Importing basic libraries

In [1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

Loading the dataset

In [2]:
```python
df = pd.read_csv('Algerian_forest_fires_dataset.csv')
df.head()
```

Out[2]:

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

In [3]:
```python
# Getting the shape of the dataset
df.shape
```

Out[3]:
```
(247, 14)
```

In [4]:
```python
df.columns
```

Out[4]:
```
Index(['day', 'month', 'year', 'Temperature', ' RH', ' Ws', 'Rain ', 'FFMC',
       'DMC', 'DC', 'ISI', 'BUI', 'FWI', 'Classes  '],
      dtype='object')
```

In [5]:
```python
# Dropping 122 and 123th row since it has nan values for all the features.

df.drop([122,123],inplace=True)

# Resetting the index and dropping the index column
df.reset_index(inplace=True)
df.drop('index',axis=1,inplace=True)
```

In [6]:
```python
df.loc[:122,'region'] = 'bejaia'
df.loc[122:,'region'] = 'Sidi-Bel Abbes'
```

In [7]:
```python
df.columns
```

Out[7]:
```
Index(['day', 'month', 'year', 'Temperature', ' RH', ' Ws', 'Rain ', 'FFMC',
       'DMC', 'DC', 'ISI', 'BUI', 'FWI', 'Classes  ', 'region'],
      dtype='object')
```

In [8]:
```python
# Stripping the names of the columns for removing unnecceasary space in the names

df.columns = [i.strip() for i in df.columns]
df.columns
```

Out[8]:
```
Index(['day', 'month', 'year', 'Temperature', 'RH', 'Ws', 'Rain', 'FFMC',
       'DMC', 'DC', 'ISI', 'BUI', 'FWI', 'Classes', 'region'],
      dtype='object')
```

In [9]:
```python
# Checking for null values
df.isnull().sum()
```

Out[9]:
```
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        1
region         0
dtype: int64
```

In [10]:
```python
# Dropping the null values row
df.dropna(inplace=True)
```

In [11]:
```python
df.dtypes
```

Out[11]:
```
day            object
month          object
year           object
Temperature    object
RH             object
Ws             object
Rain           object
FFMC           object
DMC            object
DC             object
ISI            object
BUI            object
FWI            object
Classes        object
region         object
dtype: object
```

In [12]:
```python
df.head()
```

Out[12]:

	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	bejaia
1	2	6	2012	29	61	13	1.3	64.4	4.1	7.6	1	3.9	0.4	not fire	bejaia
2	3	6	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	not fire	bejaia
3	4	6	2012	25	89	13	2.5	28.6	1.3	6.9	0	1.7	0	not fire	bejaia
4	5	6	2012	27	77	16	0	64.8	3	14.2	1.2	3.9	0.5	not fire	bejaia

In [13]:
```python
df['Classes'].unique()
```

Out[13]:
```
array(['not fire   ', 'fire   ', 'fire', 'fire ', 'not fire', 'not fire ',
       'Classes  ', 'not fire     ', 'not fire    '], dtype=object)
```

In [14]:
```python
df['Classes'] = df['Classes'].str.strip()
df['Classes'].unique()
```

Out[14]:
```
array(['not fire', 'fire', 'Classes'], dtype=object)
```

In [15]:
```python
df['Classes'] = df['Classes'].replace({'Classes':np.nan})
```

In [16]:
```python
df['Classes'].unique()
```

Out[16]:
```
array(['not fire', 'fire', nan], dtype=object)
```

In [17]:
```python
## Unique values of the Classes feature
df['Classes'].unique()
```

Out[17]:
```
array(['not fire', 'fire', nan], dtype=object)
```

In [18]:
```python
## Handling Categorical feature Classes

df['Classes']=df['Classes'].map({'not fire' :0, 'fire':1})
df.head()
```

Out[18]:

	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	0.0	bejaia
1	2	6	2012	29	61	13	1.3	64.4	4.1	7.6	1	3.9	0.4	0.0	bejaia
2	3	6	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	0.0	bejaia
3	4	6	2012	25	89	13	2.5	28.6	1.3	6.9	0	1.7	0	0.0	bejaia
4	5	6	2012	27	77	16	0	64.8	3	14.2	1.2	3.9	0.5	0.0	bejaia

In [19]:
```python
# We can replace the null value of the classes feature with mode since this technique is appropriate for categorical features

df['Classes'].mode()[0]
```

Out[19]:
```
1.0
```

In [20]:
```python
df['Classes'] = df['Classes'].fillna(df['Classes'].mode()[0])
```

In [21]:
```python
df.isnull().sum()
```

Out[21]:
```
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 [22]:
```python
df.iloc[121:123,:]
```

Out[22]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	region
121	30	9	2012	25	78	14	1.4	45	1.9	7.5	0.2	2.4	0.1	0.0	bejaia
122	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	1.0	Sidi-Bel Abbes

In [23]:
```python
df.drop([122],inplace=True)
```

In [24]:
```python
## Replacing day, month and year feature with date feature
df['date'] = pd.to_datetime(df[['day','month','year']])
df.drop(['day','month','year'],axis=1,inplace=True)
```

In [25]:
```python
df.columns
```

Out[25]:
```
Index(['Temperature', 'RH', 'Ws', 'Rain', 'FFMC', 'DMC', 'DC', 'ISI', 'BUI',
       'FWI', 'Classes', 'region', 'date'],
      dtype='object')
```

In [26]:
```python
df.head()
```

Out[26]:

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	region	date
0	29	57	18	0	65.7	3.4	7.6	1.3	3.4	0.5	0.0	bejaia	2012-06-01
1	29	61	13	1.3	64.4	4.1	7.6	1	3.9	0.4	0.0	bejaia	2012-06-02
2	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	0.0	bejaia	2012-06-03
3	25	89	13	2.5	28.6	1.3	6.9	0	1.7	0	0.0	bejaia	2012-06-04
4	27	77	16	0	64.8	3	14.2	1.2	3.9	0.5	0.0	bejaia	2012-06-05

In [27]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 243 entries, 0 to 244
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Temperature  243 non-null    object
 1   RH           243 non-null    object
 2   Ws           243 non-null    object
 3   Rain         243 non-null    object
 4   FFMC         243 non-null    object
 5   DMC          243 non-null    object
 6   DC           243 non-null    object
 7   ISI          243 non-null    object
 8   BUI          243 non-null    object
 9   FWI          243 non-null    object
 10  Classes      243 non-null    float64
 11  region       243 non-null    object
 12  date         243 non-null    datetime64[ns]
dtypes: datetime64[ns](1), float64(1), object(11)
memory usage: 26.6+ KB
```

In [58]:
```python
## Changing the datatype of the features

df['Temperature'] = df['Temperature'].astype(int)
df['RH'] = df['RH'].astype(int)
df['Ws'] = df['Ws'].astype(int)
df[['Rain','FFMC','DC','ISI','BUI']] = df[['Rain','FFMC','DC','ISI','BUI']].astype(float)
```

In [31]:
```python
## Checking the memory usage of the dataset

df.memory_usage()
```

Out[31]:
```
Index          1944
Temperature     972
RH              972
Ws              972
Rain           1944
FFMC           1944
DMC            1944
DC             1944
ISI            1944
BUI            1944
FWI            1944
Classes        1944
region         1944
date           1944
dtype: int64
```

In [37]:
```python
### Numerical and Categorical Columns

num_fea = [fea for fea in df.columns if df[fea].dtype != 'O']
cat_fea = [fea for fea in df.columns if df[fea].dtype == 'O']
```

In [38]:
```python
print(num_fea, len(num_fea))

print(cat_fea,len(cat_fea))
```

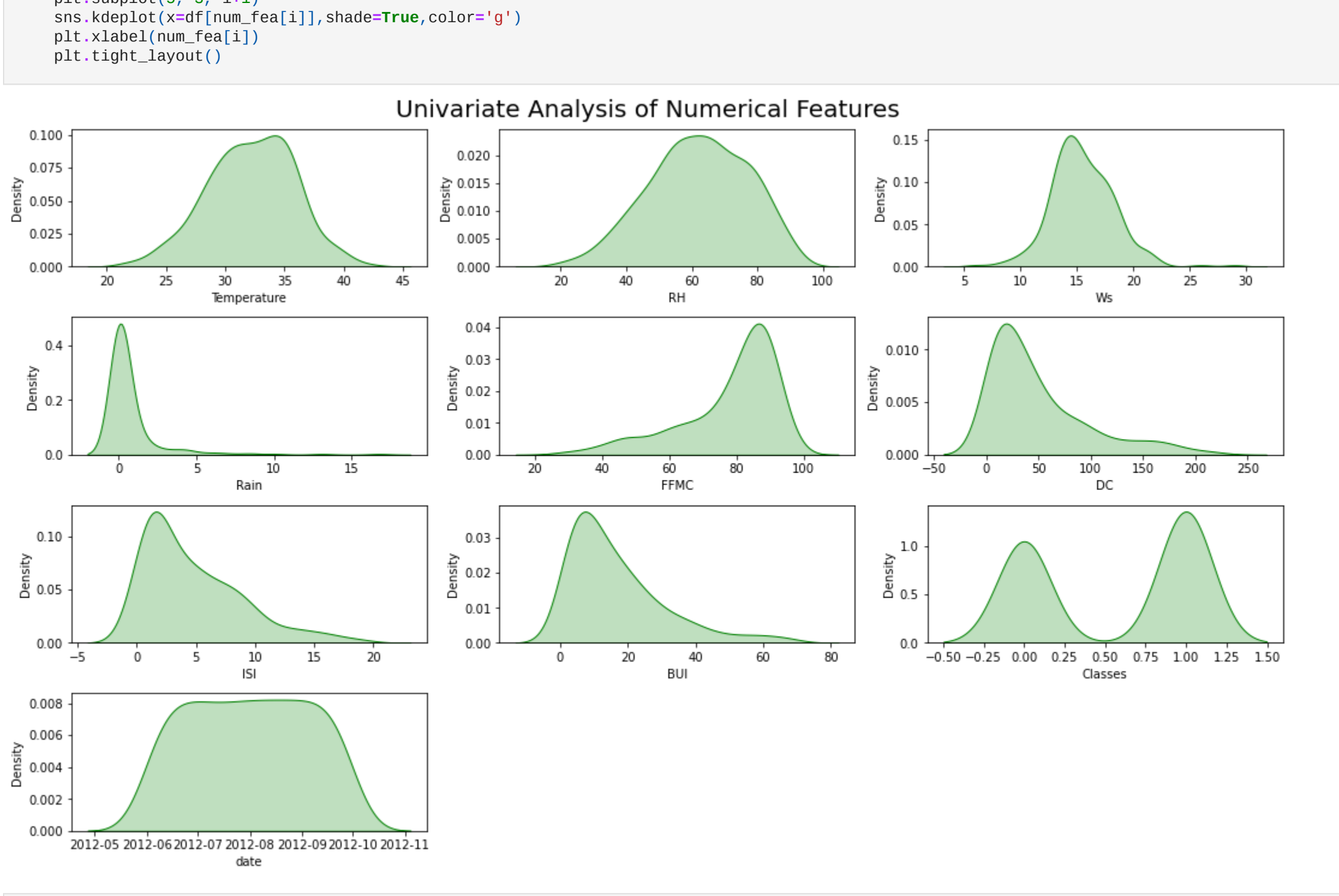
```
['Temperature', 'RH', 'Ws', 'Rain', 'FFMC', 'DC', 'ISI', 'BUI', 'Classes', 'date'] 10
['DMC', 'FWI', 'region'] 3
```

Univariate Analysis

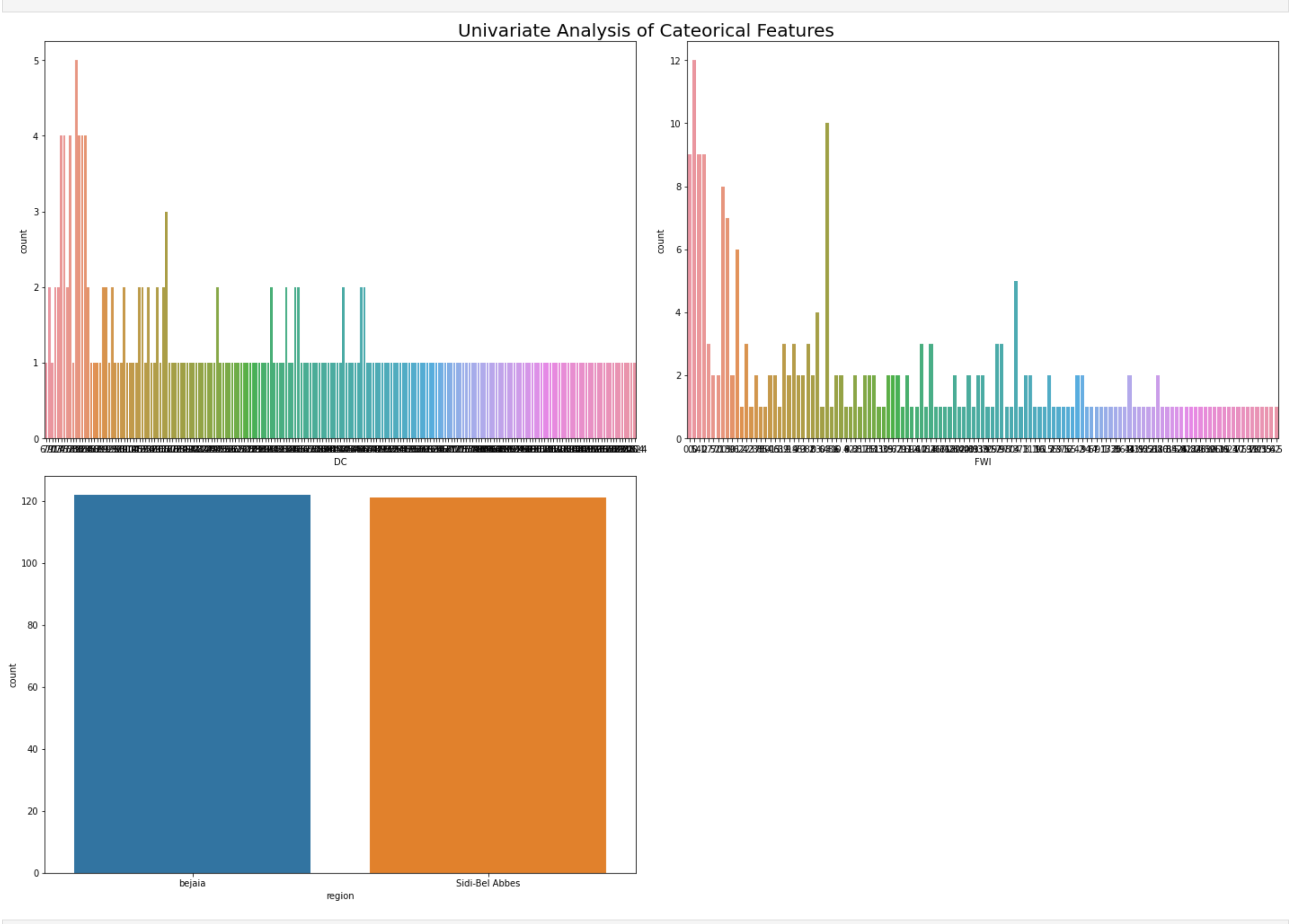
The term univariate analysis refers to the analysis of one variable prefix 'uni' means 'one'. The purpose of univariate analysis is to understand the distribution of values for a single variable.

In [41]:
```python
## Numerical Features
plt.figure(figsize=(15,11))
plt.suptitle('Univariate Analysis of Numerical Features', fontsize=20, fontweight=23)

for i in range(0, len(num_fea)):
    plt.subplot(5, 3, i+1)
    sns.kdeplot(x=df[num_fea[i]],shade=True,color='g')
    plt.xlabel(num_fea[i])
    plt.tight_layout()
```



In [42]:
```python
# Categorical Features
plt.figure(figsize=(20,14))
plt.suptitle('Univariate Analysis of Cateorical Features', fontsize=20, fontweight=32)
cat1 = ['DC','FWI','region']
for i in range(0, len(cat1)):
    plt.subplot(2,2, i+1)
    sns.countplot(x=df[cat1[i]],data=df)
    plt.xlabel(cat1[i])
    plt.tight_layout()
```



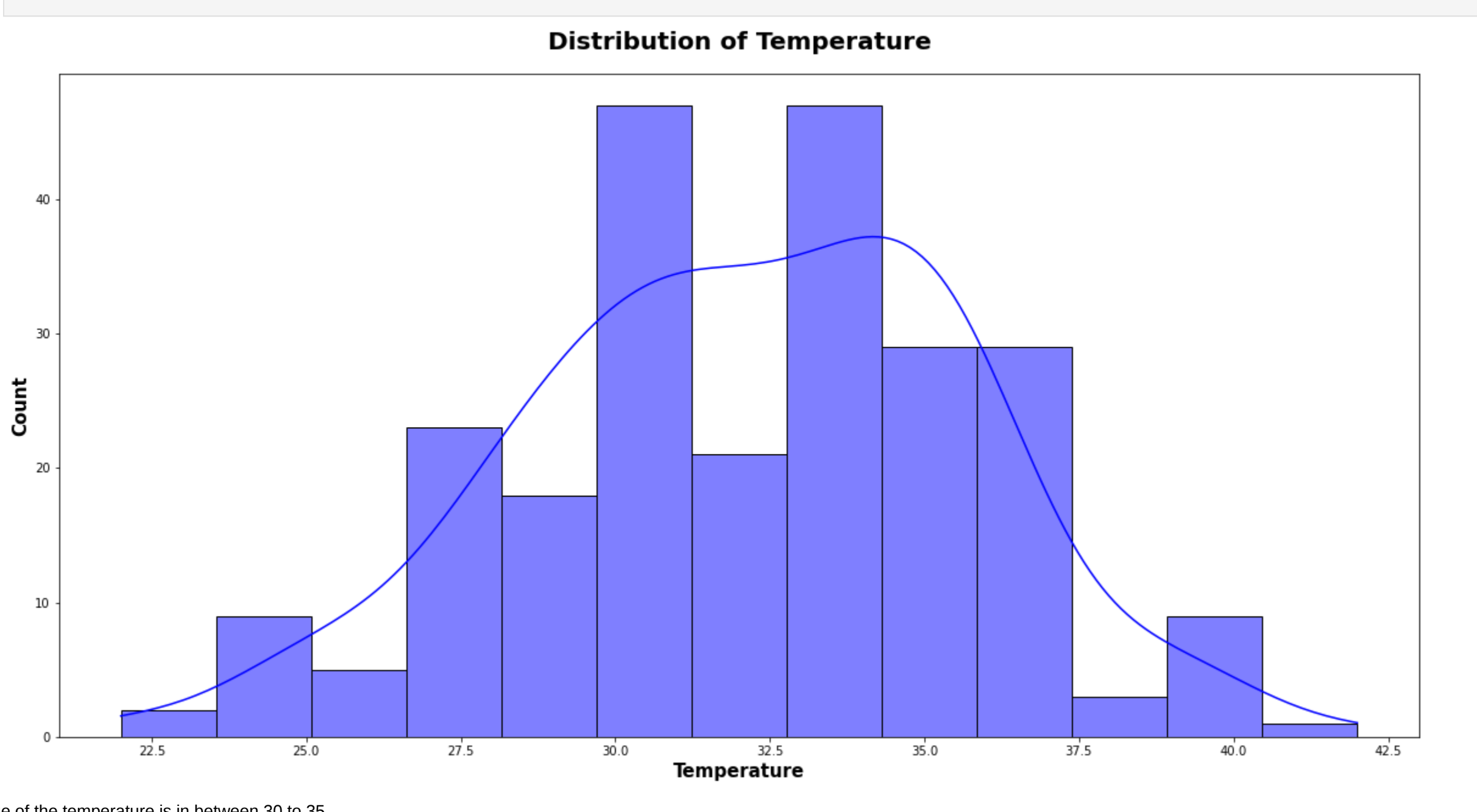
In [44]:
```python
# Which area has most of the time fire happen

sns.barplot(x='region',y='Classes',data=df)
```

Out[44]:
```
<AxesSubplot:xlabel='region', ylabel='Classes'>
```

Sidi-Bel Abbes region has most of the fire happen.

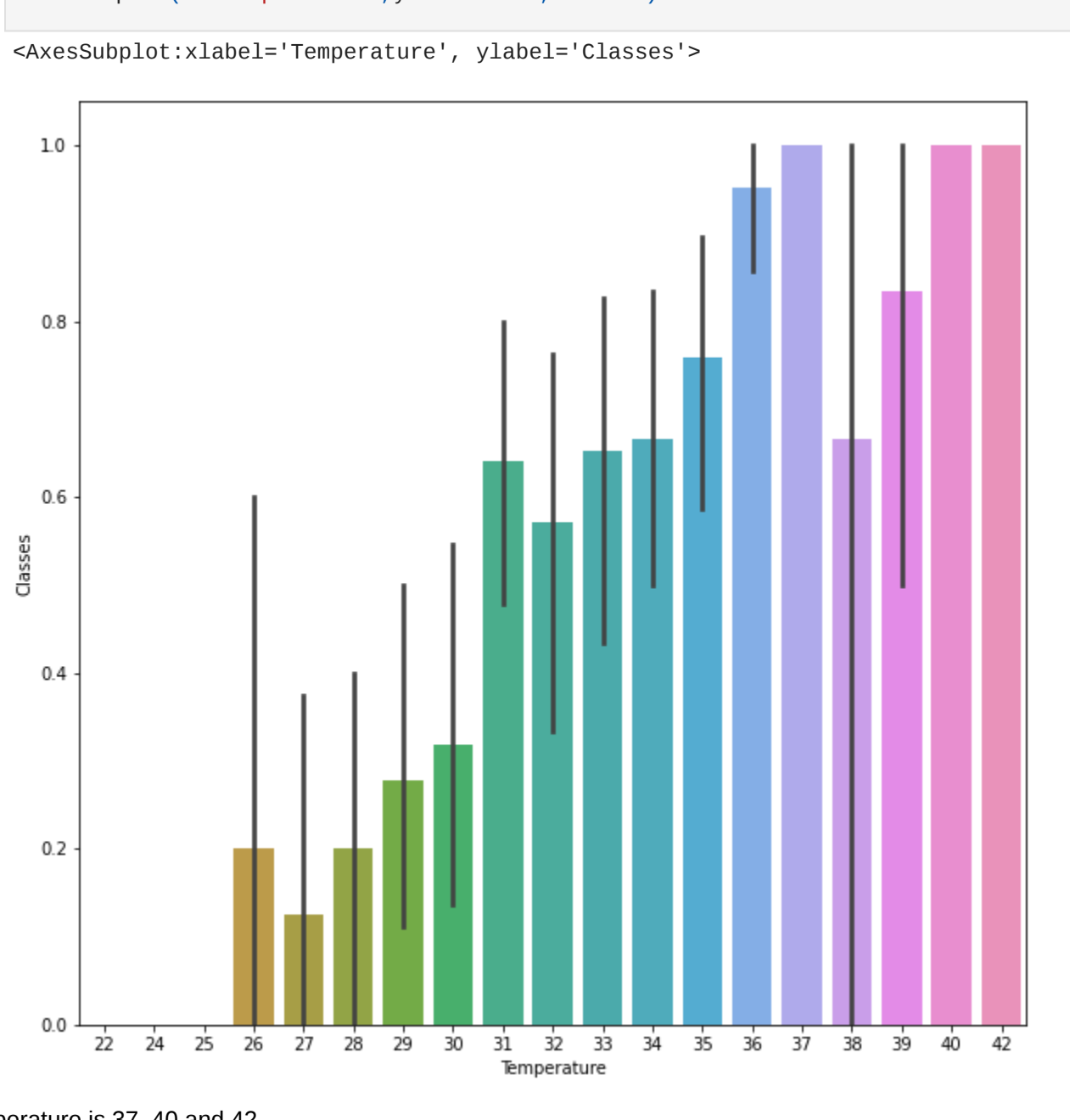
In [45]:
```python
plt.subplots(figsize=(20,10))
sns.histplot('Distribution of Temperature',x=df['Temperature'],color='b',kde=True)
plt.title('Distribution of Temperature', weight='bold', fontsize=20,pad=20)
plt.xlabel('Temperature',weight='bold',fontsize=15)
plt.ylabel('Count',weight='bold',fontsize=15)
plt.show()
```



Highest range of the temperature is in between 30 to 35.

In [47]:
```python
plt.figure(figsize=(10,10))
sns.barplot(x='Temperature',y='Classes',data=df)
```

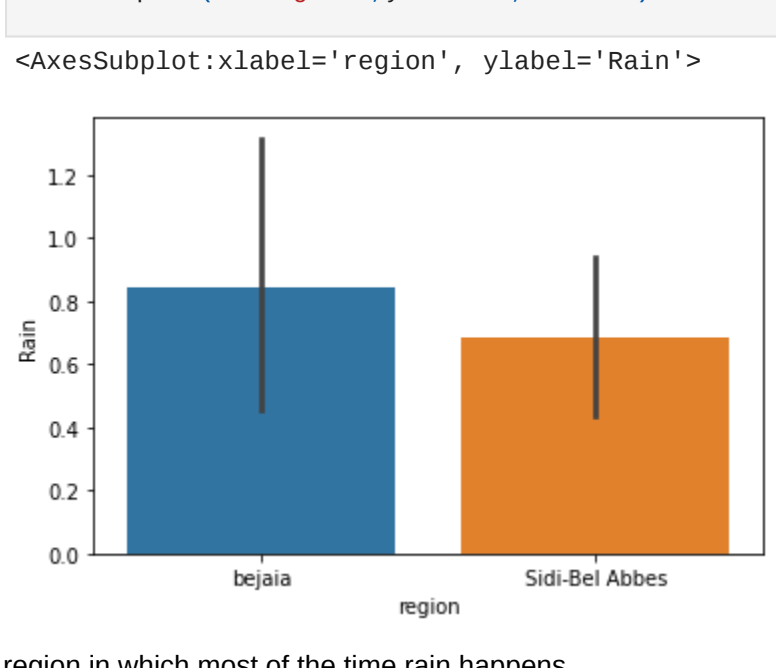
Out[47]:
```
<AxesSubplot:xlabel='Temperature', ylabel='Classes'>
```



Highest temperature is 37, 40 and 42.

In [48]:
```python
# which Region has most of the time rain
sns.barplot(x='region',y='Rain',data=df)
```

Out[48]:
```
<AxesSubplot:xlabel='region', ylabel='Rain'>
```



Bejaia is the region in which most of the time rain happens.

Multivariate Analysis

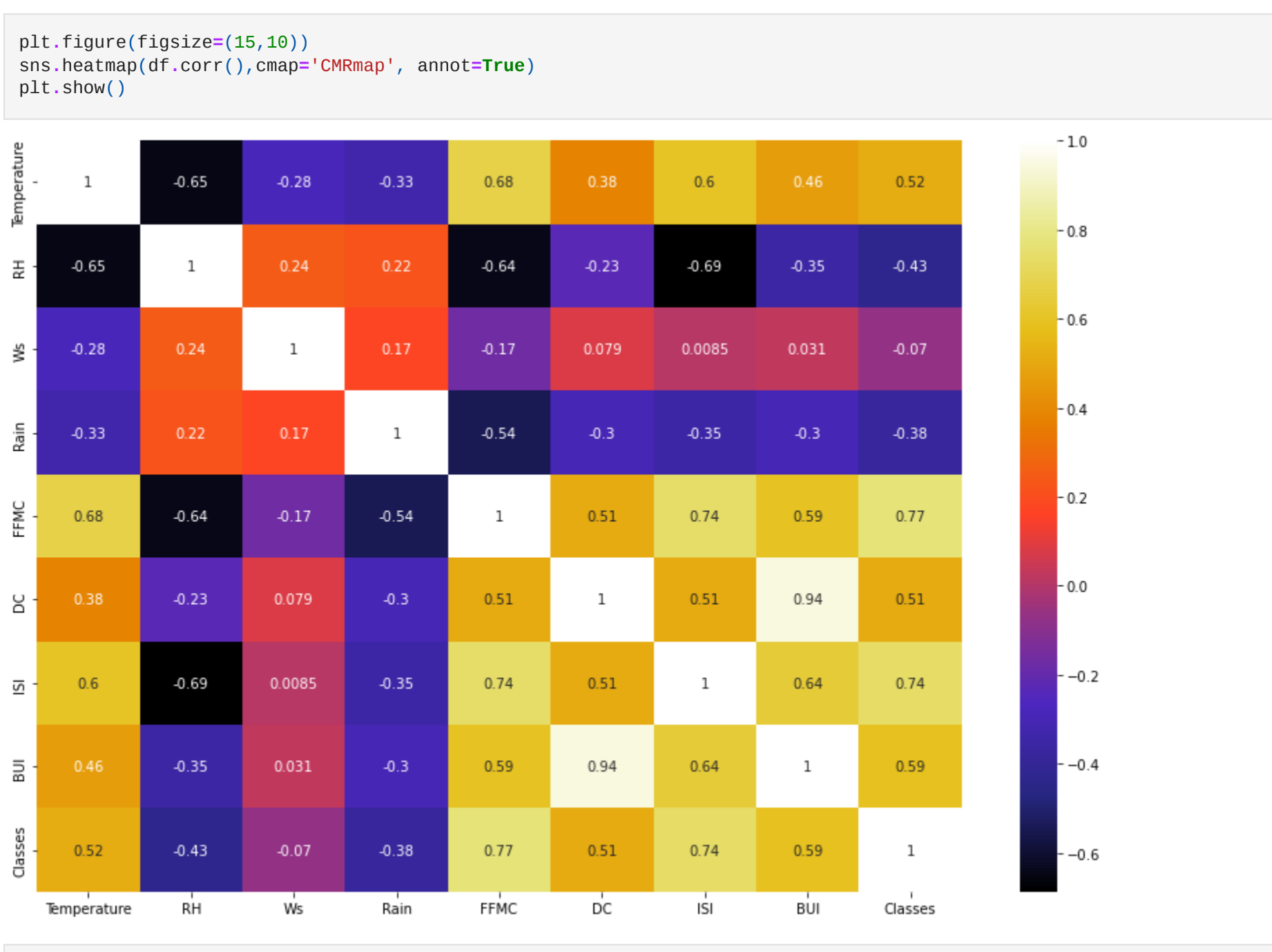
Multivariate analysis is the analysis of more than one variable.

In [50]:
```python
df.corr()
```

Out[50]:

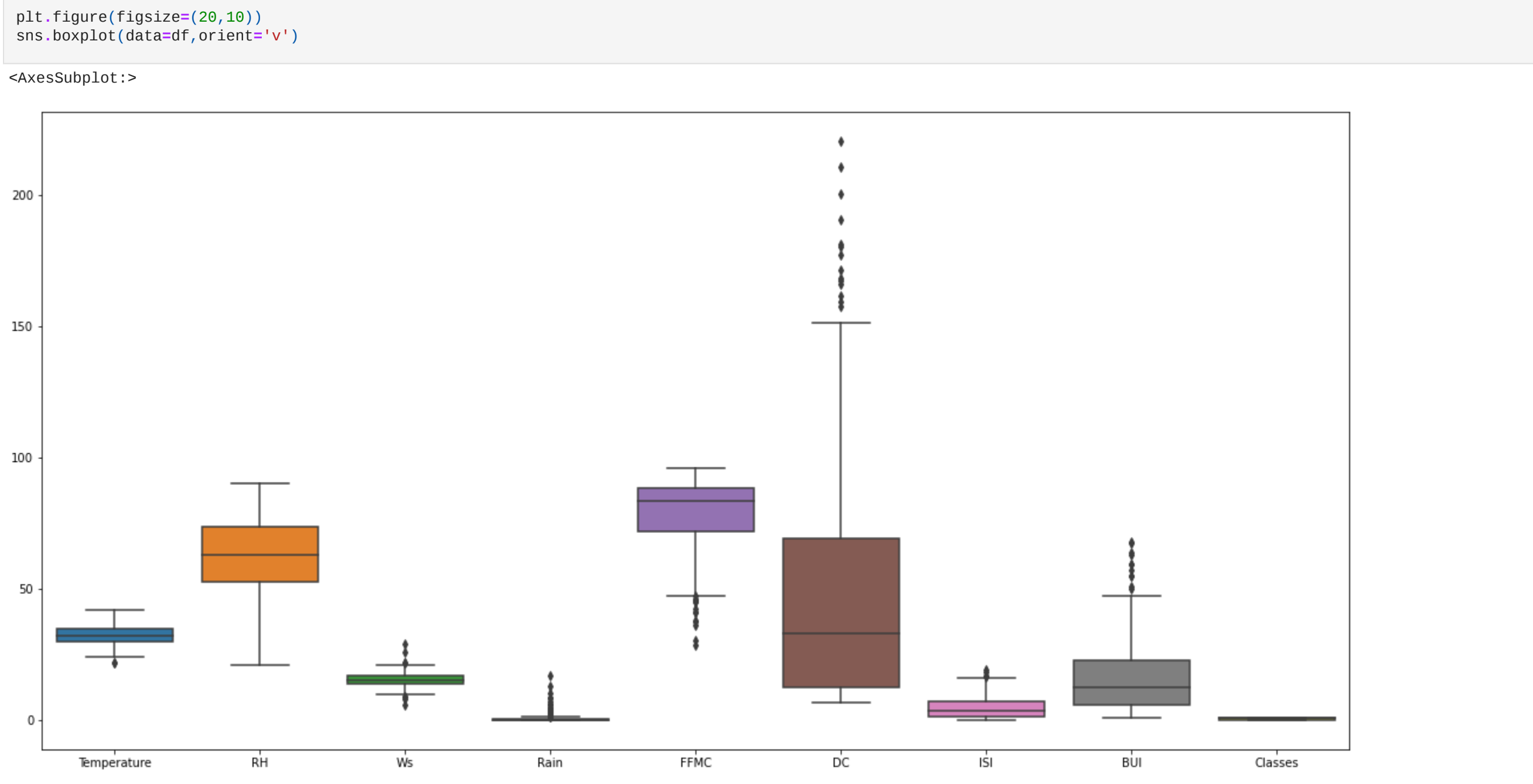
	Temperature	RH	Ws	Rain	FFMC	DC	ISI	BUI	Classes
Temperature	1.000000	-0.651400	-0.284510	-0.326492	0.676568	0.376284	0.603871	0.459789	0.516015
RH	-0.651400	1.000000	0.244048	0.222356	-0.644873	-0.226941	-0.686667	-0.353841	-0.432161
Ws	-0.284510	0.244048	1.000000	0.171506	-0.166548	0.079135	0.008532	0.031438	-0.069964
Rain	-0.326492	0.222356	0.171506	1.000000	-0.543906	-0.298023	-0.347484	-0.299852	-0.379097
FFMC	0.676568	-0.644873	-0.166548	-0.543906	1.000000	0.507397	0.740007	0.592011	0.769492
DC	0.376284	-0.226941	0.079135	-0.298023	0.507397	1.000000	0.508643	0.941988	0.511123
ISI	0.603871	-0.686667	0.008532	-0.347484	0.740007	0.508643	1.000000	0.644093	0.735197
BUI	0.459789	-0.353841	0.031438	-0.299852	0.592011	0.941988	0.644093	1.000000	0.586639
Classes	0.516015	-0.432161	-0.069964	-0.379097	0.769492	0.511123	0.735197	0.586639	1.000000

In [51]:
```python
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(),cmap='CMRmap', annot=True)
plt.show()
```



In [53]:
```python
plt.figure(figsize=(20,10))
sns.boxplot(data=df,orient='v')
```

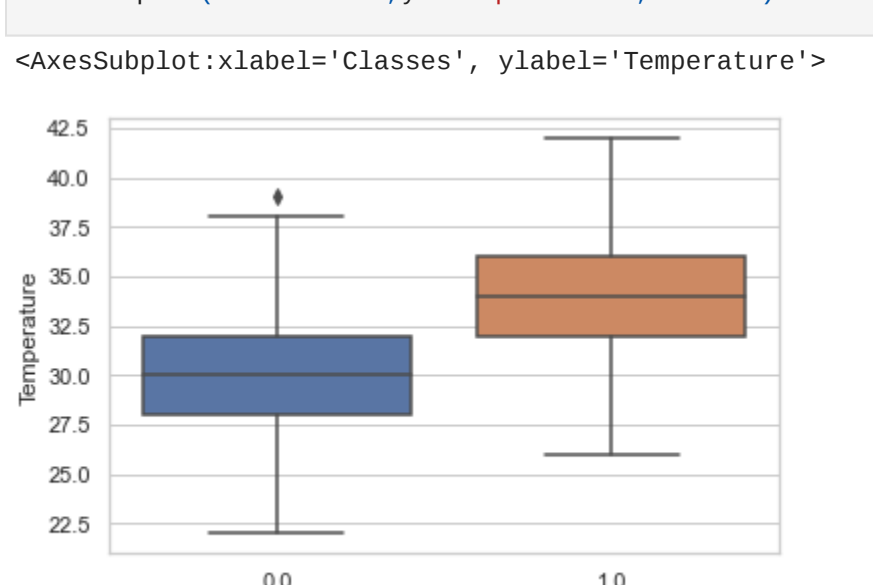
Out[53]:
```
<AxesSubplot:>
```



Rain, ws, FFMC, DC and BUI has many outliers.

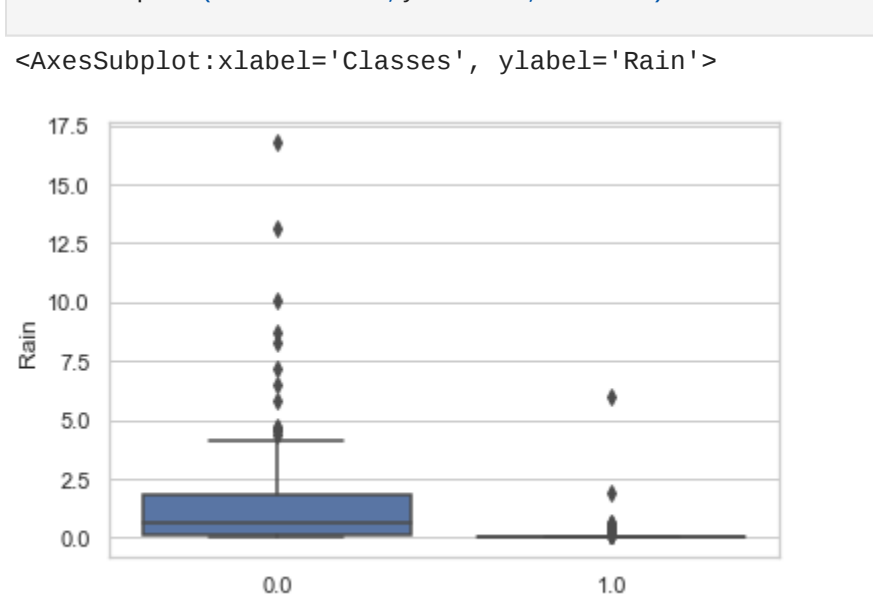
In [55]:
```python
sns.set(style='whitegrid')
sns.boxplot(x='Classes',y='Temperature',data=df)
```

Out[55]:
```
<AxesSubplot:xlabel='Classes', ylabel='Temperature'>
```



In [54]:
```python
sns.set(style='whitegrid')
sns.boxplot(x='Classes',y='Rain',data=df)
```

Out[54]:
```
<AxesSubplot:xlabel='Classes', ylabel='Rain'>
```



In [56]:
```python
## Statistical Analysis
```

In [57]:
```python
df.describe()
```

Out[57]:

	Temperature	RH	Ws	Rain	FFMC	DC	ISI	BUI	Classes
count	243.000000	243.000000	243.000000	243.000000	243.000000	243.000000	243.000000	243.000000	243.000000
mean	32.152263	62.041152	15.493827	0.762963	77.842387	49.430864	4.742387	16.690535	0.563786
std	3.628039	14.828160	2.811385	2.003207	14.349641	47.665606	4.154234	14.228421	0.496938
min	22.000000	21.000000	6.000000	0.000000	28.600000	6.900000	0.000000	1.100000	0.000000
25%	30.000000	52.500000	14.000000	0.000000	71.850000	12.350000	1.400000	6.000000	0.000000
50%	32.000000	63.000000	15.000000	0.000000	83.300000	33.100000	3.500000	12.400000	1.000000
75%	35.000000	73.500000	17.000000	0.500000	88.300000	69.100000	7.250000	22.650000	1.000000
max	42.000000	90.000000	29.000000	16.800000	96.000000	220.400000	19.000000	68.000000	1.000000

In [ ]: