Importing the Libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import warnings


warnings.filterwarnings("ignore")

%matplotlib inline
```

Data Reading And Cleaning

In [2]:
```
df = pd.read_csv('Algerian_forest_fires_dataset.csv',header=1)
df.head()
```

Out[2]:

	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

In [3]:
```
# Drop an  row
df.drop([122,123],inplace=True)
df.reset_index(inplace=True)
df.drop('index',axis=1,inplace=True)
```

In [4]:
```
df.loc[:122, 'region'] = 'bejaia'
df.loc[122:, 'region'] = 'Sidi-Bel Abbes'
```

In [5]:
```
# Stripping the names of the columns

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

Out[5]:
```
Index(['day', 'month', 'year', 'Temperature', 'RH', 'Ws', 'Rain', 'FFMC',
       'DMC', 'DC', 'ISI', 'BUI', 'FWI', 'Classes', 'region'],
      dtype='object')
```

In [6]:
```
# Stripping the Classes Features data

df.Classes = df.Classes.str.strip()
df['Classes'].unique()
```

Out[6]:
```
array(['not fire', 'fire', nan], dtype=object)
```

Changing The DataTypes of the Columns

In [7]:
```


df['day']=df['day'].astype(int)
df['month']=df['month'].astype(int)
df['year']=df['year'].astype(int)
df['Temperature']=df['Temperature'].astype(int)
df['RH']=df['RH'].astype(int)
df['Rain']=df['Rain'].astype(float)
df['FFMC']=df['FFMC'].astype(float)
df['DMC']=df['DMC'].astype(float)
df['BUI']=df['BUI'].astype(float)
df['ISI']=df['ISI'].astype(float)
df['Ws']=df['Ws'].astype(float)



df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   day          244 non-null    int32  
 1   month        244 non-null    int32  
 2   year         244 non-null    int32  
 3   Temperature  244 non-null    int32  
 4   RH           244 non-null    int32  
 5   Ws           244 non-null    float64
 6   Rain         244 non-null    float64
 7   FFMC         244 non-null    float64
 8   DMC          244 non-null    float64
 9   DC           244 non-null    object 
 10  ISI          244 non-null    float64
 11  BUI          244 non-null    float64
 12  FWI          244 non-null    object 
 13  Classes      243 non-null    object 
 14  region       244 non-null    object 
dtypes: float64(6), int32(5), object(4)
memory usage: 24.0+ KB
```

Checking the null value

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
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
```

- We got One Null Value

In [9]:
```
## Unique Value of Classes feature

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

Out[9]:
```
array(['not fire', 'fire', nan], dtype=object)
```

In [10]:
```
## Handling Categorical Feature Classes

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

Out[10]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	re
0	1	6	2012	29	57	18.0	0.0	65.7	3.4	7.6	1.3	3.4	0.5	0.0	t
1	2	6	2012	29	61	13.0	1.3	64.4	4.1	7.6	1.0	3.9	0.4	0.0	t
2	3	6	2012	26	82	22.0	13.1	47.1	2.5	7.1	0.3	2.7	0.1	0.0	t
3	4	6	2012	25	89	13.0	2.5	28.6	1.3	6.9	0.0	1.7	0	0.0	t
4	5	6	2012	27	77	16.0	0.0	64.8	3.0	14.2	1.2	3.9	0.5	0.0	t

Focus on Replacing Null Value

The best Way of Replacing Null Value by using mode

In [11]:
```
df['Classes'].mode() [0]
```

Out[11]:
```
1.0
```

In [12]:
```
df['Classes']=df['Classes'].fillna(df['Classes'].mode()[0])
```

In [13]:
```
df.isnull().sum()
```

Out[13]:
```
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
```

- Now We have Zero Null Value

In [14]:
```
df['Classes'].unique()
```

Out[14]:
```
array([0., 1.])
```

Replacing the 'day','month','year' features with 'date' feature

In [15]:
```
df['date']=pd.to_datetime(df[['day','month','year']])
df.drop(['day','month','year'],axis=1,inplace=True)
```

In [16]:
```
df
```

Out[16]:

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	region	date
0	29	57	18.0	0.0	65.7	3.4	7.6	1.3	3.4	0.5	0.0	bejaia	2012-06-01
1	29	61	13.0	1.3	64.4	4.1	7.6	1.0	3.9	0.4	0.0	bejaia	2012-06-02
2	26	82	22.0	13.1	47.1	2.5	7.1	0.3	2.7	0.1	0.0	bejaia	2012-06-03
3	25	89	13.0	2.5	28.6	1.3	6.9	0.0	1.7	0	0.0	bejaia	2012-06-04
4	27	77	16.0	0.0	64.8	3.0	14.2	1.2	3.9	0.5	0.0	bejaia	2012-06-05
...	...	...	...	...	...	...	...	...	...	...	...	...	...
239	30	65	14.0	0.0	85.4	16.0	44.5	4.5	16.9	6.5	1.0	Sidi-Bel Abbes	2012-09-26
240	28	87	15.0	4.4	41.1	6.5	8	0.1	6.2	0	0.0	Sidi-Bel Abbes	2012-09-27
241	27	87	29.0	0.5	45.9	3.5	7.9	0.4	3.4	0.2	0.0	Sidi-Bel Abbes	2012-09-28
242	24	54	18.0	0.1	79.7	4.3	15.2	1.7	5.1	0.7	0.0	Sidi-Bel Abbes	2012-09-29
243	24	64	15.0	0.2	67.3	3.8	16.5	1.2	4.8	0.5	0.0	Sidi-Bel Abbes	2012-09-30

244 rows × 13 columns

Observation after cleaning the data

In [17]:
```
# Shape of the data

df.shape
```

Out[17]:
```
(244, 13)
```

** We have 243 rows and 12 Columns

In [18]:
```
## Columns of the dataset

df.columns
```

Out[18]:
```
Index(['Temperature', 'RH', 'Ws', 'Rain', 'FFMC', 'DMC', 'DC', 'ISI', 'BUI',
       'FWI', 'Classes', 'region', 'date'],
      dtype='object')
```

In [19]:
```
## Information of the dataset

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Temperature  244 non-null    int32
 1   RH           244 non-null    int32
 2   Ws           244 non-null    float64
 3   Rain         244 non-null    float64
 4   FFMC         244 non-null    float64
 5   DMC          244 non-null    float64
 6   DC           244 non-null    object
 7   ISI          244 non-null    float64
 8   BUI          244 non-null    float64
 9   FWI          244 non-null    object
 10  Classes      244 non-null    float64
 11  region       244 non-null    object
 12  date         244 non-null    datetime64[ns]
dtypes: datetime64[ns](1), float64(7), int32(2), object(3)
memory usage: 23.0+ KB
```

In [20]:
```
## Checking for null values

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

Out[20]:
```
Temperature    0
RH             0
Ws             0
Rain           0
FFMC           0
DMC            0
DC             0
ISI            0
BUI            0
FWI            0
Classes        0
region         0
date           0
dtype: int64
```

** We got Zero null value

In [21]:
```
## Checking the usage of the memory by the dataset

df.memory_usage()
```

Out[21]:
```
Index           128
Temperature     976
RH              976
Ws             1952
Rain           1952
FFMC           1952
DMC            1952
DC             1952
ISI            1952
BUI            1952
FWI            1952
Classes        1952
region         1952
date           1952
dtype: int64
```

Numerical and Categorical Columns

In [22]:
```
# define numerical & categorical columns
numeric_features = [feature for feature in df.columns if df[feature].dtype != 'O'
categorical_features = [feature for feature in df.columns if df[feature].dtype ==

# print columns
print('We have {} numerical features : {}'.format(len(numeric_features), numeric_
print('\nWe have {} categorical features : {}'.format(len(categorical_features),
```

```
We have 10 numerical features : ['Temperature', 'RH', 'Ws', 'Rain', 'FFMC', 'DM
C', 'ISI', 'BUI', 'Classes', 'date']

We have 3 categorical features : ['DC', 'FWI', 'region']
```

Feature Information

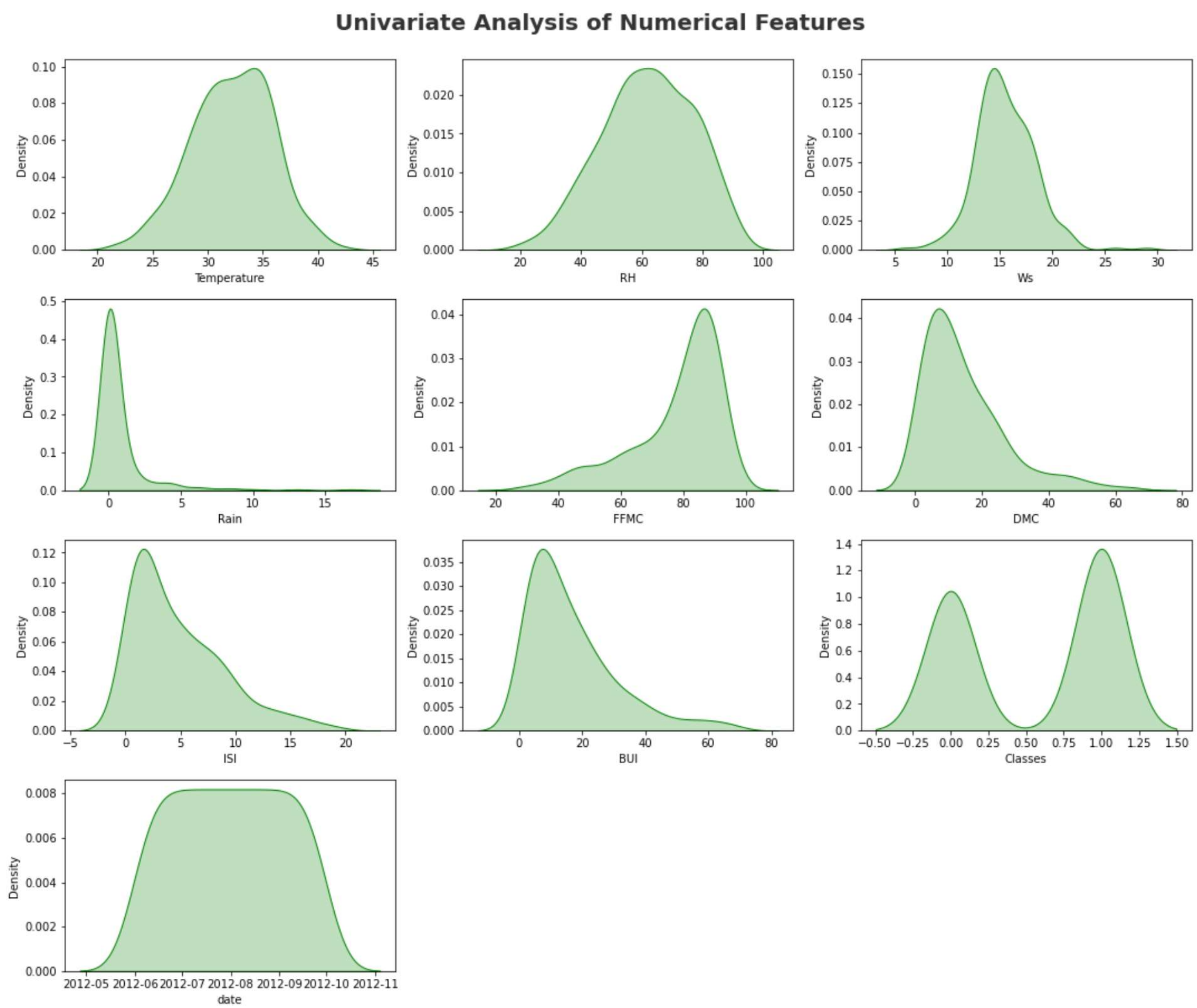
1. Date : (DD/MM/YYYY) Day, month ('june' to 'september'), year (2012) Weather data observations
2. Temp : temperature noon (temperature max) in Celsius degrees: 22 to 42
3. RH : Relative Humidity in %: 21 to 90
4. Ws :Wind speed in km/h: 6 to 29
5. Rain: total day in mm: 0 to 16.8 FWI Components
6. 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 â€œFireâ€ and â€œnot Fireâ€

Univeriate 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.

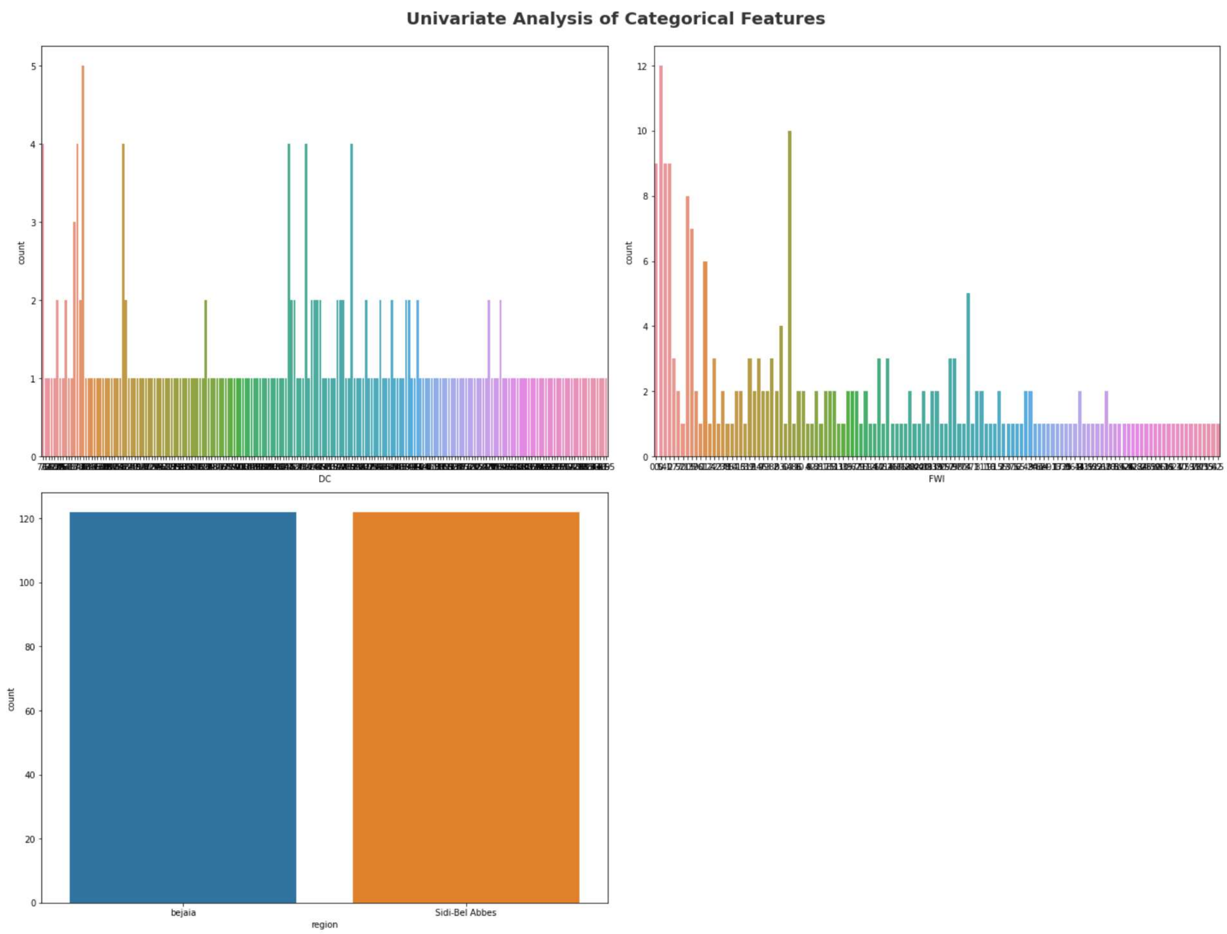
Numerical Features

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

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



Categorical Feature

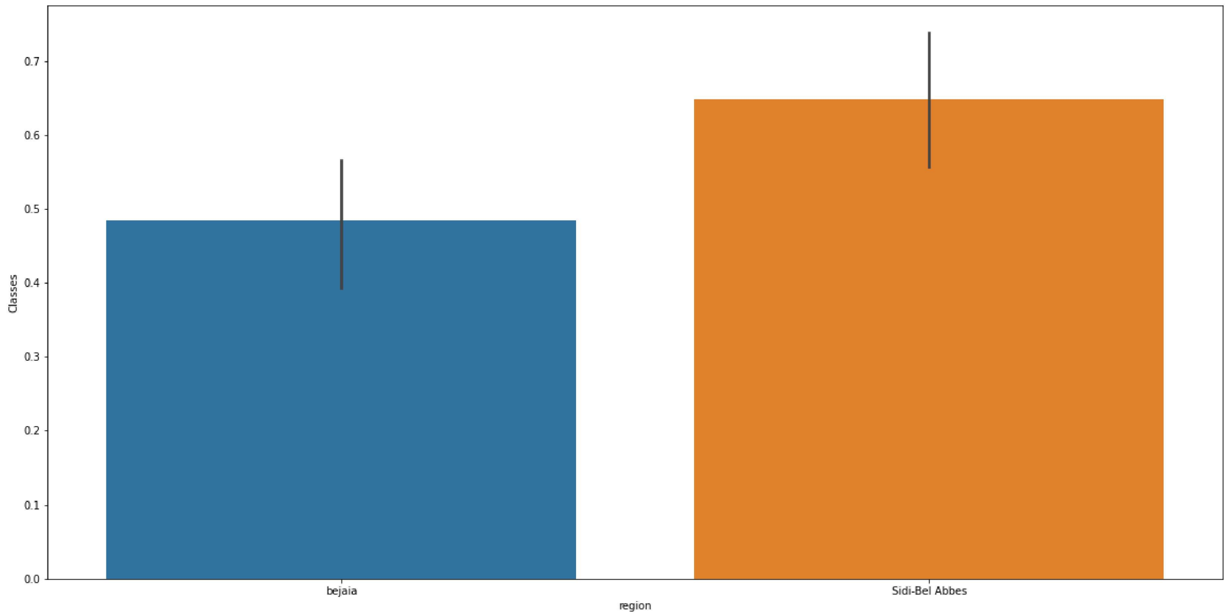
In [28]:
```
# categorical columns
plt.figure(figsize=(20, 15))
plt.suptitle('Univariate Analysis of Categorical Features', fontsize=20, fontweig
cat1 = ['DC', 'FWI','region']
for i in range(0, len(cat1)):
    plt.subplot(2,2, i+1)
    sns.countplot(x=df[cat1[i]],data=df)
    plt.xlabel(cat1[i])
    plt.tight_layout()
```



Which area has most of the time fire happen

In [29]:
```
import matplotlib
matplotlib.rcParams['figure.figsize']=(20,10)

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

Out[29]: <AxesSubplot:xlabel='region', ylabel='Classes'>



In [41]:
```
df.head()
```

Out[41]:

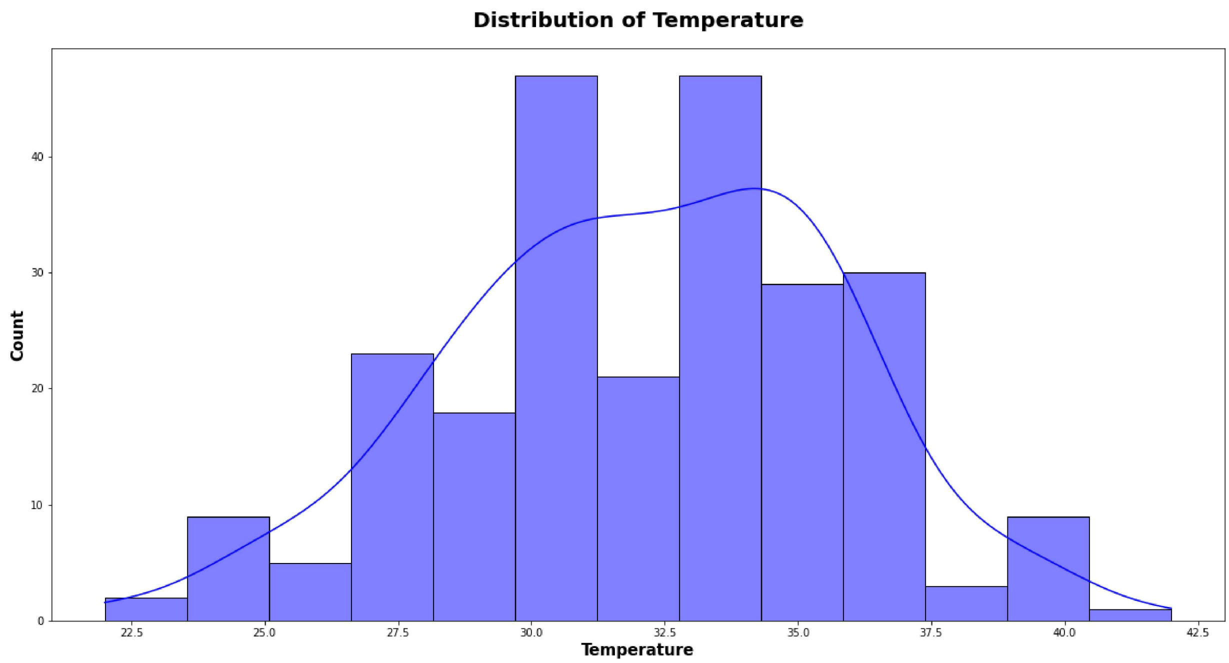
	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	region	date
0	29	57	18.0	0.0	65.7	3.4	7.6	1.3	3.4	0.5	0.0	bejaia	2012-06-01
1	29	61	13.0	1.3	64.4	4.1	7.6	1.0	3.9	0.4	0.0	bejaia	2012-06-02
2	26	82	22.0	13.1	47.1	2.5	7.1	0.3	2.7	0.1	0.0	bejaia	2012-06-03
3	25	89	13.0	2.5	28.6	1.3	6.9	0.0	1.7	0	0.0	bejaia	2012-06-04
4	27	77	16.0	0.0	64.8	3.0	14.2	1.2	3.9	0.5	0.0	bejaia	2012-06-05

Observation

Sidi=Bel Abbes region has most of the fire happen

temperature Range which is in most of the places

In [46]:
```
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()
```



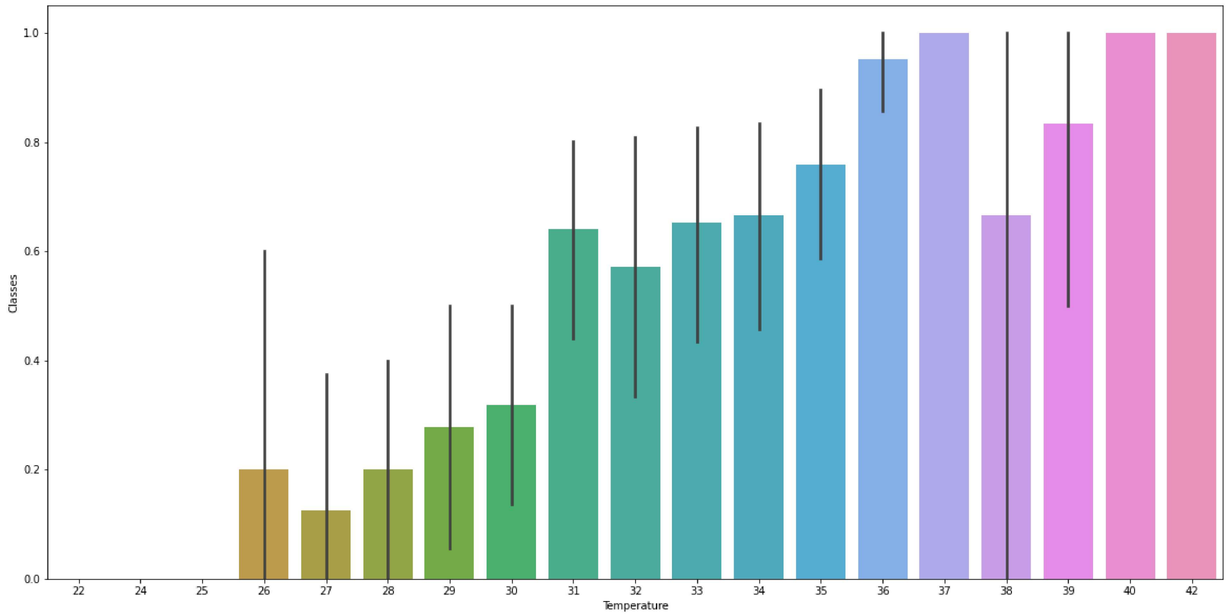
#Observation

Temperature occur most of the time in range 32.5 to 35.0

Highest Temperature attained

In [50]:
```
import matplotlib
matplotlib.rcParams['figure.figsize']=(20,10)

sns.barplot(x="Temperature",y="Classes",data=df)
```

Out[50]: <AxesSubplot:xlabel='Temperature', ylabel='Classes'>



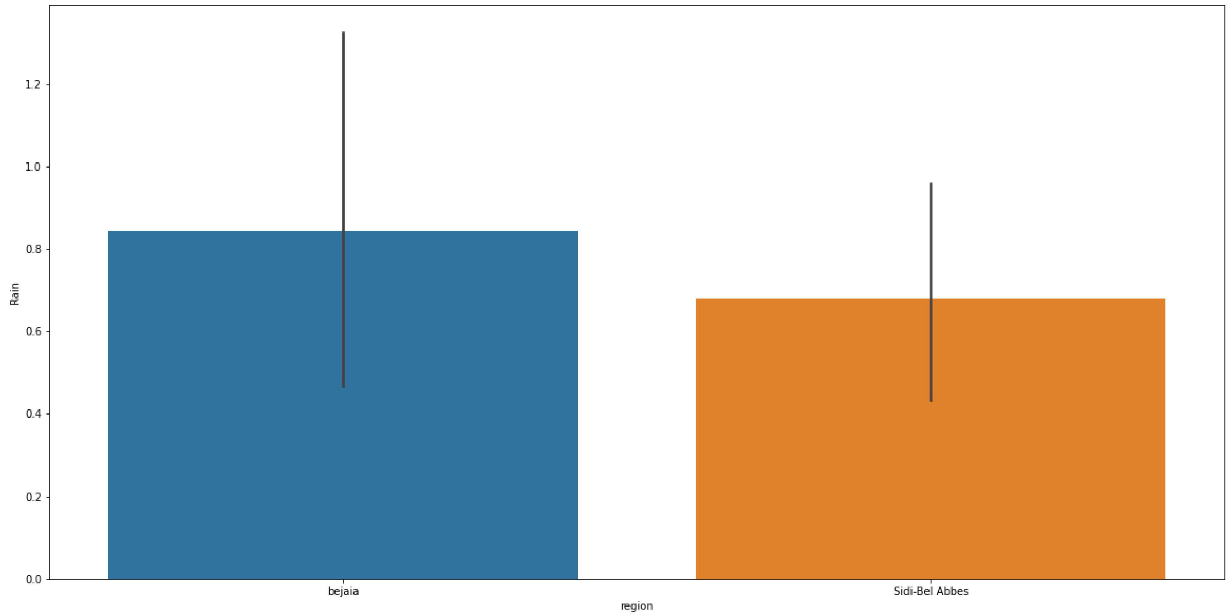
Observation

Highest temperature is 42,40,37

Which region has most time rain happens

In [51]:
```
import matplotlib
matplotlib.rcParams['figure.figsize']=(20,10)

sns.barplot(x="region",y="Rain",data=df)
```

Out[51]: <AxesSubplot:xlabel='region', ylabel='Rain'>



Observation

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

Multivariate Analysis

Multivariate analysis is the analysis of more than one variable.

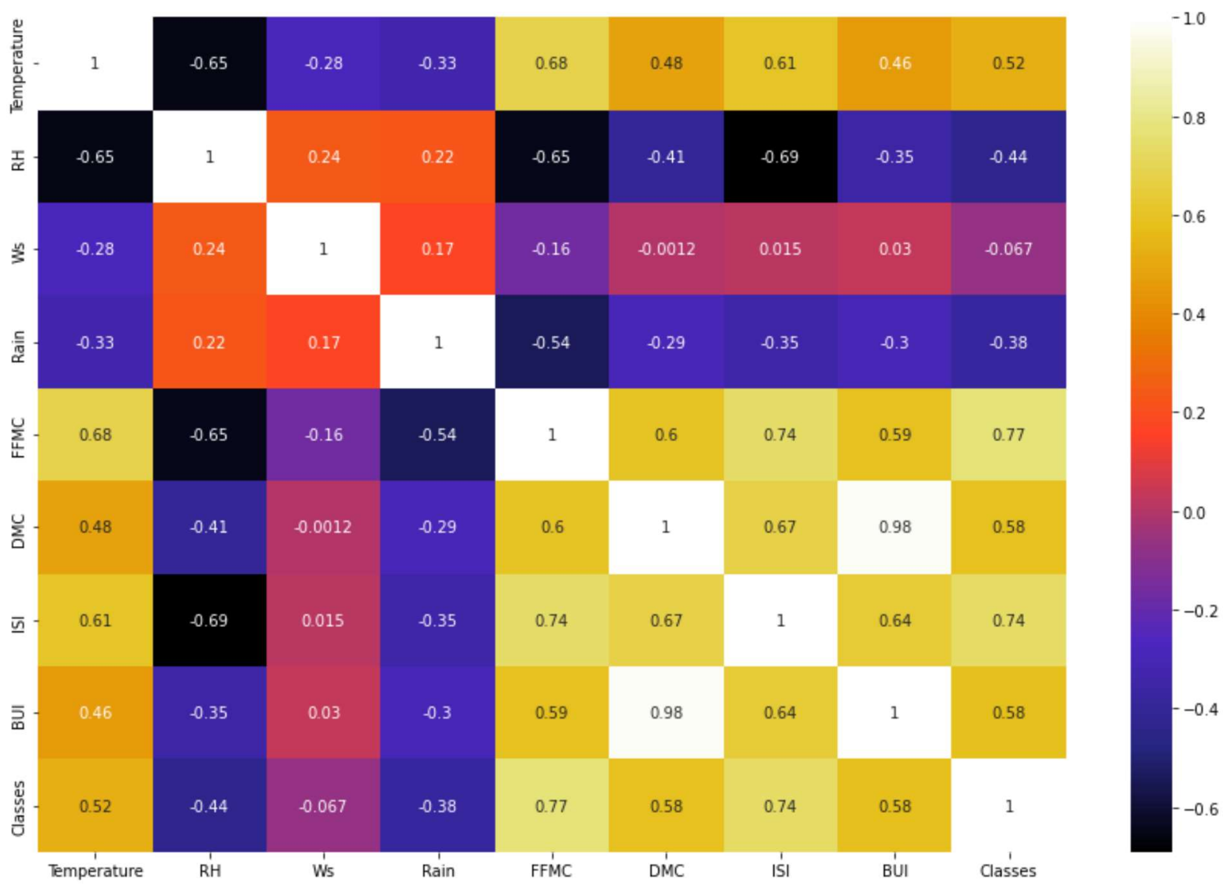
Numerical features

In [64]:
```
df.corr()
```

Out[64]:

	Temperature	RH	Ws	Rain	FFMC	DMC	ISI	B
Temperature	1.000000	-0.654443	-0.278132	-0.326786	0.677491	0.483105	0.607551	0.45550
RH	-0.654443	1.000000	0.236084	0.222968	-0.645658	-0.405133	-0.690637	-0.34858
Ws	-0.278132	0.236084	1.000000	0.170169	-0.163255	-0.001246	0.015248	0.02975
Rain	-0.326786	0.222968	0.170169	1.000000	-0.544045	-0.288548	-0.347105	-0.29917
FFMC	0.677491	-0.645658	-0.163255	-0.544045	1.000000	0.602391	0.739730	0.58965
DMC	0.483105	-0.405133	-0.001246	-0.288548	0.602391	1.000000	0.674499	0.98207
ISI	0.607551	-0.690637	0.015248	-0.347105	0.739730	0.674499	1.000000	0.63589
BUI	0.455504	-0.348587	0.029756	-0.299171	0.589652	0.982073	0.635891	1.00000
Classes	0.518119	-0.435023	-0.066529	-0.379449	0.770114	0.584188	0.735511	0.58388

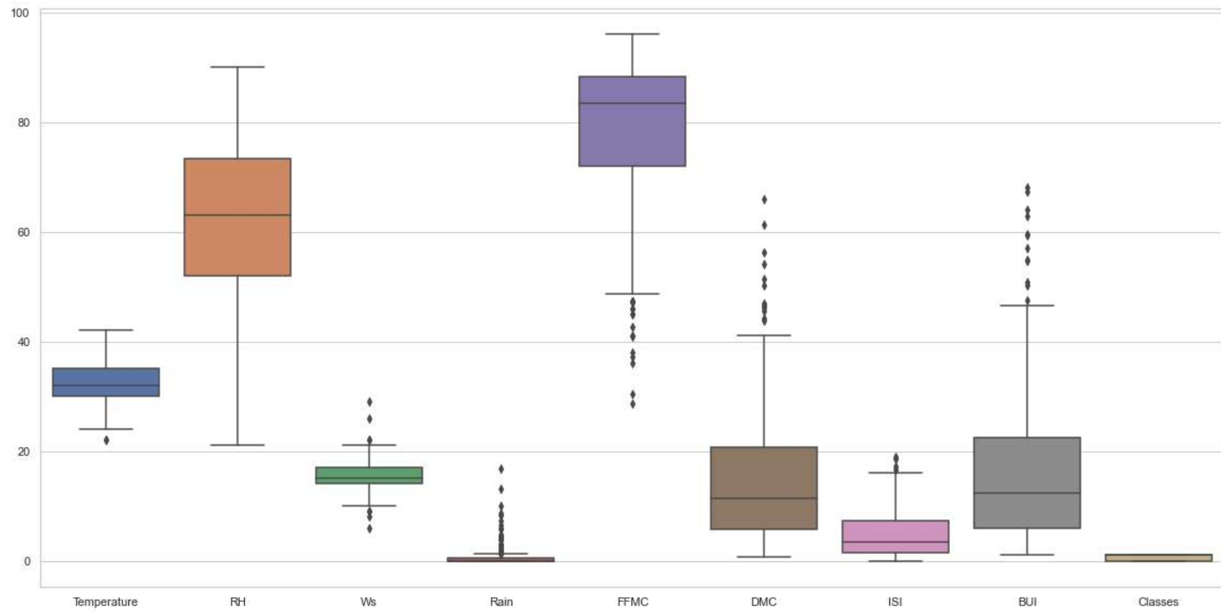
In [65]:
```
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(),cmap='CMRmap',annot=True)
plt.show()
```



Boxplot to find Outliers in the features

In [73]:
```
seaborn.boxplot(data = df,orient="v")
```

Out[73]: <AxesSubplot:>

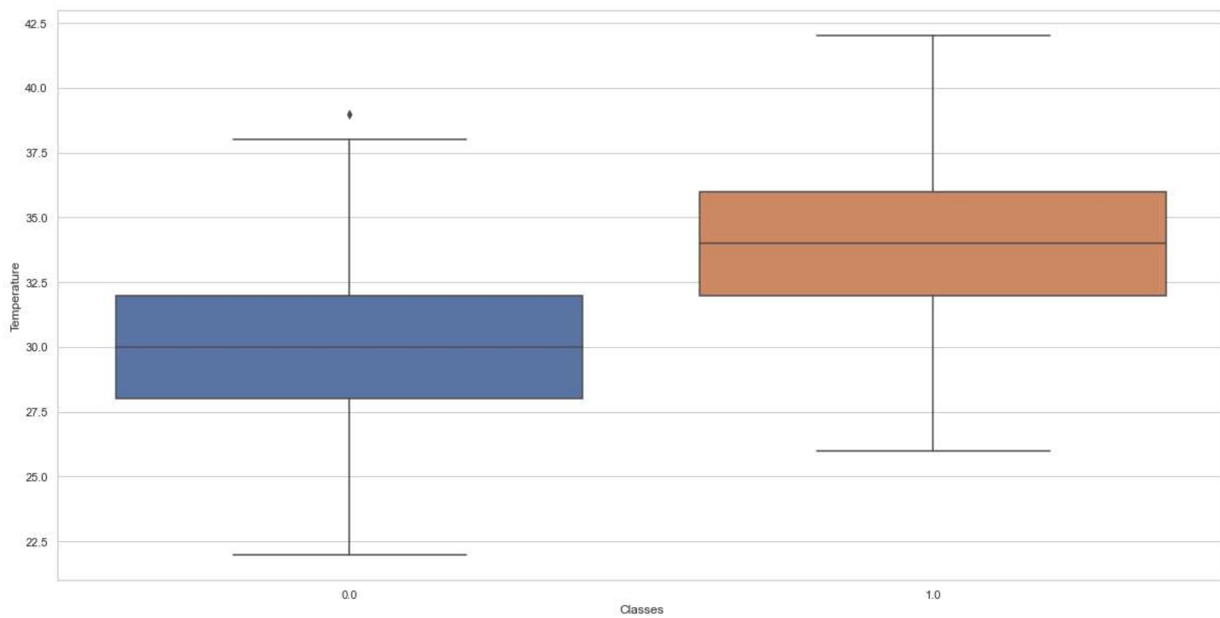


- RH, Rain, FFMC, DMC BUI has many outliers

Boxplot of Class Vs Temperature

In [75]:
```
# Python program to illustrate
# boxplot using inbuilt data-set
# given in seaborn

# importing the required module
import seaborn

# use to set style of background of plot
seaborn.set(style="whitegrid")

# loading data-set

seaborn.boxplot(x ='Classes', y ='Temperature', data = df)
```

Out[75]: <AxesSubplot:xlabel='Classes', ylabel='Temperature'>

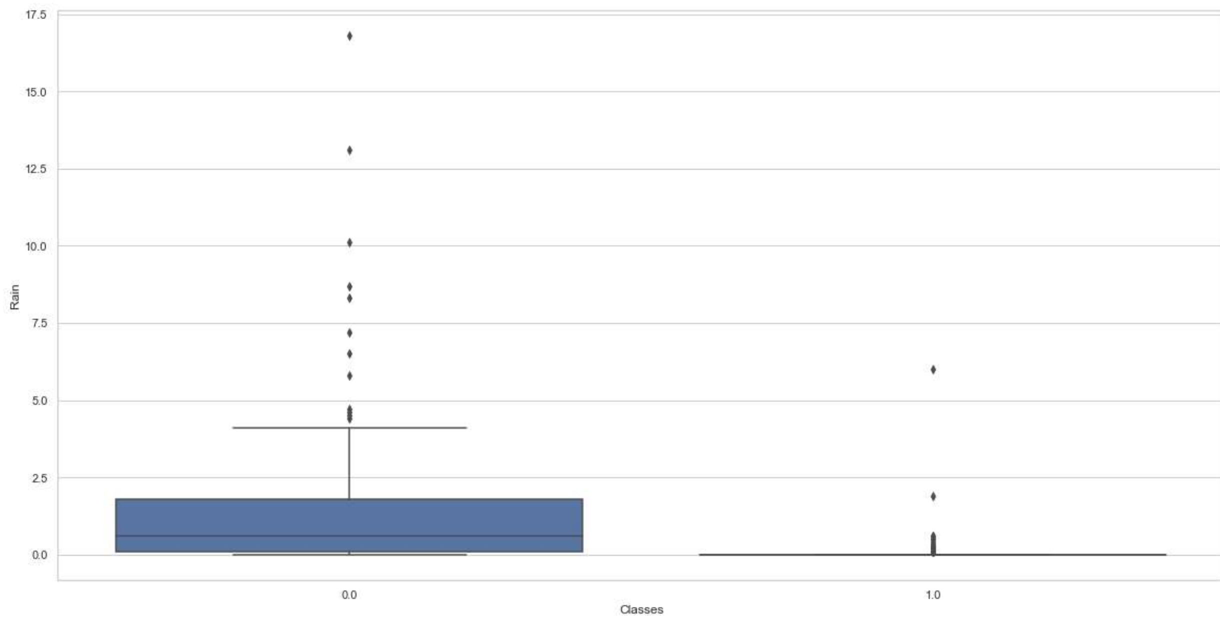


- One day at lower temperature fires occur

Boxplot Vs Rain

In [76]:
```
# Python program to illustrate
# boxplot using inbuilt data-set
# given in seaborn

# importing the required module
import seaborn

# use to set style of background of plot
seaborn.set(style="whitegrid")

# Loading data-set

seaborn.boxplot(x ='Classes', y ='Rain', data = df)
```

Out[76]: <AxesSubplot:xlabel='Classes', ylabel='Rain'>



- In many days after having rain also fire occur

STATISTICAL ANALYSIS

In [77]:
```
df.describe()
```

Out[77]:

	Temperature	RH	Ws	Rain	FFMC	DMC	ISI	
count	244.000000	244.000000	244.000000	244.000000	244.000000	244.000000	244.000000	244.0
mean	32.172131	61.938525	15.504098	0.760656	77.887705	14.673361	4.774180	16.6
std	3.633843	14.884200	2.810178	1.999406	14.337571	12.368039	4.175318	14.2
min	22.000000	21.000000	6.000000	0.000000	28.600000	0.700000	0.000000	1.1
25%	30.000000	52.000000	14.000000	0.000000	72.075000	5.800000	1.400000	6.0
50%	32.000000	63.000000	15.000000	0.000000	83.500000	11.300000	3.500000	12.2
75%	35.000000	73.250000	17.000000	0.500000	88.300000	20.750000	7.300000	22.5
max	42.000000	90.000000	29.000000	16.800000	96.000000	65.900000	19.000000	68.0