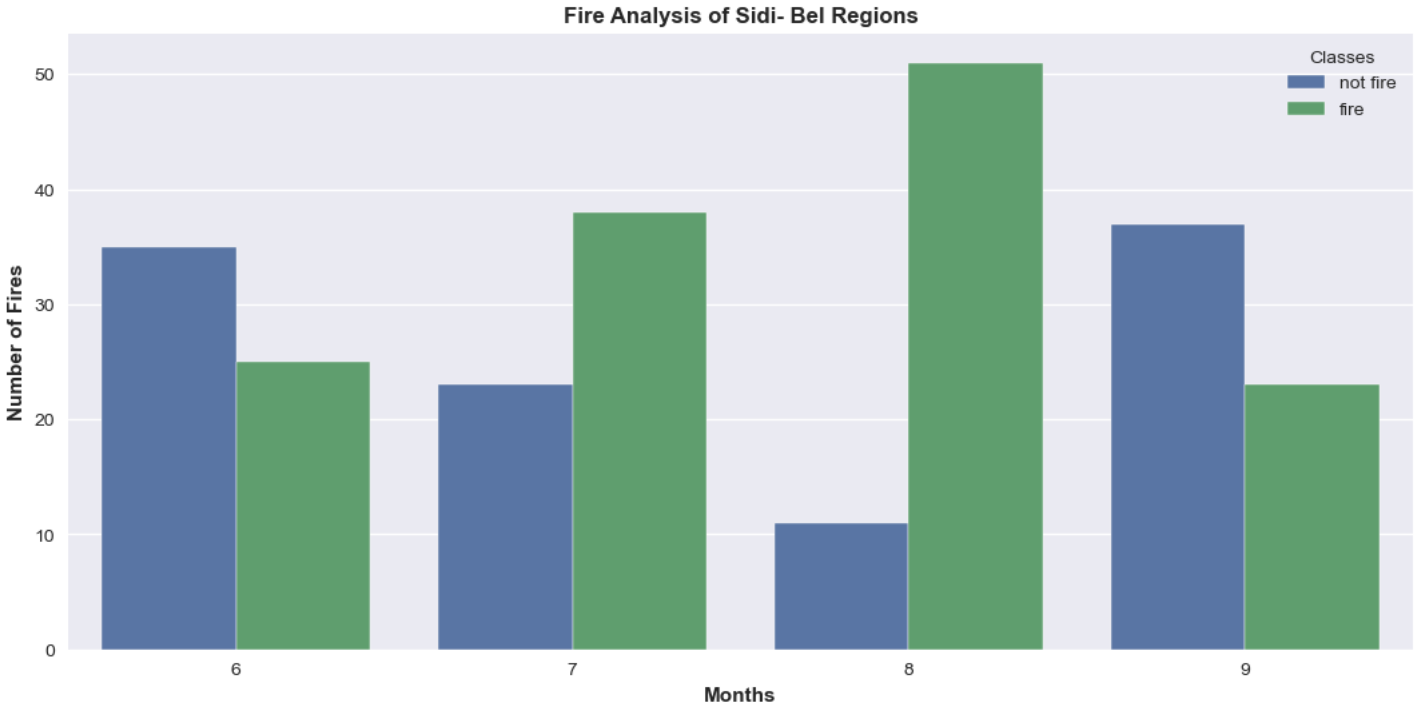
```
df['Classes']=np.where(df['Classes'].str.contains('not fire'),'not fire','fire')
```

In [51]:

```
## Monthly Fire Analysis
dftemp=df.loc[df['Region']==1]
plt.subplots(figsize=(13,6))
sns.set_style('whitegrid')
sns.countplot(x='month',hue='Classes',data=df)
plt.ylabel('Number of Fires',weight='bold')
plt.xlabel('Months',weight='bold')
plt.title("Fire Analysis of Sidi- Bel Regions",weight='bold')
```

Out[51]:

```
Text(0.5, 1.0, 'Fire Analysis of Sidi- Bel Regions')
```

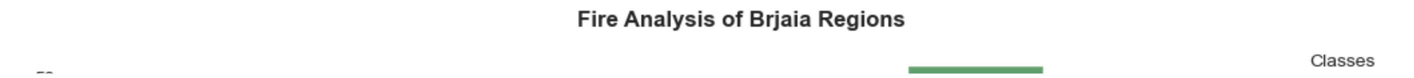


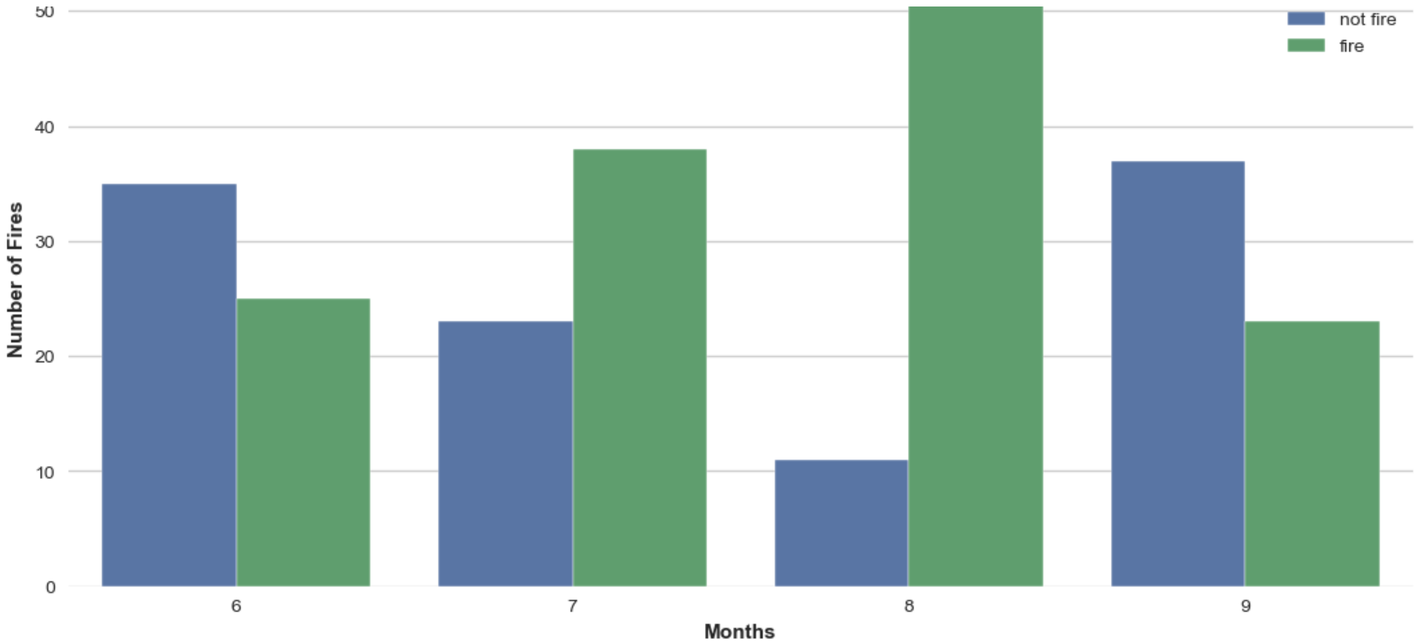
In [52]:

```
## Monthly Fire Analysis
dftemp=df.loc[df['Region']==0]
plt.subplots(figsize=(13,6))
sns.set_style('whitegrid')
sns.countplot(x='month',hue='Classes',data=df)
plt.ylabel('Number of Fires',weight='bold')
plt.xlabel('Months',weight='bold')
plt.title("Fire Analysis of Brjaia Regions",weight='bold')
```

Out[52]:

```
Text(0.5, 1.0, 'Fire Analysis of Brjaia Regions')
```





Its observed that August and September had the most number of forest fires for both regions. And from the above plot of months, we can understand few things

Most of the fires happened in August and very high Fires happened in only 3 months - June, July and August.

Less Fires was on September

In [53]:

```
df=pd.read_csv('Algerian_forest_fires_cleaned_dataset.csv')
```

In [54]:

```
df.head()
```

Out[54]:

	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

In [55]:

```
df.columns
```

Out[55]:

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

In [56]:

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

In [57]:

```
df.head()
```

Out[57]:

	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

In [58]:

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

Out[58]:

```
fire            131
not fire        101
fire              4
fire              2
not fire          2
not fire          1
not fire          1
not fire          1
Name: Classes, dtype: int64
```

In [59]:

```
## Encoding
df['Classes']=np.where(df['Classes'].str.contains("not fire"),0,1)
```

In [60]:

```
df.tail()
```

Out[60]:

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	Region
238	30	65	14	0.0	85.4	16.0	44.5	4.5	16.9	6.5	1	1
239	28	87	15	4.4	41.1	6.5	8.0	0.1	6.2	0.0	0	1
240	27	87	29	0.5	45.9	3.5	7.9	0.4	3.4	0.2	0	1
241	24	54	18	0.1	79.7	4.3	15.2	1.7	5.1	0.7	0	1
242	24	64	15	0.2	67.3	3.8	16.5	1.2	4.8	0.5	0	1

In [61]:

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

Out[61]:

```
1    137
0    106
Name: Classes, dtype: int64
```

In [62]:

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

In [63]:

```
X.head()
```

Out[63]:

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	Classes	Region

0	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	Classes	Region
	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

In [64]:

```
y
```

Out[64]:

```
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
Name: FWI, Length: 243, dtype: float64
```

In [65]:

```
#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)
```

In [66]:

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

Out[66]:

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

In [67]:

```
## Feature Selection based on correlaltion
X_train.corr()
```

Out[67]:

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	Classes	Regio
Temperature	1.000000	-0.656095	-0.305977	-0.317512	0.694768	0.498173	0.390684	0.629848	0.473609	0.542141	0.25454
RH	-0.656095	1.000000	0.225736	0.241656	-0.653023	-0.414601	-0.236078	-0.717804	-0.362317	-0.456876	-0.39466
Ws	-0.305977	0.225736	1.000000	0.251932	-0.190076	0.000379	0.096576	-0.023558	0.035633	-0.082570	-0.19996
Rain	-0.317512	0.241656	0.251932	1.000000	-0.545491	-0.289754	-0.302341	-0.345707	-0.300964	-0.369357	-0.05902
FFMC	0.694768	-0.653023	-0.190076	-0.545491	1.000000	0.620807	0.524101	0.750799	0.607210	0.781259	0.24951
DMC	0.498173	-0.414601	0.000379	-0.289754	0.620807	1.000000	0.868647	0.685656	0.983175	0.617273	0.21258
DC	0.390684	-0.236078	0.096576	-0.302341	0.524101	0.868647	1.000000	0.513701	0.942414	0.543581	-0.06083
ISI	0.629848	-0.717804	-0.023558	-0.345707	0.750799	0.685656	0.513701	1.000000	0.643818	0.742977	0.29644
BUI	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	Classes	Regio
BUI	0.473609	0.362317	0.035633	0.300964	0.607210	0.983175	0.942414	0.643818	1.000000	0.612239	0.11489
Classes	0.542141	-0.456876	-0.082570	-0.369357	0.781259	0.617273	0.543581	0.742977	0.612239	1.000000	0.18883
Region	0.254549	-0.394665	-0.199969	-0.059022	0.249514	0.212582	-0.060838	0.296441	0.114897	0.188837	1.00000

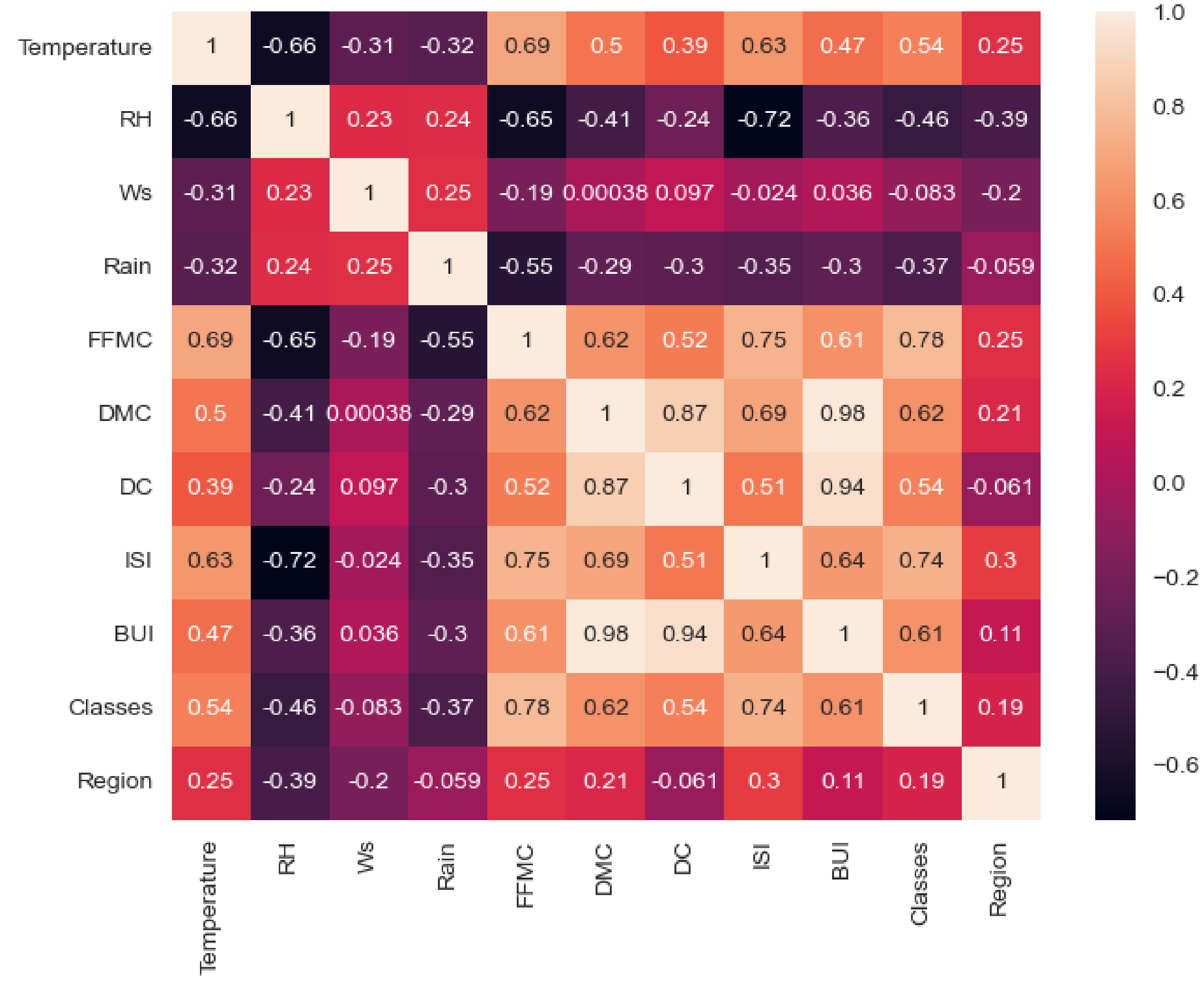
Feature Selection

In [70]:

```
## Check for multicollinearity
plt.figure(figsize=(8,6))
corr=X_train.corr()
sns.heatmap(corr,annot=True)
```

Out[70]:

<AxesSubplot:>



In [71]:

```
X_train.corr()
```

Out[71]:

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	Classes	Regio
Temperature	1.000000	-0.656095	-0.305977	-0.317512	0.694768	0.498173	0.390684	0.629848	0.473609	0.542141	0.25454
RH	-0.656095	1.000000	0.225736	0.241656	-0.653023	-0.414601	-0.236078	-0.717804	-0.362317	-0.456876	-0.39466

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	Classes	Regio
Ws	-0.305977	0.225736	1.000000	0.251932	0.190076	0.000379	0.096576	0.023558	0.035633	0.082570	0.19996
Rain	-0.317512	0.241656	0.251932	1.000000	-0.545491	-0.289754	-0.302341	-0.345707	-0.300964	-0.369357	-0.05902
FFMC	0.694768	-0.653023	-0.190076	-0.545491	1.000000	0.620807	0.524101	0.750799	0.607210	0.781259	0.24951
DMC	0.498173	-0.414601	0.000379	-0.289754	0.620807	1.000000	0.868647	0.685656	0.983175	0.617273	0.21258
DC	0.390684	-0.236078	0.096576	-0.302341	0.524101	0.868647	1.000000	0.513701	0.942414	0.543581	-0.06083
ISI	0.629848	-0.717804	-0.023558	-0.345707	0.750799	0.685656	0.513701	1.000000	0.643818	0.742977	0.29644
BUI	0.473609	-0.362317	0.035633	-0.300964	0.607210	0.983175	0.942414	0.643818	1.000000	0.612239	0.11489
Classes	0.542141	-0.456876	-0.082570	-0.369357	0.781259	0.617273	0.543581	0.742977	0.612239	1.000000	0.18883
Region	0.254549	-0.394665	-0.199969	-0.059022	0.249514	0.212582	-0.060838	0.296441	0.114897	0.188837	1.00000

In [72]:

```
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
```

In [73]:

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

In [74]:

```
corr_features
```

Out[74]:

```
{'BUI', 'DC'}
```

In [75]:

```
## 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)
X_train.shape,X_test.shape
```

Out[75]:

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

Feature Scaling Or Standardization

In [76]:

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

In [77]:

```
X_train_scaled
```

Out[77]:

```
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]])
```

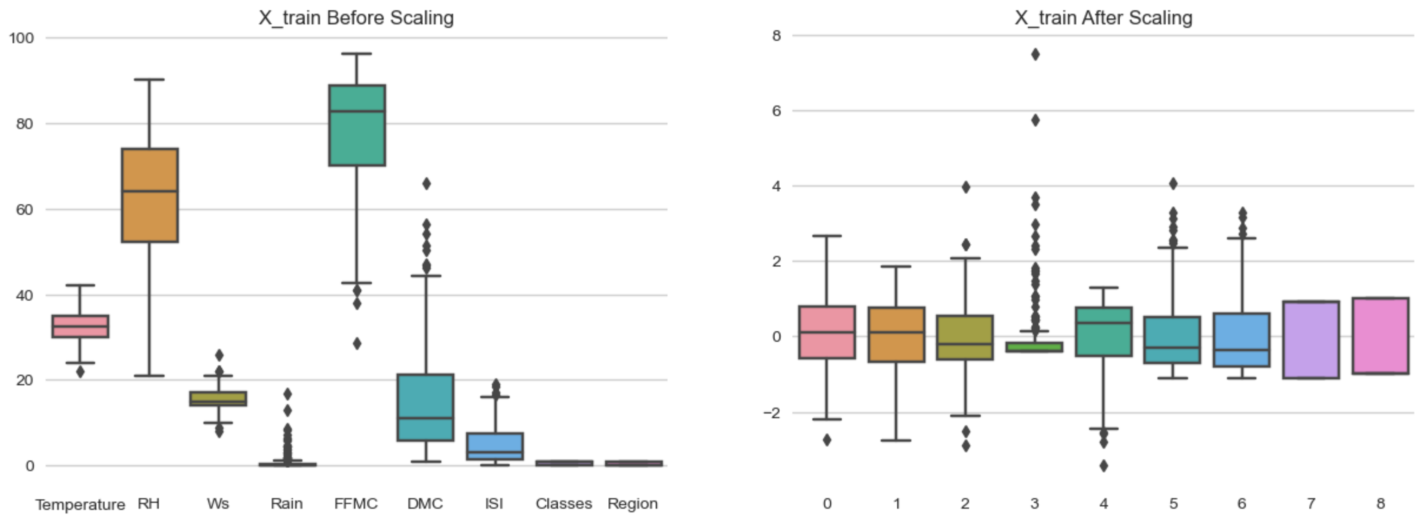
Box Plots To understand Effect Of Standard Scaler

In [78]:

```
plt.subplots(figsize=(15, 5))
plt.subplot(1, 2, 1)
sns.boxplot(data=X_train)
plt.title('X_train Before Scaling')
plt.subplot(1, 2, 2)
sns.boxplot(data=X_train_scaled)
plt.title('X_train After Scaling')
```

Out[78]:

```
Text(0.5, 1.0, 'X_train After Scaling')
```



Linear Regression Model

Linear regression is a statistical approach used to model the relationship between a dependent variable and one or more independent variables, by fitting a linear equation to the observed data.

In [79]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
linreg=LinearRegression()
linreg.fit(X_train_scaled,y_train)
y_pred=linreg.predict(X_test_scaled)
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 0.5468236465249986
R2 Score 0.9847657384266951
```

Lasso Regression

Feature Selection

In [80]:

```
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
lasso=Lasso()
lasso.fit(X_train_scaled,y_train)
y_pred=lasso.predict(X_test_scaled)
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 1.1331759949144087
R2 Score 0.9492020263112388
```

Ridge Regression model

Reducing Overfitting thats why we use Ridge Regression

In [81]:

```
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
ridge=Ridge()
ridge.fit(X_train_scaled,y_train)
y_pred=ridge.predict(X_test_scaled)
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 0.5642305340105719
R2 Score 0.9842993364555512
```

Elasticnet Regression Model

In [82]:

```
from sklearn.linear_model import ElasticNet
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
elastic=ElasticNet()
elastic.fit(X_train_scaled,y_train)
y_pred=elastic.predict(X_test_scaled)
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 1.8822353634896
R2 Score 0.8753460589519703
```

Import Pickle

In [ ]:

```
import pickle
pickle.dump(scaler,open('scaler.pkl','wb'))
pickle.dump(ridge,open('ridge.pkl','wb'))
```