EDA, Feature Engineering and Logistic regression model implementation on dataset.

1.Data Profiling

2.Data Cleaning

3.Statistical Analysis

4.Graphical Analysis

5.Data Scaling

6.Logistic Regression Model on Original dataset and performance metrices on this dataset

7.Creating an imbalanced dataset from original dataset

Balancing the imbalanced Dataset

Logistic Regression on above dataset

Performance Metrics for above Dataset

In [1]:
```
# Importing the libraries

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [2]:
```
df = pd.read_csv(r'C:\Users\Rakhi Amberiya\Downloads\Algerian_forest_fires_dataset_UPDATE (2).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]:
```
df.shape
```

Out[3]: (246, 14)

In [4]:
```
# Dropping the unwanted rows and column

df.drop([122,123],inplace=True)
df.reset_index(inplace=True)
df.drop('index',axis=1,inplace=True)
```

In [5]:
```
df.shape
```

Out[5]: (244, 14)

In [6]:
```
df.tail(124)
```

Out[6]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes
120	29	09	2012	26	80	16	1.8	47.4	2.9	7.7	0.3	3	0.1	not fire
121	30	09	2012	25	78	14	1.4	45	1.9	7.5	0.2	2.4	0.1	not fire
122	01	06	2012	32	71	12	0.7	57.1	2.5	8.2	0.6	2.8	0.2	not fire
123	02	06	2012	30	73	13	4	55.7	2.7	7.8	0.6	2.9	0.2	not fire
124	03	06	2012	29	80	14	2	48.7	2.2	7.6	0.3	2.6	0.1	not fire
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
239	26	09	2012	30	65	14	0	85.4	16	44.5	4.5	16.9	6.5	fire
240	27	09	2012	28	87	15	4.4	41.1	6.5	8	0.1	6.2	0	not fire
241	28	09	2012	27	87	29	0.5	45.9	3.5	7.9	0.4	3.4	0.2	not fire
242	29	09	2012	24	54	18	0.1	79.7	4.3	15.2	1.7	5.1	0.7	not fire
243	30	09	2012	24	64	15	0.2	67.3	3.8	16.5	1.2	4.8	0.5	not fire

124 rows × 14 columns

In [7]:
```
#Creating a column region

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

In [8]:
```
df.columns
```

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

In [9]:
```
#Stripping the names of the columns as it has unwanted spaces

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

Out[9]:
```
Index(['day', 'month', 'year', 'Temperature', 'RH', 'Ws', 'Rain', 'FFMC',
       'DMC', 'DC', 'ISI', 'BUI', 'FWI', 'Classes', 'region'],
      dtype='object')
```

In [10]:
```
#Stripping the data of below features as it has unwanted spaces


for feature in ['Rain','FFMC','DMC','DC','ISI','BUI','FWI','Classes']:
    df[feature]=df[feature].str.replace(" ","")
```

In [11]:
```
#Checking the unique data points of FWI feature


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

Out[11]:
```
array(['0.5', '0.4', '0.1', '0', '2.5', '7.2', '7.1', '0.3', '0.9', '5.6',
       '0.2', '1.4', '2.2', '2.3', '3.8', '7.5', '8.4', '10.6', '15',
       '13.9', '3.9', '12.9', '1.7', '4.9', '6.8', '3.2', '8', '0.6',
       '3.4', '0.8', '3.6', '6', '10.9', '4', '8.8', '2.8', '2.1', '1.3',
       '7.3', '15.3', '11.3', '11.9', '10.7', '15.7', '6.1', '2.6', '9.9',
       '11.6', '12.1', '4.2', '10.2', '6.3', '14.6', '16.1', '17.2',
       '16.8', '18.4', '20.4', '22.3', '20.9', '20.3', '13.7', '13.2',
       '19.9', '30.2', '5.9', '7.7', '9.7', '8.3', '0.7', '4.1', '1',
       '3.1', '1.9', '10', '16.7', '1.2', '5.3', '6.7', '9.5', '12',
       '6.4', '5.2', '3', '9.6', '4.7', 'fire', '14.1', '9.1', '13',
       '17.3', '30', '25.4', '16.3', '9', '14.5', '13.5', '19.5', '12.6',
       '12.7', '21.6', '18.8', '10.5', '5.5', '14.8', '24', '26.3',
       '12.2', '18.1', '24.5', '26.9', '31.1', '30.3', '26.1', '16',
       '19.4', '2.7', '3.7', '10.3', '5.7', '9.8', '19.3', '17.5', '15.4',
       '15.2', '6.5'], dtype=object)
```

Here we got 'fire' data point in FWI column , so to remove this it is replaced by mode

In [12]:
```
df[df['FWI']=='fire'].index
```

Out[12]:
```
Int64Index([165], dtype='int64')
```

In [13]:
```
df['FWI'].mode()
```

Out[13]:
```
0    0.4
dtype: object
```

In [14]:
```
df.loc[165,'FWI']='0.4'
```

In [15]:
```
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    object
 1   month        244 non-null    object
 2   year         244 non-null    object
 3   Temperature  244 non-null    object
 4   RH           244 non-null    object
 5   Ws           244 non-null    object
 6   Rain         244 non-null    object
 7   FFMC         244 non-null    object
 8   DMC          244 non-null    object
 9   DC           244 non-null    object
 10  ISI          244 non-null    object
 11  BUI          244 non-null    object
 12  FWI          244 non-null    object
 13  Classes      243 non-null    object
 14  region       244 non-null    object
dtypes: object(15)
memory usage: 28.7+ KB
```

Here all the features are in categorical datatypes.

In [16]:
```
#Checking the null value in the features

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

Out[16]:
```
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
```

1 null value is present in Classes feature

In [17]:
```
#Replacing the null value of class feature


df[df['Classes']=='nan'].index
```

Out[17]:
```
Int64Index([], dtype='int64')
```

In [18]:
```
df.loc[165,'Classes']='fire'
```

In [19]:
```
df['Classes'].unique()
```

Out[19]:
```
array(['notfire', 'fire'], dtype=object)
```

In [20]:
```
df['Classes']=df['Classes'].str.replace('notfire','0')
df['Classes']=df['Classes'].str.replace('fire','1')
```

Replacing the values of region feature with 0 and 1

In [21]:
```
df['region'].unique()
```

Out[21]:
```
array(['bejaia', 'Sidi-Bel Abbes'], dtype=object)
```

In [22]:
```
df['region']=df['region'].str.replace('bejaia','0')
df['region']=df['region'].str.replace('Sidi-Bel Abbes','1')
```

Converting the datatype from categorical to numerical

In [23]:
```
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['Ws']=df['Ws'].astype(int)
df['Rain']=df['Rain'].astype(float)
df['FFMC']=df['FFMC'].astype(float)
df['DMC']=df['DMC'].astype(float)
df['ISI']=df['ISI'].astype(float)
df['BUI']=df['BUI'].astype(float)
df['DC']=df['DC'].astype(float)
df['FWI']=df['FWI'].astype(float)
df['Classes']=df['Classes'].astype(int)
df['region']=df['region'].astype(int)
```

In [24]:
```
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    int32  
 6   Rain         244 non-null    float64
 7   FFMC         244 non-null    float64
 8   DMC          244 non-null    float64
 9   DC           244 non-null    float64
 10  ISI          244 non-null    float64
 11  BUI          244 non-null    float64
 12  FWI          244 non-null    float64
 13  Classes      244 non-null    int32  
 14  region       244 non-null    int32  
dtypes: float64(7), int32(8)
memory usage: 21.1 KB
```

So, all the features are converted from categorical to numerical datatypes

Creating a Copy of dataframe from Original Dataframe

In [25]:
```
data=df.copy()
data.head()
```

Out[25]:

	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

Feature Information

*Date : (DD/MM/YYYY) Day, month ('june' to 'september'), year (2012) Weather data observations

*Temp : temperature noon (temperature max) in Celsius degrees: 22 to 42

*RH : Relative Humidity in %: 21 to 90

*Ws :Wind speed in km/h: 6 to 29

*Rain: total day in mm: 0 to 16.8 FWI Components

*Fine Fuel Moisture Code (FFMC) index from the FWI system: 28.6 to 92.5

*Duff Moisture Code (DMC) index from the FWI system: 1.1 to 65.9

*Drought Code (DC) index from the FWI system: 7 to 220.4

*Initial Spread Index (ISI) index from the FWI system: 0 to 18.5

*Buildup Index (BUI) index from the FWI system: 1.1 to 68

*Fire Weather Index (FWI) Index: 0 to 31.1

Statistical Analysis

In [26]:
```
data.describe()
```

Out[26]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FW
count	244.000000	244.000000	244.0	244.000000	244.000000	244.000000	244.000000	244.000000	244.000000	244.000000	244.000000	244.000000	244.00000
mean	15.754098	7.500000	2012.0	32.172131	61.938525	15.504098	0.760656	77.887705	14.673361	49.288484	4.774180	16.664754	7.00819
std	8.825059	1.112961	0.0	3.633843	14.884200	2.810178	1.999406	14.337571	12.368039	47.619393	4.175318	14.204824	7.43738
min	1.000000	6.000000	2012.0	22.000000	21.000000	6.000000	0.000000	28.600000	0.700000	6.900000	0.000000	1.100000	0.00000
25%	8.000000	7.000000	2012.0	30.000000	52.000000	14.000000	0.000000	72.075000	5.800000	13.275000	1.400000	6.000000	0.70000
50%	16.000000	7.500000	2012.0	32.000000	63.000000	15.000000	0.000000	83.500000	11.300000	33.100000	3.500000	12.250000	4.20000
75%	23.000000	8.000000	2012.0	35.000000	73.250000	17.000000	0.500000	88.300000	20.750000	68.150000	7.300000	22.525000	11.37500
max	31.000000	9.000000	2012.0	42.000000	90.000000	29.000000	16.800000	96.000000	65.900000	220.400000	19.000000	68.000000	31.10000

Observation

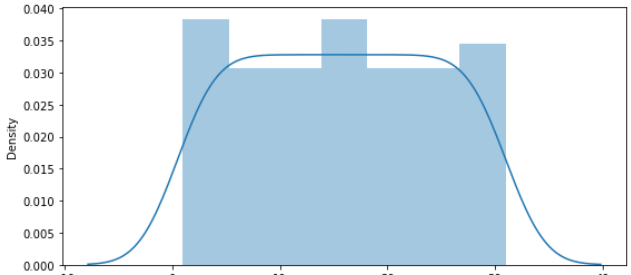
No missing values are present in the dataset

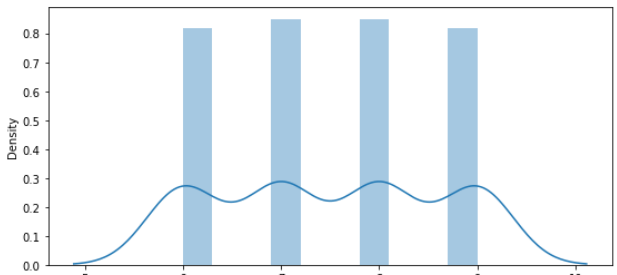
In [27]:
```
data.cov()
```

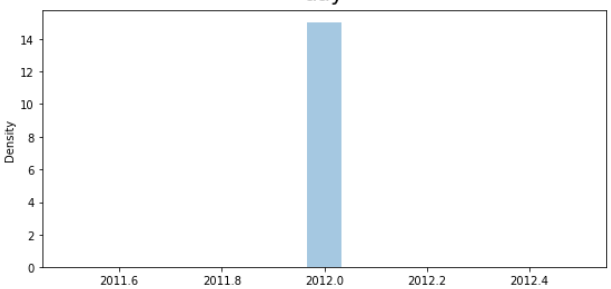
Out[27]:

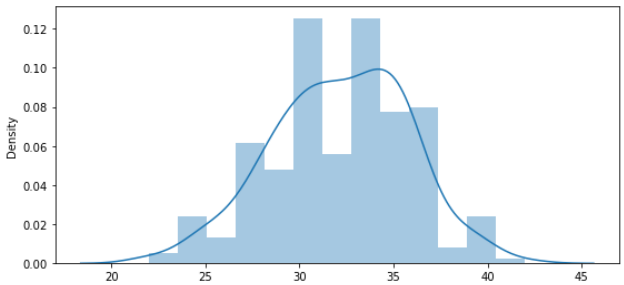
	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI
day	7.788167e+01	4.641920e-16	0.0	3.071308	-9.747689	1.165621	-1.980908	28.346758	53.654328	221.859379	6.548769	64.839034
month	4.641920e-16	1.238683e+00	0.0	-0.238683	-0.627572	-0.129630	0.078601	0.248560	0.938477	6.766276	0.286626	1.356790
year	0.000000e+00	0.000000e+00	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Temperature	3.071308e+00	-2.386831e-01	0.0	13.204817	-35.396782	-2.840215	-2.374270	35.297598	21.712423	64.113719	9.218043	23.512265
RH	-9.747689e+00	-6.275720e-01	0.0	-35.396782	221.539415	9.874739	6.635431	-137.785533	-74.580245	-156.174991	-42.920524	-73.700941
Ws	1.165621e+00	-1.296296e-01	0.0	-2.840215	9.874739	7.897102	0.956129	-6.577727	-0.043306	10.204060	0.178913	1.187799
Rain	-1.980908e+00	7.860082e-02	0.0	-2.374270	6.635431	0.956129	3.997623	-15.595918	-7.135415	-28.259196	-2.897687	-8.496825
FFMC	2.834676e+01	2.485597e-01	0.0	35.297598	-137.785533	-6.577727	-15.595918	205.565939	106.820535	344.048788	44.283138	120.090018
DMC	5.365433e+01	9.384774e-01	0.0	21.712423	-74.580245	-0.043306	-7.135415	106.820535	152.968382	515.551947	34.831449	172.536341
DC	2.218594e+02	6.766276e+00	0.0	64.113719	-156.174991	10.204060	-28.259196	344.048788	515.551947	2267.606583	99.199508	637.129111
ISI	6.548769e+00	2.866255e-01	0.0	9.218043	-42.920524	0.178913	-2.897687	44.283138	34.831449	99.199508	17.433281	37.714477
BUI	6.483903e+01	1.356790e+00	0.0	23.512265	-73.700941	1.187799	-8.496825	120.090018	172.536341	637.129111	37.714477	201.777024
FWI	2.303207e+01	6.962963e-01	0.0	15.102287	-63.152169	0.606139	-4.800293	73.187426	80.480590	262.143165	28.198196	90.628767
Classes	8.845038e-01	1.234568e-02	0.0	0.935168	-3.216117	-0.092862	-0.376833	5.484349	3.588791	11.994976	1.525363	4.119605
region	2.924044e-17	0.000000e+00	0.0	0.497942	-3.030864	-0.248971	-0.041152	1.613992	1.184156	-1.944053	0.561523	0.621811

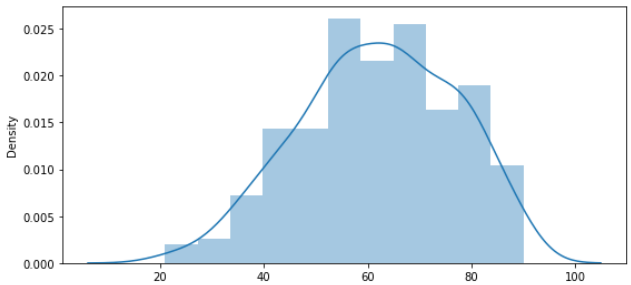
In [28]:
```
#Checking the distribution of the features

# let's see how data is distributed for every column


plt.figure(figsize=(20,40), facecolor='white')
plotnumber = 1

for column in data:
    if plotnumber<=15 :     # as there are 15 columns in the data
        ax = plt.subplot(8,2,plotnumber)
        sns.distplot(data[column])
        plt.xlabel(column,fontsize=20)

    plotnumber+=1
plt.show()
```

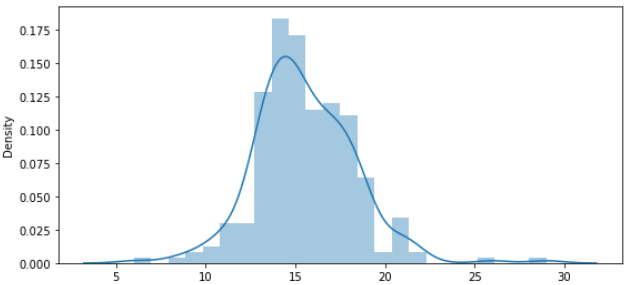

Density
day

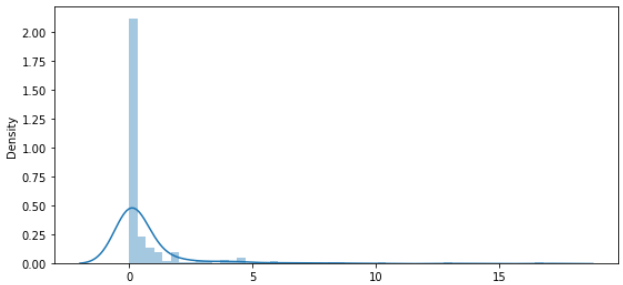

Density
month

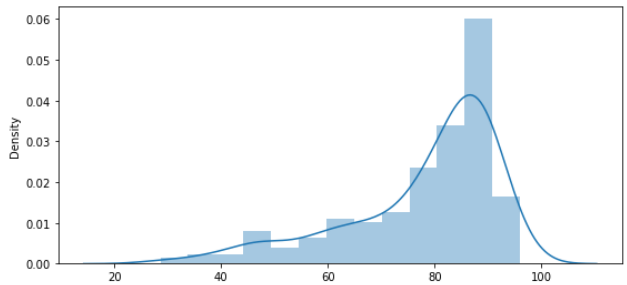

Density
year

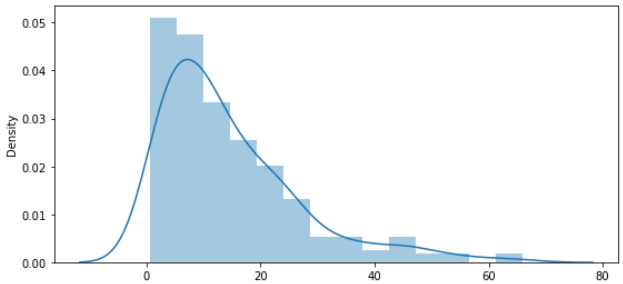

Density
Temperature

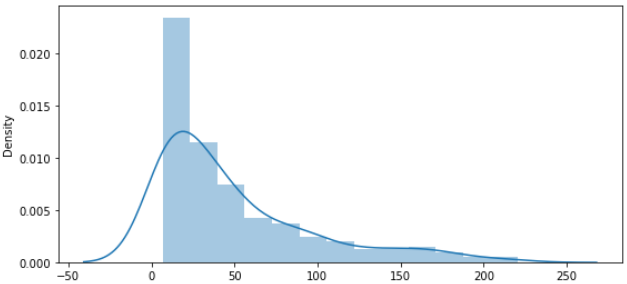

Density
RH

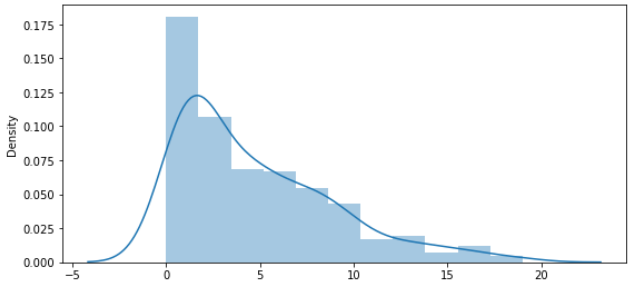

Density
Ws

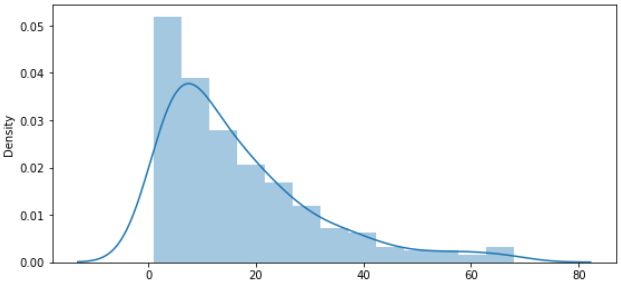

Density
Rain

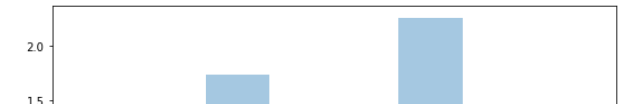

Density
FFMC

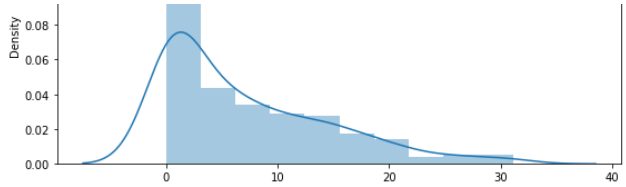

Density
DMC

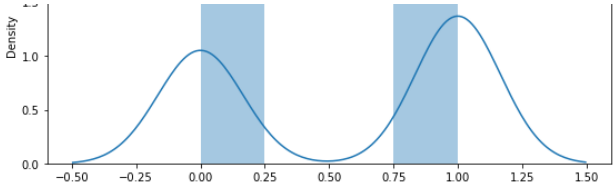

Density
DC

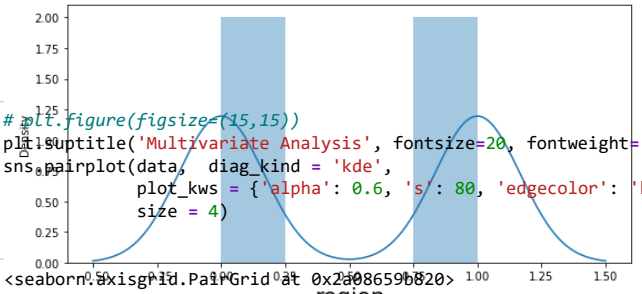

Density
ISI


Density
BUI





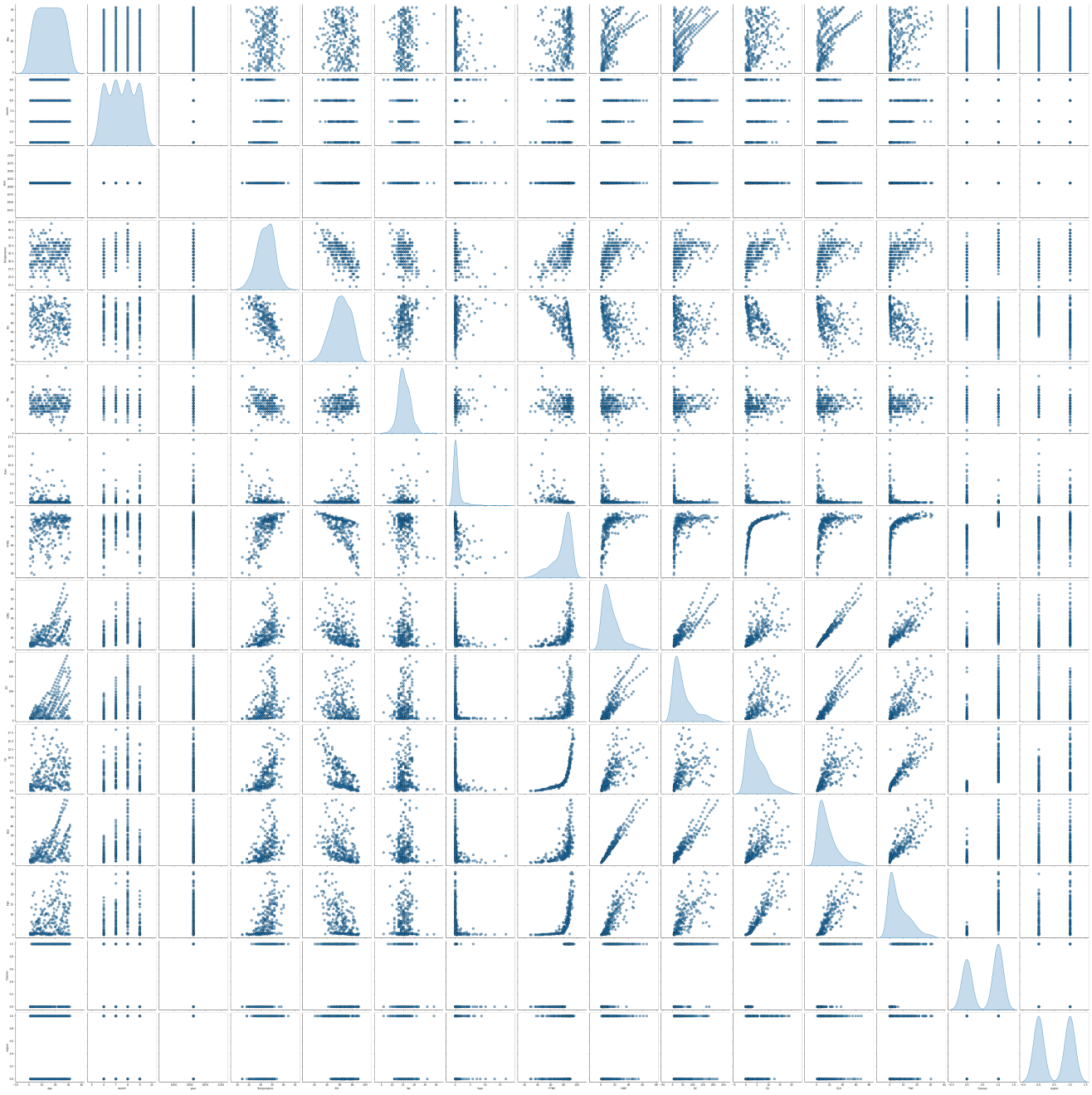




In [29]:
```
# plt.figure(figsize=(15,15))
plt.suptitle('Multivariate Analysis', fontsize=20, fontweight='bold', alpha=0.8, y=1.)
sns.pairplot(data,  diag_kind = 'kde',
             plot_kws = {'alpha': 0.6, 's': 80, 'edgecolor': 'k'},
             size = 4)
```

Out[29]: <seaborn.axisgrid.PairGrid at 0x2a08659b820>

<Figure size 432x288 with 0 Axes>



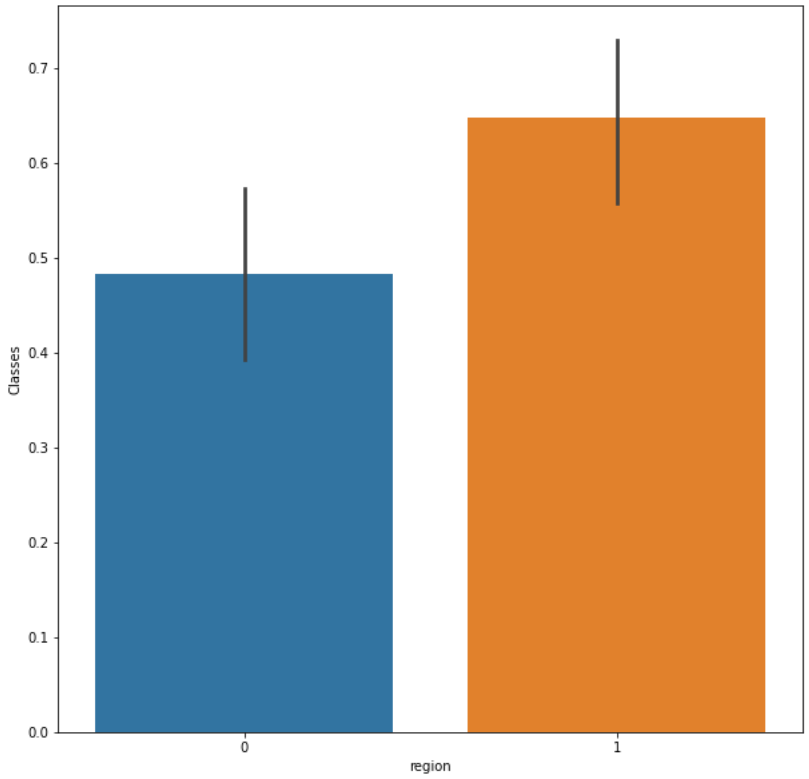
Visualisation of Target Feature

In [30]:
```
data.Classes.value_counts()
```

Out[30]:
```
1    138
0    106
Name: Classes, dtype: int64
```

In [31]:
```
import matplotlib
matplotlib.rcParams['figure.figsize']=(10,10)

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

Out[31]: <AxesSubplot:xlabel='region', ylabel='Classes'>

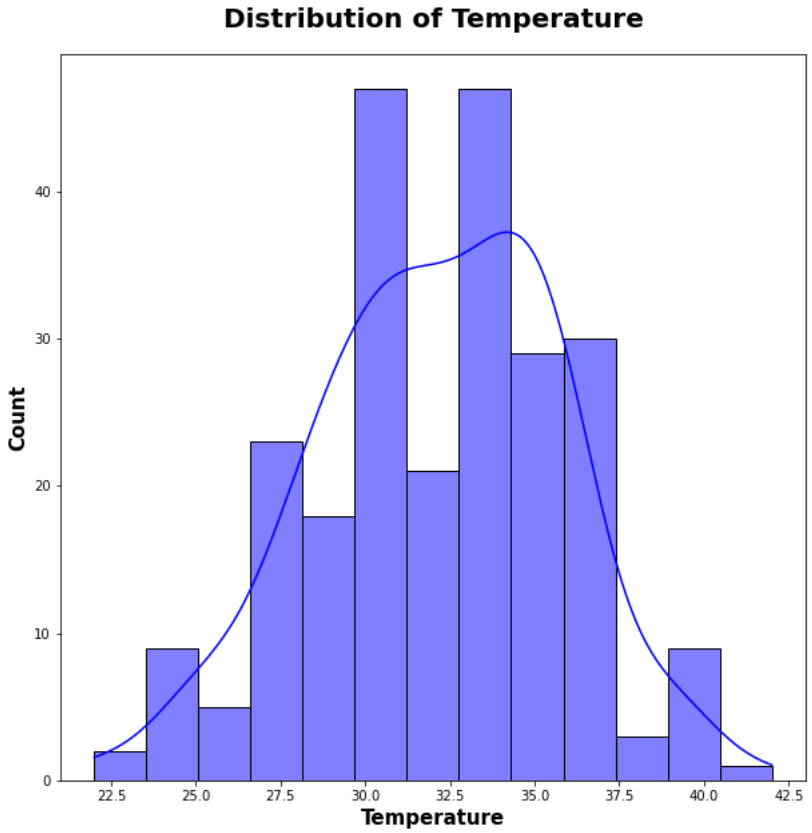


Observation

Sidi=Bel Abbes region has most of the fire feature

Visualisation of Temperature Feature

In [32]:
```
plt.subplots(figsize=(10,10))
sns.histplot("Distribution of Temperature",x=data.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()
```



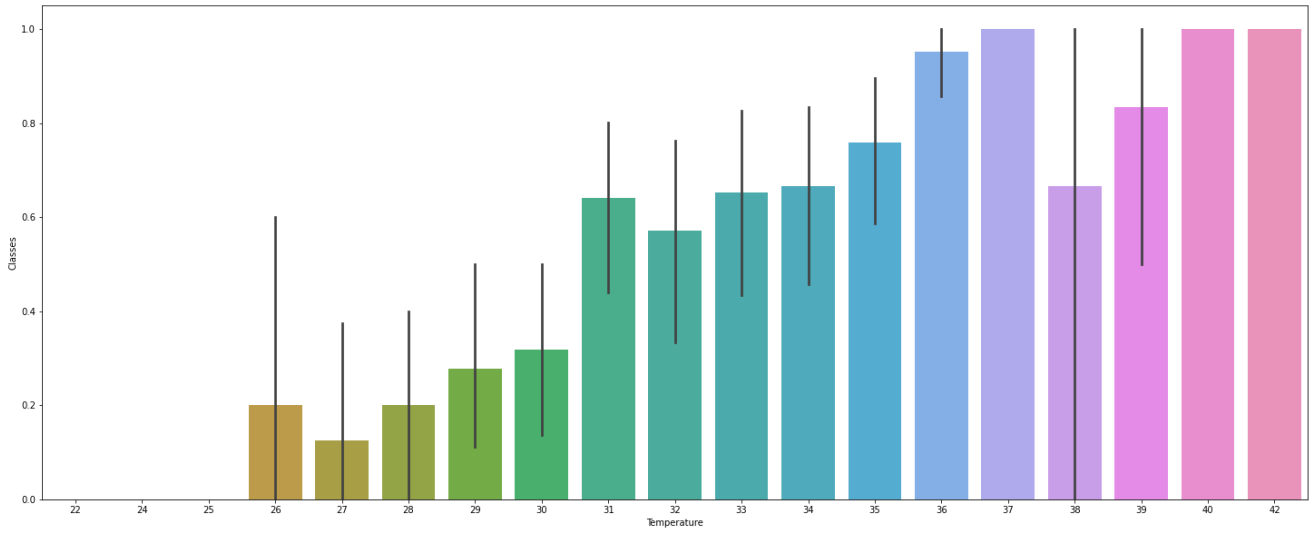
Observation

Temperature occur most of the time in range 32.5 to 35.0

Highest Temperature attained

In [33]:
```
import matplotlib
matplotlib.rcParams['figure.figsize']=(25,10)

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

Out[33]: <AxesSubplot:xlabel='Temperature', ylabel='Classes'>



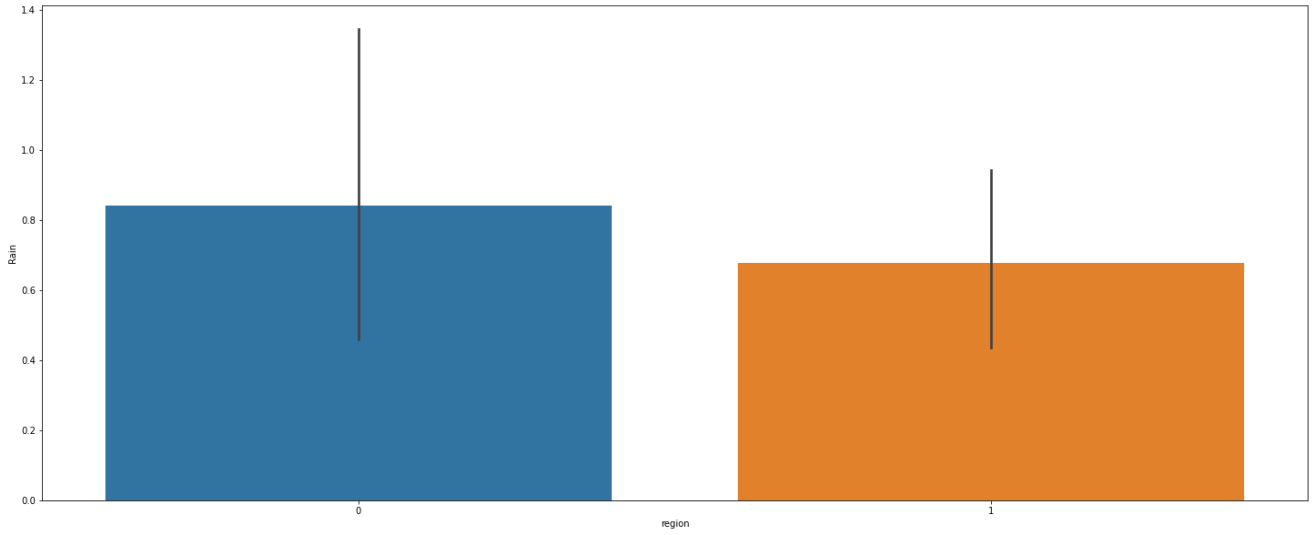
Observation

Highest temperature is 42, 40, 37

Which region is mostly effected by rain

In [34]:
```
import matplotlib
matplotlib.rcParams['figure.figsize']=(25,10)

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

Out[34]: <AxesSubplot:xlabel='region', ylabel='Rain'>



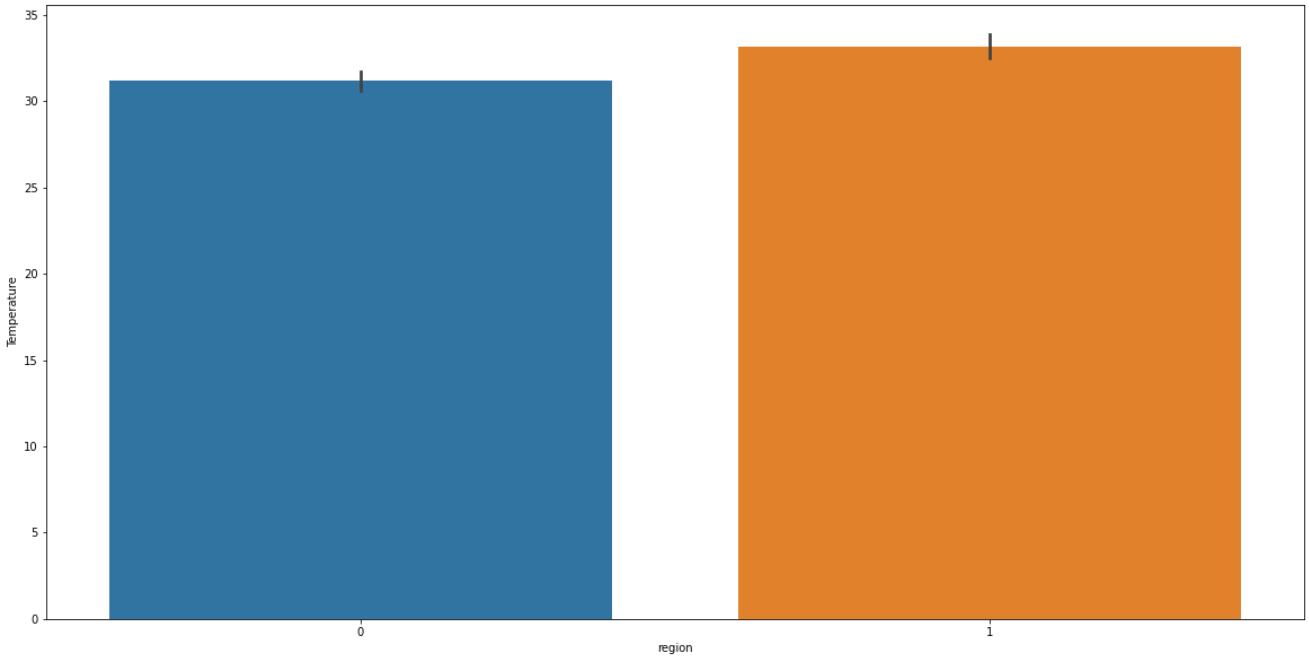
Observation

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

Which region is highly effected by Temperature

In [35]:
```
import matplotlib
matplotlib.rcParams['figure.figsize']=(20,10)

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

Out[35]: <AxesSubplot:xlabel='region', ylabel='Temperature'>



Observation

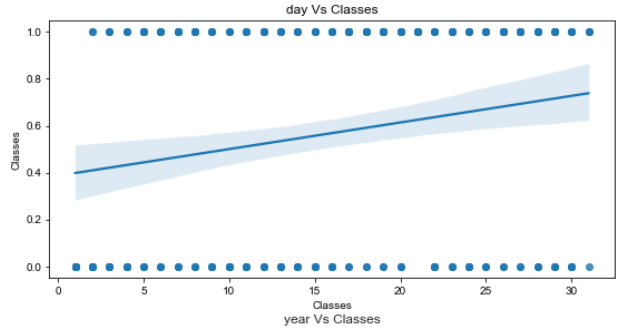
Sidi=Bel Abbes region mostly effected by Temperature

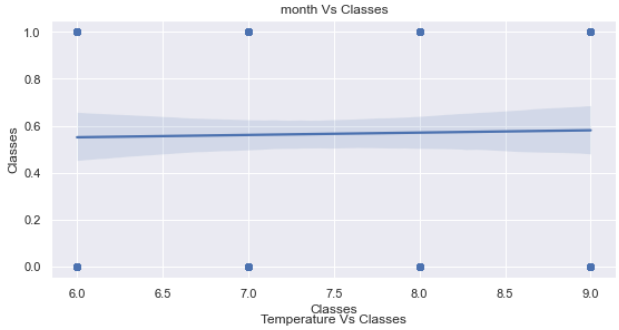
Reg plot

In [36]:
```
num_col=[feature for feature in data.columns if data[feature].dtype != 'O']
num_col
```

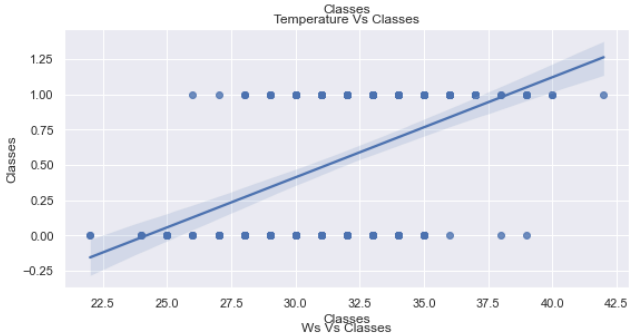
Out[36]:
```
['day',
 'month',
 'year',
 'Temperature',
 'RH',
 'Ws',
 'Rain',
 'FFMC',
 'DMC',
 'DC',
 'ISI',
 'BUI',
 'FWI',
 'Classes',
 'region']
```

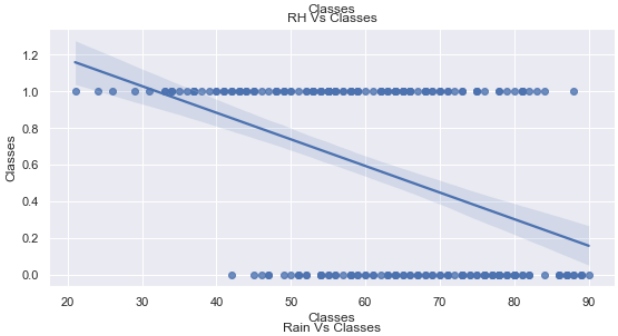
In [37]:
```
plt.figure(figsize=(20,40))
for i in enumerate(num_col):
    plt.subplot(8,2,i[0]+1)
    sns.set(rc={'figure.figsize':(8,10)})
    sns.regplot(data=data,x=i[1],y='Classes')
    plt.xlabel('Classes')
    plt.title('{} Vs Classes'.format(i[1]))
```

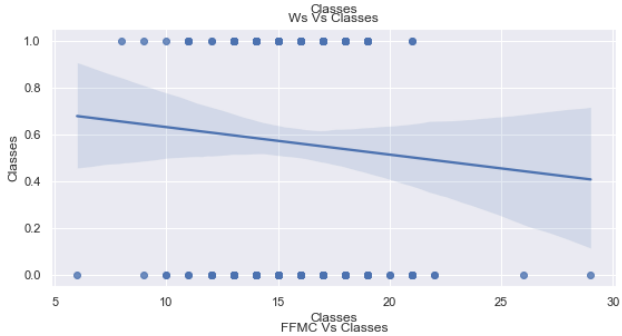

day Vs Classes
Classes
Classes

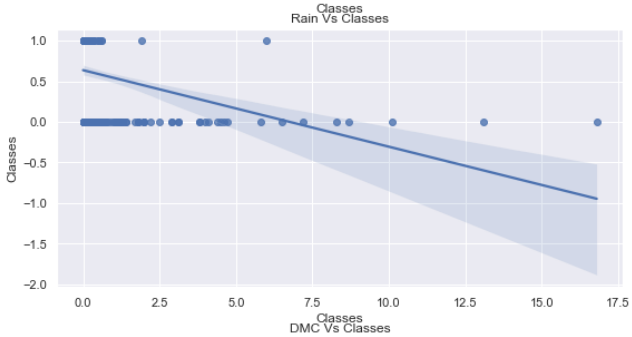

month Vs Classes
Classes
Classes

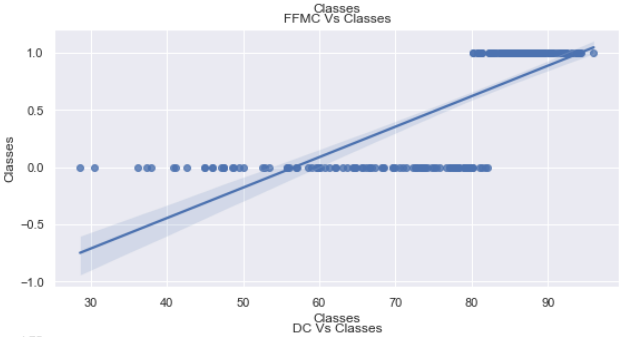

year Vs Classes
Classes
Classes

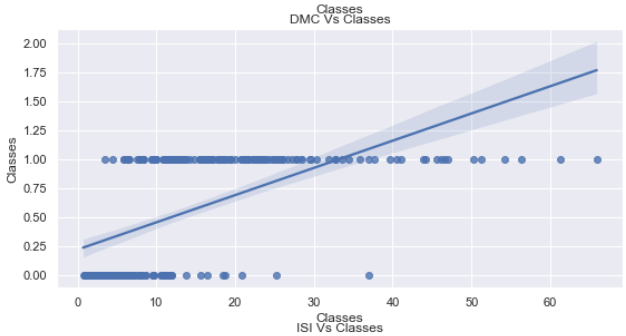

Temperature Vs Classes
Classes
Classes

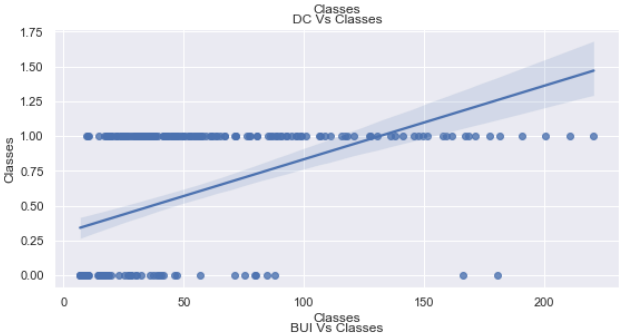

RH Vs Classes
Classes
Classes

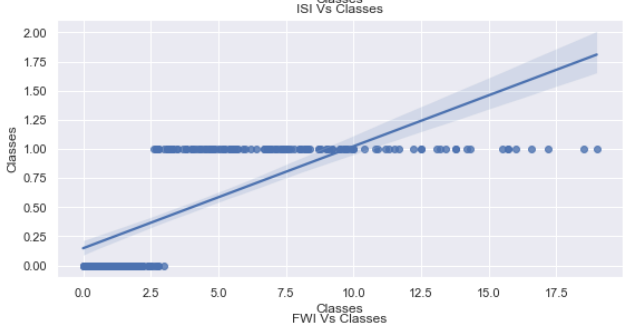

Ws Vs Classes
Classes
Classes

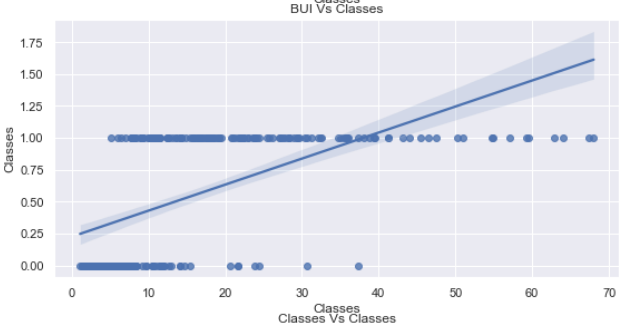

Rain Vs Classes
Classes
Classes

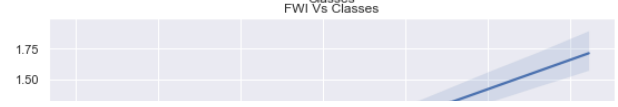

FFMC Vs Classes
Classes
Classes

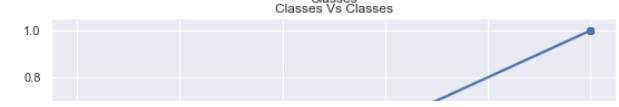

DMC Vs Classes
Classes
Classes

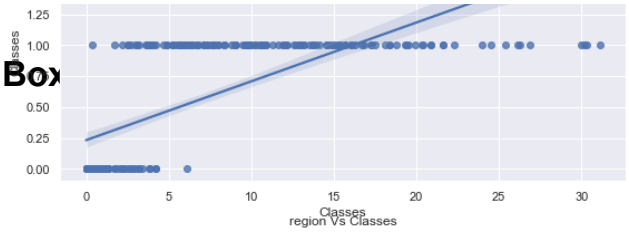

DC Vs Classes
Classes
Classes


ISI Vs Classes
Classes
Classes

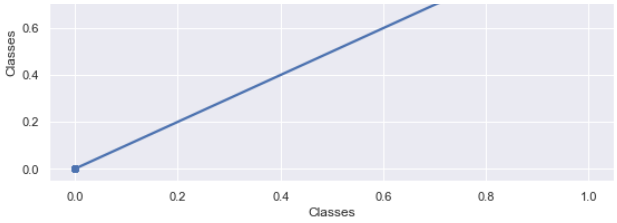

BUI Vs Classes
Classes
Classes

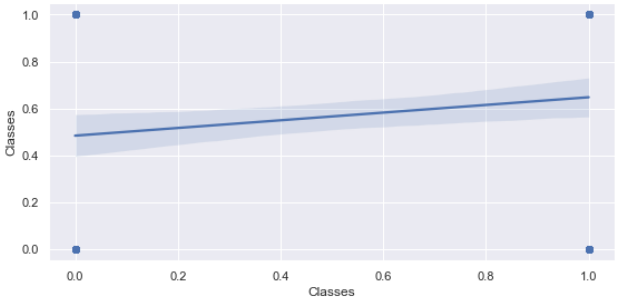

FWI Vs Classes


Classes Vs Classes

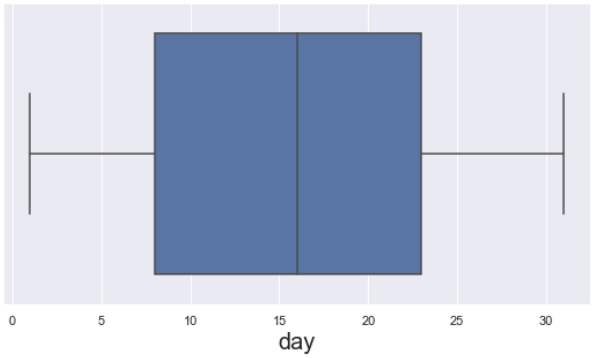


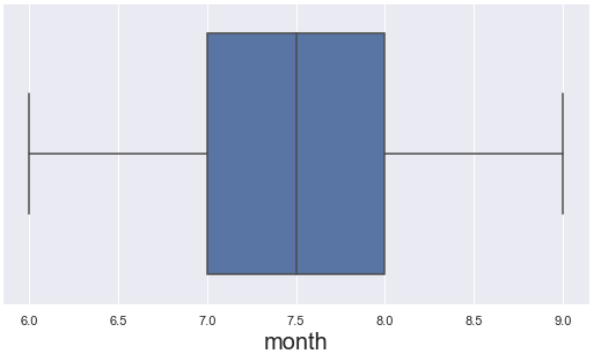
region Vs Classes

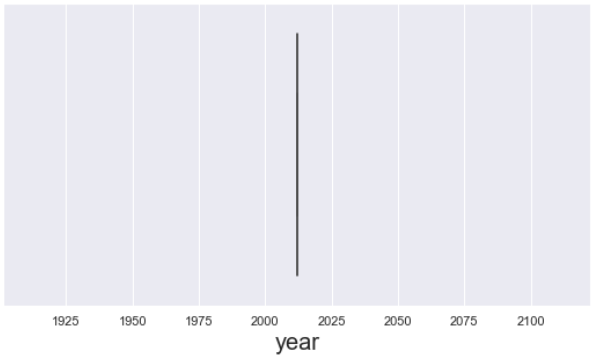


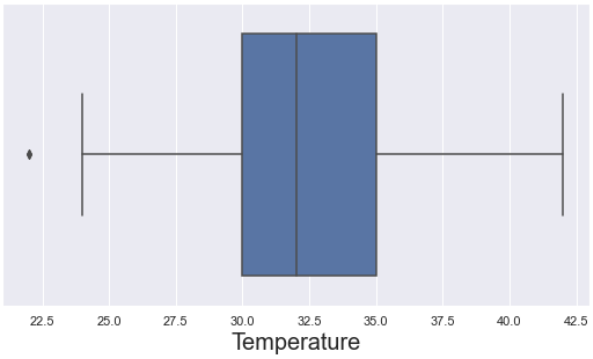


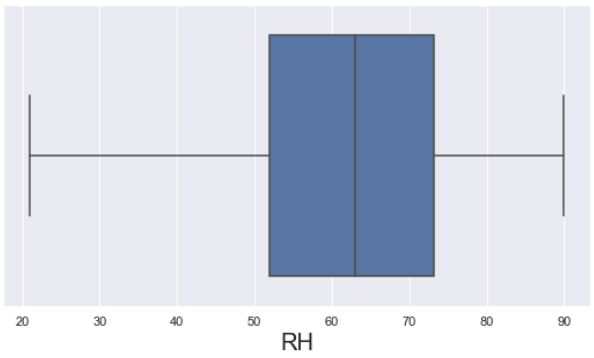
In [38]:
```
plt.figure(figsize=(20,45), facecolor='white')
plotnumber = 1

for column in data:
    if plotnumber<=15 :     # as there are 15 columns in the data
        ax = plt.subplot(8,2,plotnumber)
        sns.boxplot(data[column])
        plt.xlabel(column,fontsize=20)

    plotnumber+=1
plt.show()
```

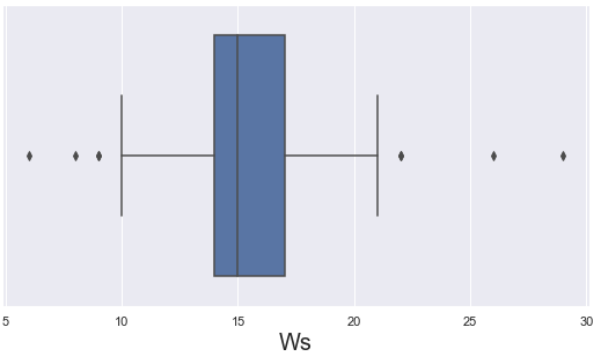

0
5
10
15
20
25
30
day

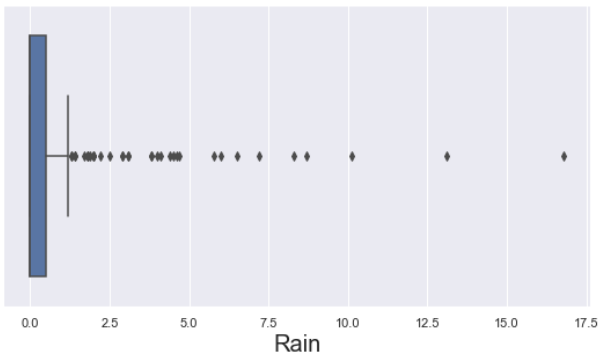

6.0
6.5
7.0
7.5
8.0
8.5
9.0
month

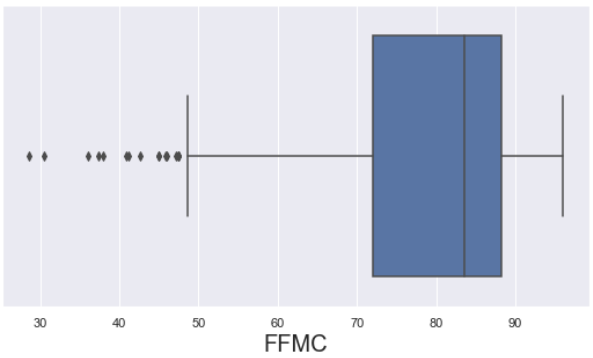

1925
1950
1975
2000
2025
2050
2075
2100
year

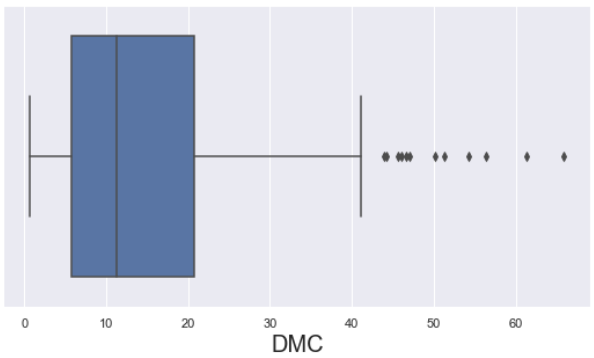

22.5
25.0
27.5
30.0
32.5
35.0
37.5
40.0
42.5
Temperature

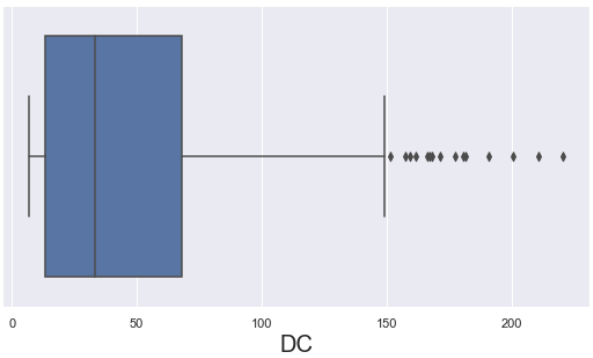

20
30
40
50
60
70
80
90
RH

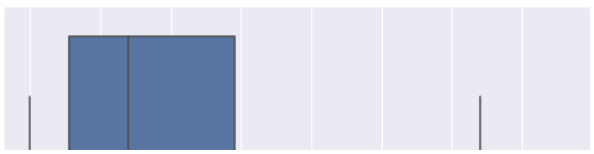

5
10
15
20
25
30
Ws

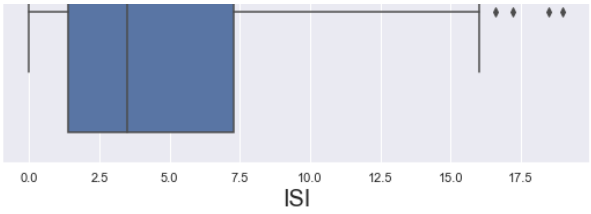

0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Rain

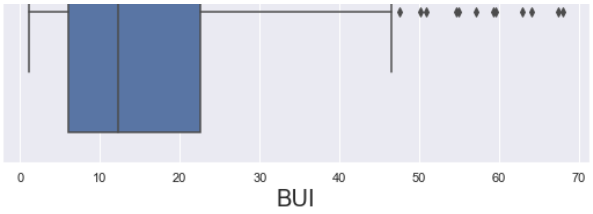

30
40
50
60
70
80
90
FFMC


0
10
20
30
40
50
60
DMC

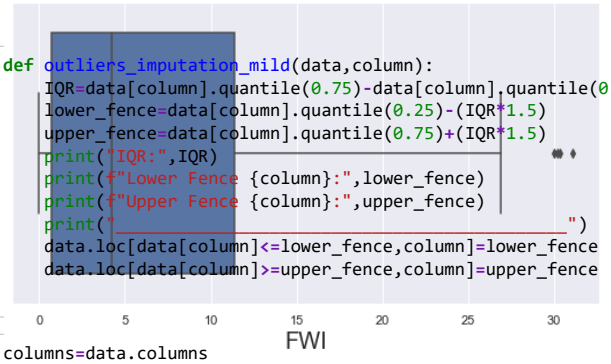

0
50
100
150
200
DC

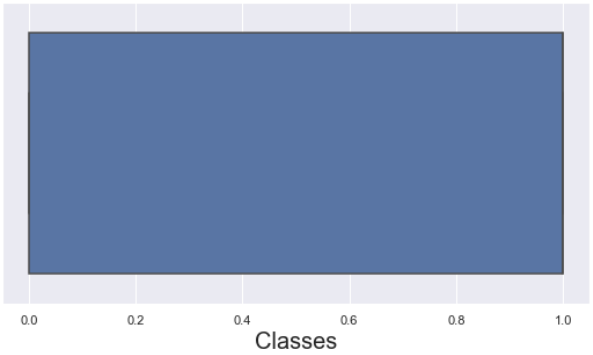




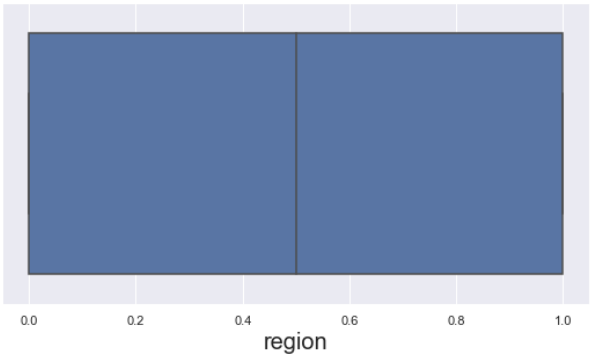


In [39]:
```
def outliers_imputation_mild(data,column):
    IQR=data[column].quantile(0.75)-data[column],quantile(0.25)
    lower_fence=data[column].quantile(0.25)-(IQR*1.5)
    upper_fence=data[column].quantile(0.75)+(IQR*1.5)
    print("IQR:",IQR)
    print(f"Lower Fence {column}:",lower_fence)
    print(f"Upper Fence {column}:",upper_fence)
    print("_____________________________________________")
    data.loc[data[column]<=lower_fence,column]=lower_fence
    data.loc[data[column]>=upper_fence,column]=upper_fence
```





In [40]:
```
columns=data.columns
```

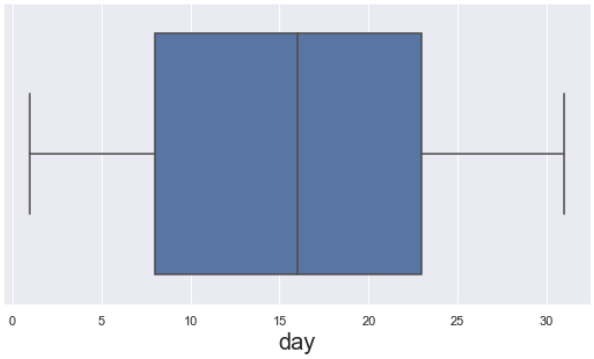


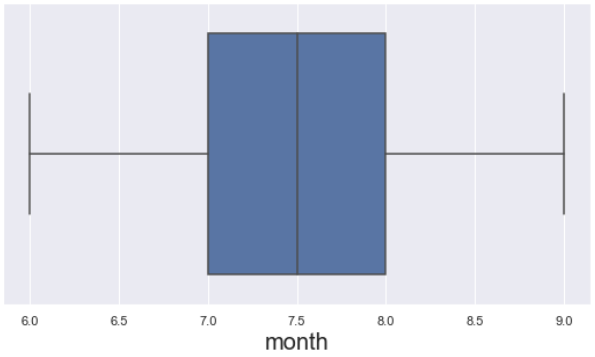
In [41]:
```
for col in columns:
    outliers_imputation_mild(data,col)
```

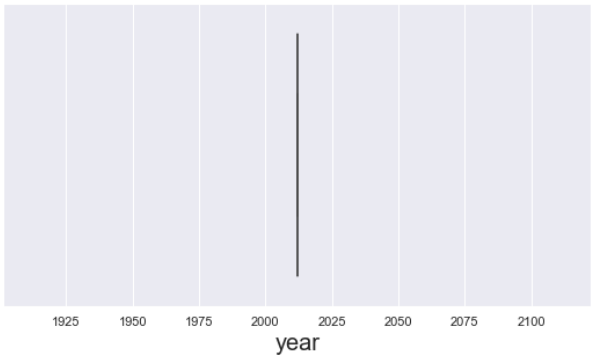
```
IQR: 15.0
Lower Fence day: -14.5
Upper Fence day: 45.5
_____________________________________________
IQR: 1.0
Lower Fence month: 5.5
Upper Fence month: 9.5
_____________________________________________
IQR: 0.0
Lower Fence year: 2012.0
Upper Fence year: 2012.0
_____________________________________________
IQR: 5.0
Lower Fence Temperature: 22.5
Upper Fence Temperature: 42.5
_____________________________________________
IQR: 21.25
Lower Fence RH: 20.125
Upper Fence RH: 105.125
_____________________________________________
IQR: 3.0
Lower Fence Ws: 9.5
Upper Fence Ws: 21.5
_____________________________________________
IQR: 0.5
Lower Fence Rain: -0.75
Upper Fence Rain: 1.25
_____________________________________________
IQR: 16.224999999999994
Lower Fence FFMC: 47.73750000000001
Upper Fence FFMC: 112.63749999999999
_____________________________________________
IQR: 14.95
Lower Fence DMC: -16.624999999999996
Upper Fence DMC: 43.175
_____________________________________________
IQR: 54.87500000000001
Lower Fence DC: -69.03750000000002
Upper Fence DC: 150.46250000000003
_____________________________________________
IQR: 5.9
Lower Fence ISI: -7.450000000000001
Upper Fence ISI: 16.150000000000002
_____________________________________________
IQR: 16.525
Lower Fence BUI: -18.787499999999998
Upper Fence BUI: 47.3125
_____________________________________________
IQR: 10.675
Lower Fence FWI: -15.312500000000004
Upper Fence FWI: 27.387500000000003
_____________________________________________
IQR: 1.0
Lower Fence Classes: -1.5
Upper Fence Classes: 2.5
_____________________________________________
IQR: 1.0
Lower Fence region: -1.5
Upper Fence region: 2.5
_____________________________________________
```

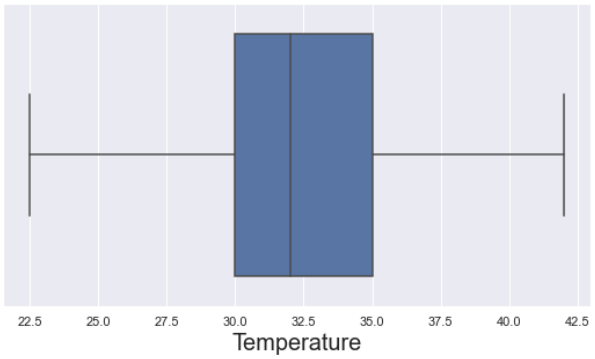
Rechecking the outliers after dropping it

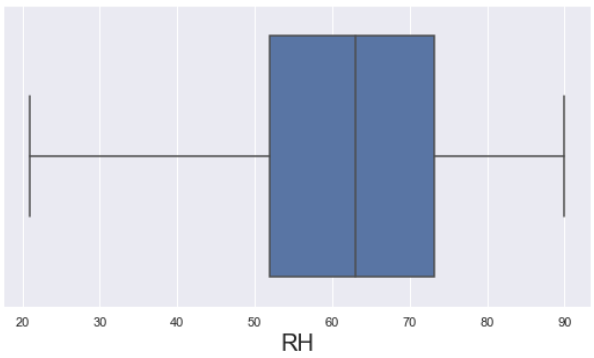
In [42]:
```
plt.figure(figsize=(20,45), facecolor='white')
plotnumber = 1

for column in data:
    if plotnumber<=15 :     # as there are 15 columns in the data
        ax = plt.subplot(8,2,plotnumber)
        sns.boxplot(data[column])
        plt.xlabel(column,fontsize=20)

    plotnumber+=1
plt.show()
```

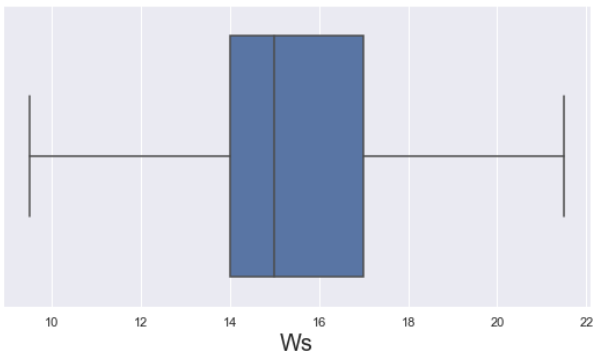

0
5
10
15
20
25
30
day

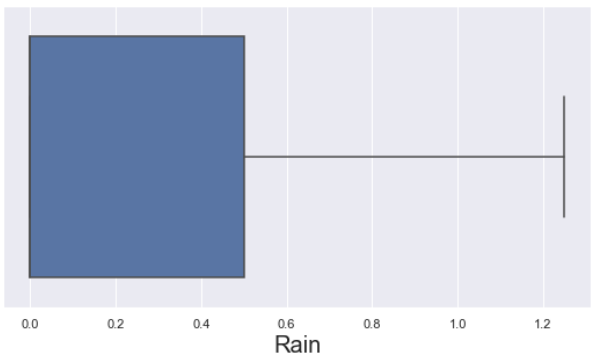

6.0
6.5
7.0
7.5
8.0
8.5
9.0
month

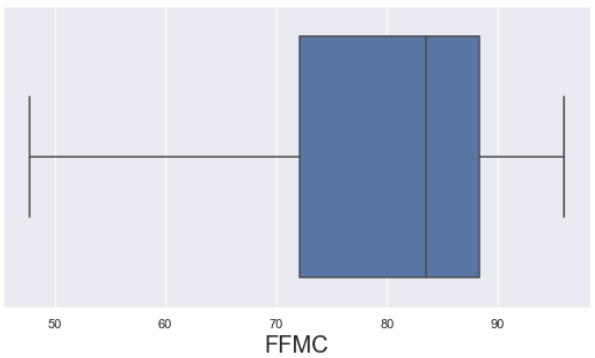

1925
1950
1975
2000
2025
2050
2075
2100
year

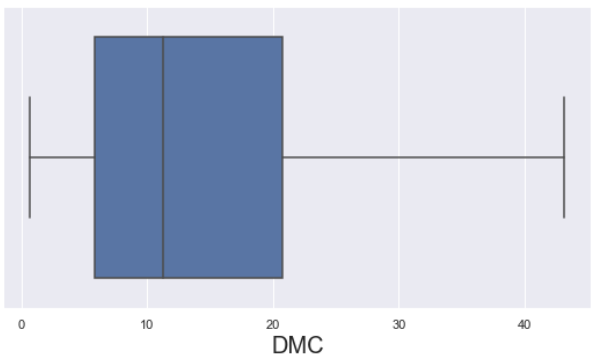

22.5
25.0
27.5
30.0
32.5
35.0
37.5
40.0
42.5
Temperature

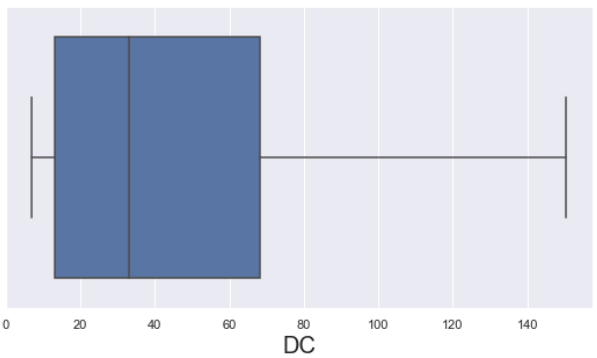

20
30
40
50
60
70
80
90
RH

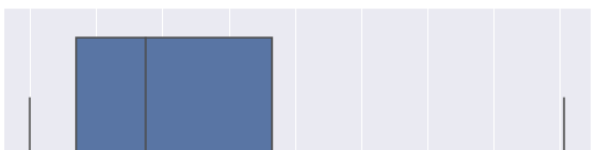

10
12
14
16
18
20
22
Ws

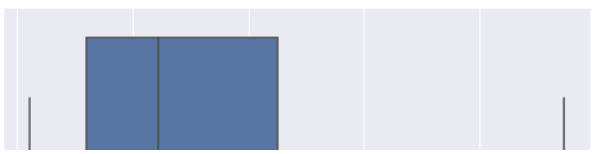

0.0
0.2
0.4
0.6
0.8
1.0
1.2
Rain

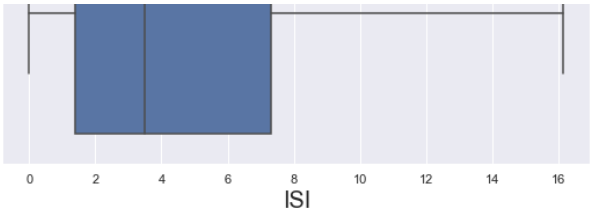

50
60
70
80
90
FFMC

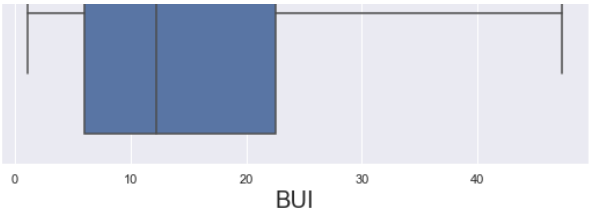

0
10
20
30
40
DMC


0
20
40
60
80
100
120
140
DC

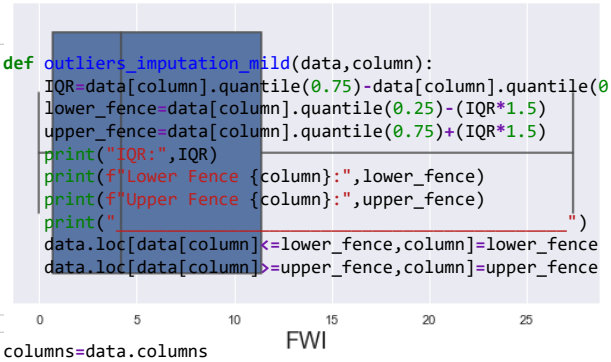


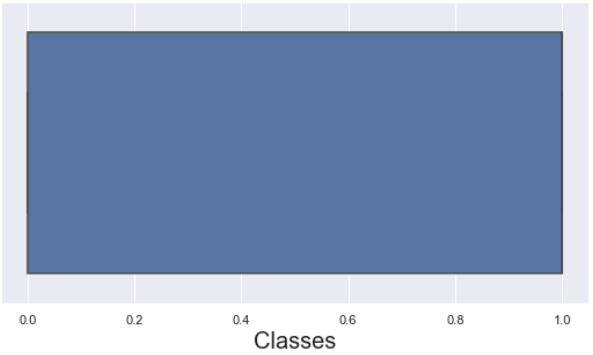




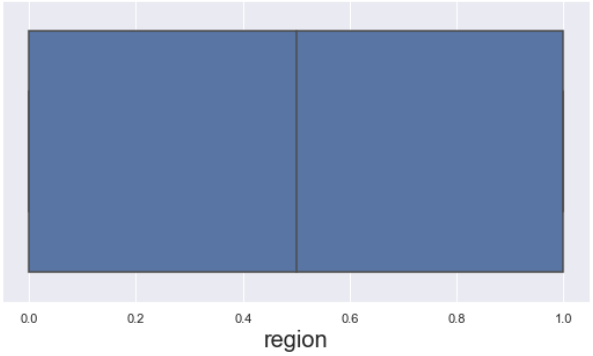


In [43]:
```
def outliers_imputation_mild(data,column):
    IQR=data[column].quantile(0.75)-data[column].quantile(0.25)
    lower_fence=data[column].quantile(0.25)-(IQR*1.5)
    upper_fence=data[column].quantile(0.75)+(IQR*1.5)
    print("IQR:",IQR)
    print(f"Lower Fence {column}:",lower_fence)
    print(f"Upper Fence {column}:",upper_fence)
    print("_____________________________________________")
    data.loc[data[column]<=lower_fence,column]=lower_fence
    data.loc[data[column]>=upper_fence,column]=upper_fence
```





In [44]:
```
columns=data.columns
```

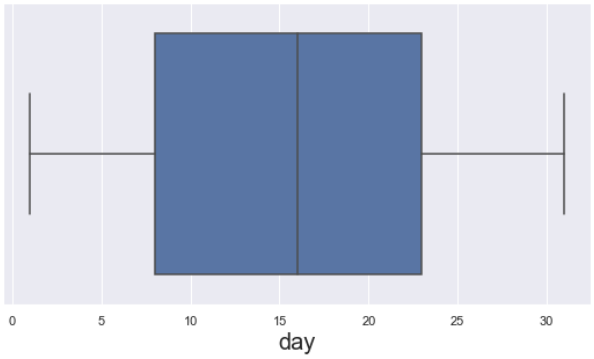


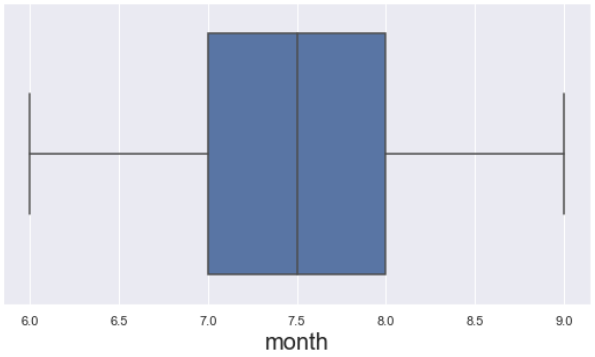
In [45]:
```
for col in columns:
    outliers_imputation_mild(data,col)
```

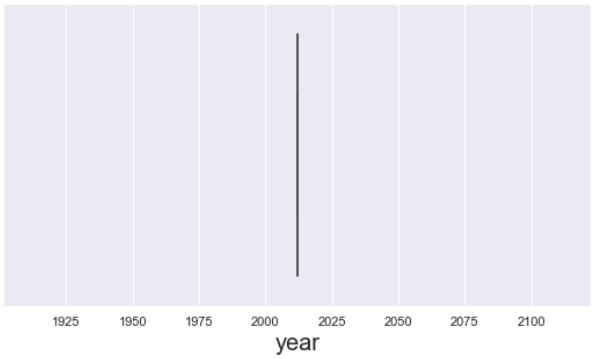
```
IQR: 15.0
Lower Fence day: -14.5
Upper Fence day: 45.5
_____________________________________________
IQR: 1.0
Lower Fence month: 5.5
Upper Fence month: 9.5
_____________________________________________
IQR: 0.0
Lower Fence year: 2012.0
Upper Fence year: 2012.0
_____________________________________________
IQR: 5.0
Lower Fence Temperature: 22.5
Upper Fence Temperature: 42.5
_____________________________________________
IQR: 21.25
Lower Fence RH: 20.125
Upper Fence RH: 105.125
_____________________________________________
IQR: 3.0
Lower Fence Ws: 9.5
Upper Fence Ws: 21.5
_____________________________________________
IQR: 0.5
Lower Fence Rain: -0.75
Upper Fence Rain: 1.25
_____________________________________________
IQR: 16.224999999999994
Lower Fence FFMC: 47.73750000000001
Upper Fence FFMC: 112.63749999999999
_____________________________________________
IQR: 14.95
Lower Fence DMC: -16.624999999999996
Upper Fence DMC: 43.175
_____________________________________________
IQR: 54.87500000000001
Lower Fence DC: -69.03750000000002
Upper Fence DC: 150.46250000000003
_____________________________________________
IQR: 5.9
Lower Fence ISI: -7.450000000000001
Upper Fence ISI: 16.150000000000002
_____________________________________________
IQR: 16.525
Lower Fence BUI: -18.787499999999998
Upper Fence BUI: 47.3125
_____________________________________________
IQR: 10.675
Lower Fence FWI: -15.312500000000004
Upper Fence FWI: 27.387500000000003
_____________________________________________
IQR: 1.0
Lower Fence Classes: -1.5
Upper Fence Classes: 2.5
_____________________________________________
IQR: 1.0
Lower Fence region: -1.5
Upper Fence region: 2.5
_____________________________________________
```

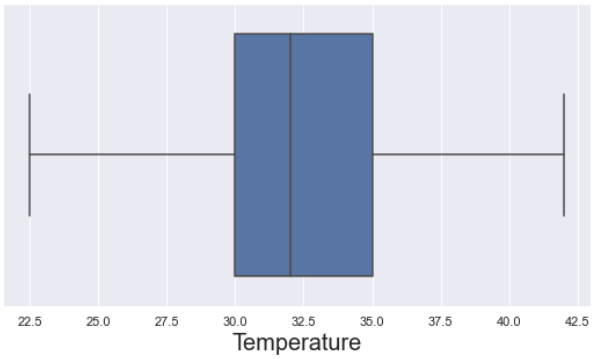
Checking the outliers after dropping it

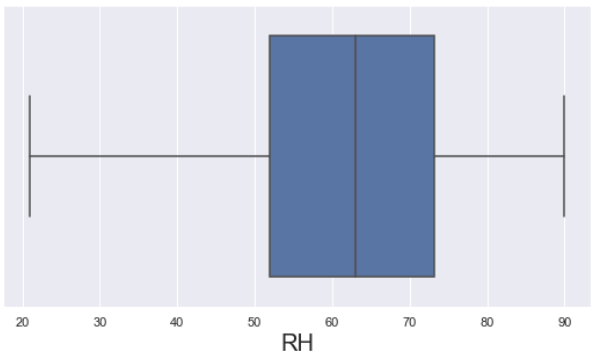
In [46]:
```
plt.figure(figsize=(20,45), facecolor='white')
plotnumber = 1

for column in data:
    if plotnumber<=15 :     # as there are 15 columns in the data
        ax = plt.subplot(8,2,plotnumber)
        sns.boxplot(data[column])
        plt.xlabel(column,fontsize=20)

    plotnumber+=1
plt.show()
```

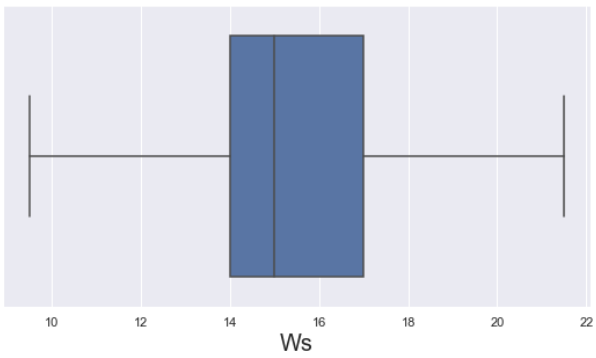

0
5
10
15
20
25
30
day

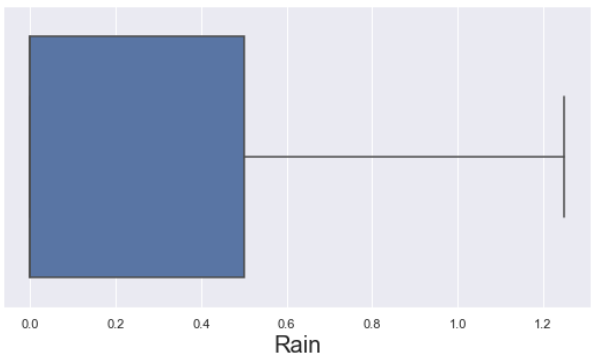

6.0
6.5
7.0
7.5
8.0
8.5
9.0
month

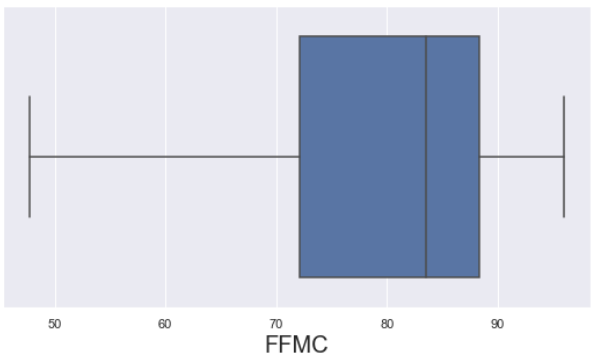

1925
1950
1975
2000
2025
2050
2075
2100
year

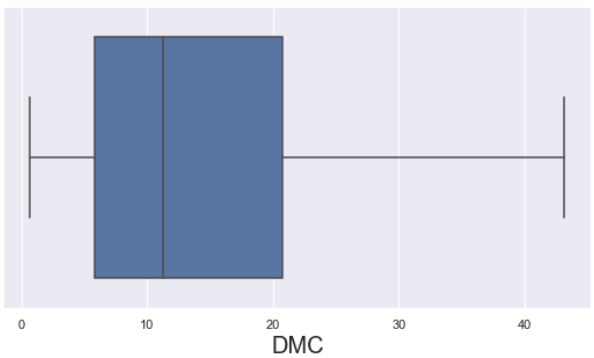

22.5
25.0
27.5
30.0
32.5
35.0
37.5
40.0
42.5
Temperature

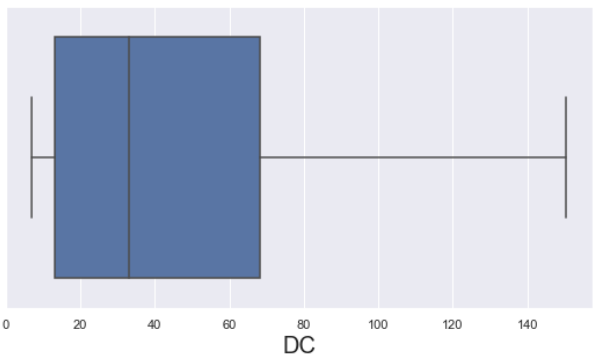

20
30
40
50
60
70
80
90
RH

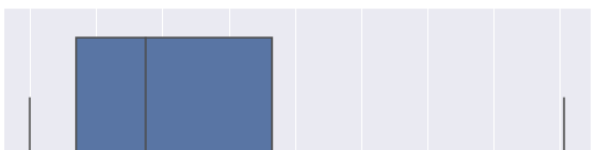

10
12
14
16
18
20
22
Ws

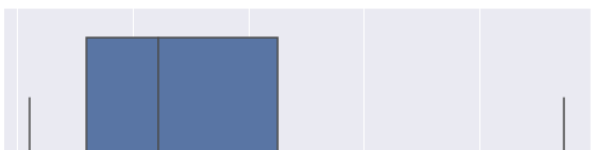

0.0
0.2
0.4
0.6
0.8
1.0
1.2
Rain

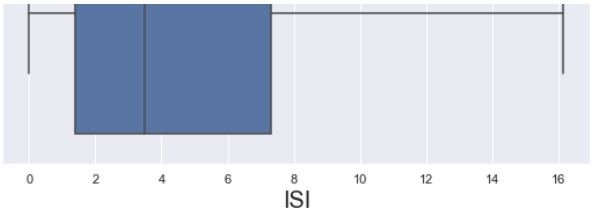

50
60
70
80
90
FFMC

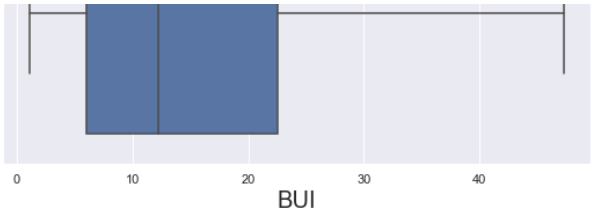

0
10
20
30
40
DMC


0
20
40
60
80
100
120
140
DC

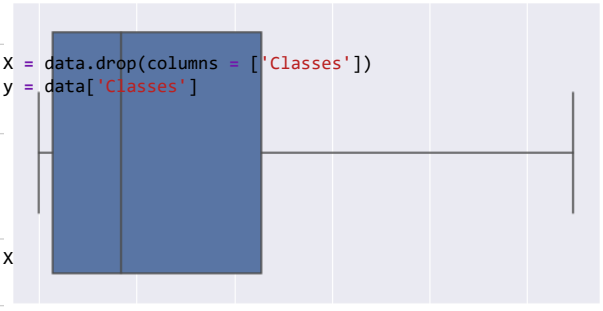








In [47]:
```
X = data.drop(columns = ['Classes'])
y = data['Classes']
```



In [48]:
```
X
```

Out[48]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	region
0	1.0	6.0	2012	29.0	57.0	18.0	0.00	65.7000	3.4	7.6	1.3	3.4	0.5	0.0
1	2.0	6.0	2012	29.0	61.0	13.0	1.25	64.4000	4.1	7.6	1.0	3.9	0.4	0.0
2	3.0	6.0	2012	26.0	82.0	21.5	1.25	47.7375	2.5	7.1	0.3	2.7	0.1	0.0
3	4.0	6.0	2012	25.0	89.0	13.0	1.25	47.7375	1.3	6.9	0.0	1.7	0.0	0.0
4	5.0	6.0	2012	27.0	77.0	16.0	0.00	64.8000	3.0	14.2	1.2	3.9	0.5	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
239	26.0	9.0	2012	30.0	65.0	14.0	0.00	85.4000	16.0	44.5	4.5	16.9	6.5	1.0
240	27.0	9.0	2012	28.0	87.0	15.0	1.25	47.7375	6.5	8.0	0.1	6.2	0.0	1.0
241	28.0	9.0	2012	27.0	87.0	21.5	0.50	47.7375	3.5	7.9	0.4	3.4	0.2	1.0
242	29.0	9.0	2012	24.0	54.0	18.0	0.10	79.7000	4.3	15.2	1.7	5.1	0.7	1.0
243	30.0	9.0	2012	24.0	64.0	15.0	0.20	67.3000	3.8	16.5	1.2	4.8	0.5	1.0

244 rows × 14 columns

Dependent Features

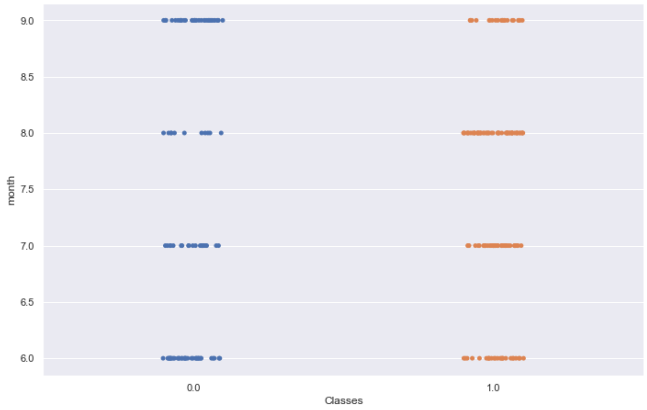
In [49]:
```
y
```

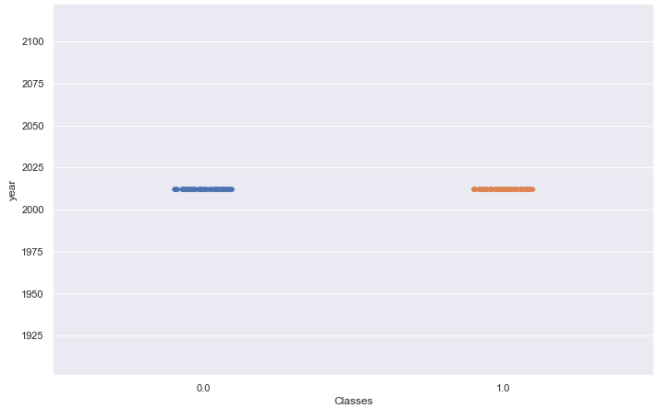
Out[49]:
```
0      0.0
1      0.0
2      0.0
3      0.0
4      0.0
      ...
239    1.0
240    0.0
241    0.0
242    0.0
243    0.0
Name: Classes, Length: 244, dtype: float64
```

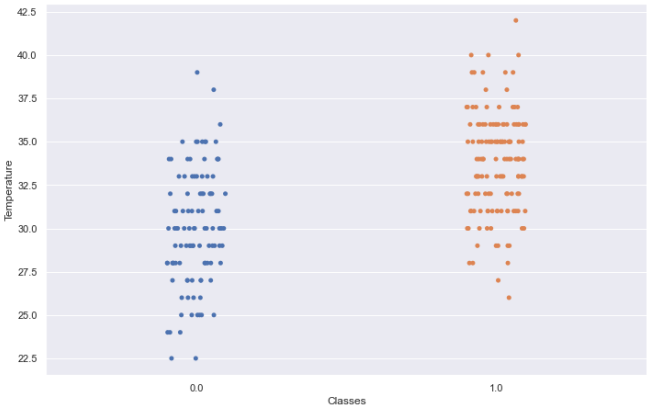
Visualizing the relationship between our independent and dependent Features

In [50]:
```
plt.figure(figsize=(20,50), facecolor='white')
plotnumber = 1

for column in X:
    if plotnumber<=15 :
        ax = plt.subplot(8,2,plotnumber)
        sns.stripplot(y,X[column])
    plotnumber+=1
plt.tight_layout()
```

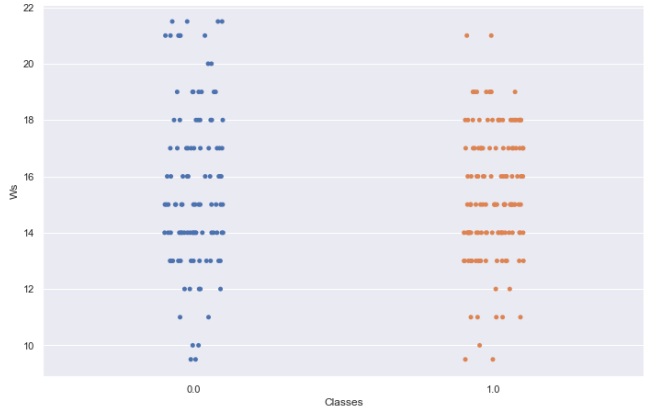

day
Classes
0.0
1.0

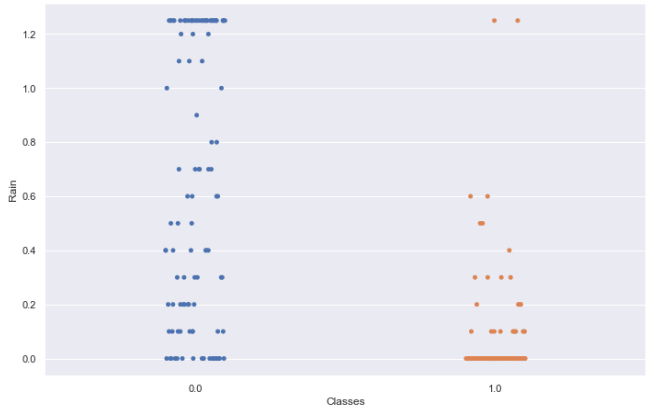

month
Classes
0.0
1.0


year
Classes
0.0
1.0


Temperature
Classes
0.0
1.0


RH
Classes
0.0
1.0

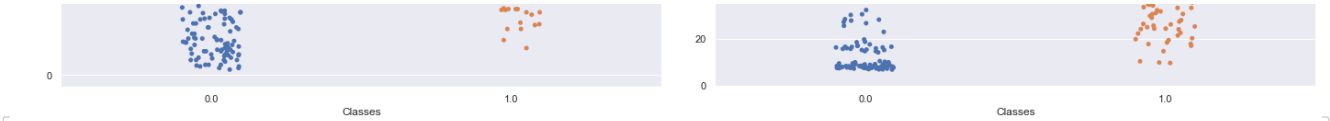

Ws
Classes
0.0
1.0


Rain
Classes
0.0
1.0

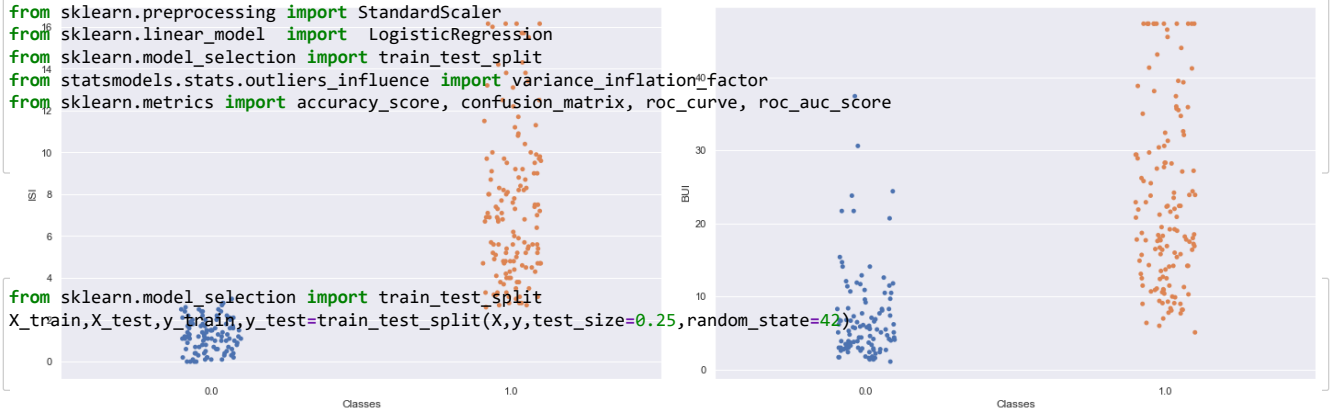

FFMC
Classes
0.0
1.0


DMC

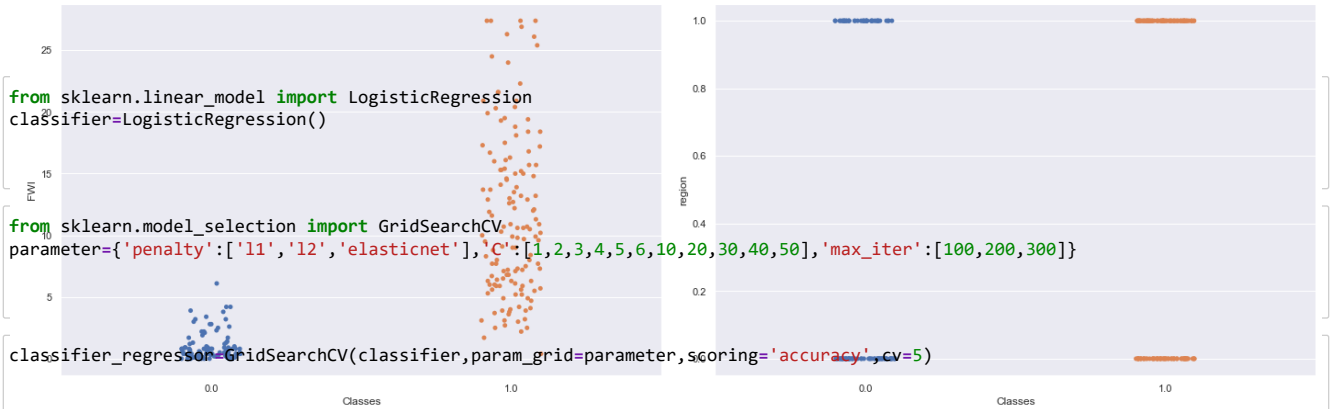

DC



In [51]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model  import  LogisticRegression
from sklearn.model_selection import train_test_split
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, roc_auc_score
```



In [52]:
```
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 [53]:
```
from sklearn.linear_model import LogisticRegression
classifier=LogisticRegression()
```

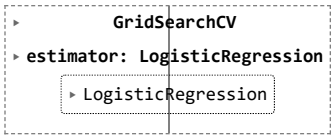
In [54]:
```
from sklearn.model_selection import GridSearchCV
parameter={'penalty':['l1','l2','elasticnet'],'C':[1,2,3,4,5,6,10,20,30,40,50],'max_iter':[100,200,300]}
```

In [55]:
```
classifier_regressor=GridSearchCV(classifier,param_grid=parameter,scoring='accuracy',cv=5)
```

Standardizing or Feature Selection

In [56]:
```
classifier_regressor.fit(X_train,y_train)
```

Out[56]:



In [57]:
```
print(classifier_regressor.best_params_)  ## Best parameter
```

```
{'C': 1, 'max_iter': 100, 'penalty': 'l2'}
```

In [58]:
```
print(classifier_regressor.best_score_)  ## Best Score
```

```
0.9725225225225225
```

prediction

In [59]:
```
y_pred = classifier_regressor.predict(X_test)
```

In [60]:
```
y_pred
```

Out[60]:
```
array([1., 1., 1., 1., 1., 0., 1., 0., 0., 0., 0., 0., 1., 0., 1., 1., 1.,
       0., 1., 1., 0., 0., 1., 0., 1., 0., 1., 1., 0., 1., 1., 0., 1., 1.,
       0., 0., 0., 1., 0., 0., 1., 1., 0., 0., 1., 1., 1., 0., 1., 1., 1.,
       0., 1., 1., 1., 1., 1., 0., 0., 1., 1.])
```

acuurracy score

In [61]:
```
from sklearn.metrics import accuracy_score,classification_report
score=accuracy_score(y_pred,y_test)
print(score)
```

```
0.9508196721311475
```

Classification Report

In [62]:
```
print(classification_report(y_pred,y_test))
```

```
              precision    recall  f1-score   support

         0.0       0.96      0.92      0.94        25
         1.0       0.95      0.97      0.96        36

    accuracy                           0.95        61
   macro avg       0.95      0.95      0.95        61
weighted avg       0.95      0.95      0.95        61
```

Performance Metrics

1. Confusion Metrics

In [63]:
```
conf_mat=confusion_matrix(y_pred,y_test)
```

In [64]:
```
conf_mat
```

Out[64]:
```
array([[23,  2],
       [ 1, 35]], dtype=int64)
```

In [65]:
```
true_positive = conf_mat[0][0]
false_positive = conf_mat[0][1]
false_negative = conf_mat[1][0]
true_negative = conf_mat[1][1]
```

formula for Accuracy

In [66]:
```
Accuracy = (true_positive + true_negative) / (true_positive +false_positive + false_negative + true_negative)
Accuracy
```

Out[66]:
```
0.9508196721311475
```

2. Precison

In [67]:
```
Precision = true_positive/(true_positive+false_positive)
Precision
```

Out[67]:
```
0.92
```

3. Recall

In [68]:
```
Recall = true_positive/(true_positive+false_negative)
Recall
```

Out[68]: 0.9583333333333334

3. F1 Score

In [69]:
```
F1_Score = 2*(Recall * Precision) / (Recall + Precision)
F1_Score
```

Out[69]: 0.9387755102040817

Creating Imbalance dataset from the original balanced dataset

In [70]:
```
df.head()
```

Out[70]:

	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

In [71]:
```
df.shape
```

Out[71]: (244, 15)

In [72]:
```
### Creating imbalance
### 1. splitting data in 90:10 percent ratio using train test split
X1 = pd.DataFrame(df, columns = ['day', 'month','year','Temperature','RH','Ws','Rain','FFMC','DMC','DC','ISI','BUI','FWI','regior
y1=pd.DataFrame(df,columns = ['Classes'])
```

In [73]:
```
X_train_imb, X_test_imb, y_train_imb, y_test_imb = train_test_split(X1, y1, test_size=0.10, random_state=17)
```

In [74]:
```
## Both will have same shape
X_train_imb.shape, y_train_imb.shape
```

Out[74]: ((219, 14), (219, 1))

In [75]:
```
## Both will have same shape
X_test_imb.shape, y_test_imb.shape
```

Out[75]: ((25, 14), (25, 1))

Replacing all values as 1 in y_train and all values as zero in y_test to create imbalance

In [76]:
```
y_train_imb=y_train_imb.replace(0,1)
y_train_imb.head()
```

Out[76]:

	Classes
156	1
183	1
11	1
75	1
130	1

In [77]:
```
y_test_imb=y_test_imb.replace(1,0)
y_test_imb.head()
```

Out[77]:

	Classes
48	0
216	0
101	0
38	0
86	0

In [78]:
```
X_train_imb.head()
```

Out[78]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	region
156	5	7	2012	34	45	18	0.0	90.5	18.7	46.4	11.3	18.7	15.0	1
183	1	8	2012	38	52	14	0.0	78.3	4.4	10.5	2.0	4.4	0.8	1
11	12	6	2012	26	81	19	0.0	84.0	13.8	61.4	4.8	17.7	7.1	0
75	15	8	2012	36	55	13	0.3	82.4	15.6	92.5	3.7	22.0	6.3	0
130	9	6	2012	27	59	18	0.1	78.1	8.5	14.7	2.4	8.3	1.9	1

In [79]:
```
### Combining X_train_imb and y_train_imb
train_imb=X_train_imb.join(pd.DataFrame(y_train_imb))
train_imb.head()
```

Out[79]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	region	Classes
156	5	7	2012	34	45	18	0.0	90.5	18.7	46.4	11.3	18.7	15.0	1	1
183	1	8	2012	38	52	14	0.0	78.3	4.4	10.5	2.0	4.4	0.8	1	1
11	12	6	2012	26	81	19	0.0	84.0	13.8	61.4	4.8	17.7	7.1	0	1
75	15	8	2012	36	55	13	0.3	82.4	15.6	92.5	3.7	22.0	6.3	0	1
130	9	6	2012	27	59	18	0.1	78.1	8.5	14.7	2.4	8.3	1.9	1	1

In [80]:
```
### Combining X_test_imb and y_test_imb
test_imb=X_test_imb.join(pd.DataFrame(y_test_imb))
test_imb.head()
```

Out[80]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	region	Classes
48	19	7	2012	35	59	17	0.0	88.1	12.0	52.8	7.7	18.2	10.9	0	0
216	3	9	2012	28	75	16	0.0	82.2	4.4	24.3	3.3	6.0	2.5	1	0
101	10	9	2012	33	73	12	1.8	59.9	2.2	8.9	0.7	2.7	0.3	0	0
38	9	7	2012	32	68	14	1.4	66.6	7.7	9.2	1.1	7.4	0.6	0	0
86	26	8	2012	31	78	18	0.0	85.8	45.6	190.6	4.7	57.1	13.7	0	0

In [81]:
```
### the shape of imbalanced Data
train_imb.shape, test_imb.shape
```

Out[81]: ((219, 15), (25, 15))

In [82]:
```
### Combining train_imb dataset and test_imb dataset into data_imb dataset
df_imb=pd.concat([train_imb, test_imb], ignore_index=True, sort=False)
df_imb.head()
```

Out[82]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	region	Classes
0	5	7	2012	34	45	18	0.0	90.5	18.7	46.4	11.3	18.7	15.0	1	1
1	1	8	2012	38	52	14	0.0	78.3	4.4	10.5	2.0	4.4	0.8	1	1
2	12	6	2012	26	81	19	0.0	84.0	13.8	61.4	4.8	17.7	7.1	0	1
3	15	8	2012	36	55	13	0.3	82.4	15.6	92.5	3.7	22.0	6.3	0	1
4	9	6	2012	27	59	18	0.1	78.1	8.5	14.7	2.4	8.3	1.9	1	1

In [83]:
```
df_imb.shape
```

Out[83]: (244, 15)

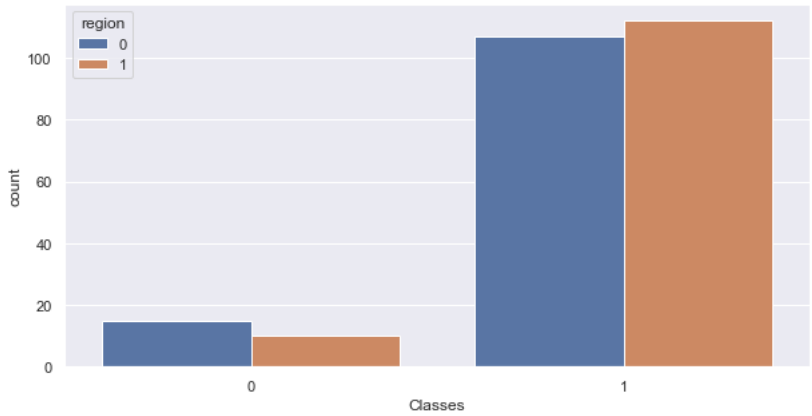
Checking the imbalancing

In [84]:
```
df_imb.Classes.value_counts()
```

Out[84]:
```
1    219
0     25
Name: Classes, dtype: int64
```

In [85]:
```
## 0 is 'Bejaia' and 1 is 'Sidi Bel-abbes region'
plt.figure(figsize=(10,5))
sns.countplot(data=df_imb,x='Classes',hue='region')
```

Out[85]: <AxesSubplot:xlabel='Classes', ylabel='count'>



Logistic Regression on imbalanced Dataset

In [86]:
```
df_imb.head()
```

Out[86]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	region	Classes
0	5	7	2012	34	45	18	0.0	90.5	18.7	46.4	11.3	18.7	15.0	1	1
1	1	8	2012	38	52	14	0.0	78.3	4.4	10.5	2.0	4.4	0.8	1	1
2	12	6	2012	26	81	19	0.0	84.0	13.8	61.4	4.8	17.7	7.1	0	1
3	15	8	2012	36	55	13	0.3	82.4	15.6	92.5	3.7	22.0	6.3	0	1
4	9	6	2012	27	59	18	0.1	78.1	8.5	14.7	2.4	8.3	1.9	1	1

Separating Independent and Dependent feature

In [87]:
```
X1 = df_imb.drop(columns = ['Classes'])
y1 = df_imb['Classes']
```

In [88]:
```
X1
```

Out[88]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	region
0	5	7	2012	34	45	18	0.0	90.5	18.7	46.4	11.3	18.7	15.0	1
1	1	8	2012	38	52	14	0.0	78.3	4.4	10.5	2.0	4.4	0.8	1
2	12	6	2012	26	81	19	0.0	84.0	13.8	61.4	4.8	17.7	7.1	0
3	15	8	2012	36	55	13	0.3	82.4	15.6	92.5	3.7	22.0	6.3	0
4	9	6	2012	27	59	18	0.1	78.1	8.5	14.7	2.4	8.3	1.9	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
239	26	8	2012	33	37	16	0.0	92.2	61.3	167.2	13.1	64.0	30.3	1
240	1	6	2012	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	0.5	0
241	2	9	2012	28	67	19	0.0	75.4	2.9	16.3	2.0	4.0	0.8	1
242	11	8	2012	35	63	13	0.0	88.9	21.7	77.0	7.1	25.5	12.1	0
243	9	8	2012	39	43	12	0.0	91.7	16.5	30.9	9.6	16.4	12.7	1

244 rows × 14 columns

In [89]:
```
y1
```

Out[89]:
```
0      1
1      1
2      1
3      1
4      1
      ..
239    0
240    0
241    0
242    0
243    0
Name: Classes, Length: 244, dtype: int32
```

Handling Imbalanced Dataset

Upsampling

In [90]:
```
pip install imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in c:\users\rakhi amberiya\anaconda3\lib\site-packages (0.9.1)
Requirement already satisfied: scipy>=1.3.2 in c:\users\rakhi amberiya\anaconda3\lib\site-packages (from imbalanced-learn) (1.
7.1)
Requirement already satisfied: scikit-learn>=1.1.0 in c:\users\rakhi amberiya\anaconda3\lib\site-packages (from imbalanced-lear
n) (1.1.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\rakhi amberiya\anaconda3\lib\site-packages (from imbalanced-lea
rn) (2.2.0)
Requirement already satisfied: numpy>=1.17.3 in c:\users\rakhi amberiya\anaconda3\lib\site-packages (from imbalanced-learn) (1.
20.3)
Requirement already satisfied: joblib>=1.0.0 in c:\users\rakhi amberiya\anaconda3\lib\site-packages (from imbalanced-learn) (1.
1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [91]:
```
from imblearn.combine import SMOTETomek
```

In [92]:
```
smk=SMOTETomek()
smk
```

Out[92]:
```
SMOTETomek
SMOTETomek()
```

In [93]:
```
X_bal,y_bal=smk.fit_resample(X1,y1)
```

In [94]:
```
X_bal.head()
```

Out[94]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	region
0	5	7	2012	34	45	18	0.0	90.5	18.7	46.4	11.3	18.7	15.0	1
1	1	8	2012	38	52	14	0.0	78.3	4.4	10.5	2.0	4.4	0.8	1
2	12	6	2012	26	81	19	0.0	84.0	13.8	61.4	4.8	17.7	7.1	0
3	15	8	2012	36	55	13	0.3	82.4	15.6	92.5	3.7	22.0	6.3	0
4	9	6	2012	27	59	18	0.1	78.1	8.5	14.7	2.4	8.3	1.9	1

In [95]:
```
y_bal.head()
```

Out[95]:
```
0    1
1    1
2    1
3    1
4    1
Name: Classes, dtype: int32
```

In [96]:
```
X_bal.shape,y_bal.shape
```

Out[96]:
```
((420, 14), (420,))
```

In [97]:
```
## Creating Balanced data from imbalanced data
data_bal=X_bal.join(pd.DataFrame(y_bal))
data_bal.head()
```

Out[97]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	region	Classes
0	5	7	2012	34	45	18	0.0	90.5	18.7	46.4	11.3	18.7	15.0	1	1
1	1	8	2012	38	52	14	0.0	78.3	4.4	10.5	2.0	4.4	0.8	1	1
2	12	6	2012	26	81	19	0.0	84.0	13.8	61.4	4.8	17.7	7.1	0	1
3	15	8	2012	36	55	13	0.3	82.4	15.6	92.5	3.7	22.0	6.3	0	1
4	9	6	2012	27	59	18	0.1	78.1	8.5	14.7	2.4	8.3	1.9	1	1

EDA on balanced Dataset

In [98]:
```
data_bal.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 420 entries, 0 to 419
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   day          420 non-null    int32  
 1   month        420 non-null    int32  
 2   year         420 non-null    int32  
 3   Temperature  420 non-null    int32  
 4   RH           420 non-null    int32  
 5   Ws           420 non-null    int32  
 6   Rain         420 non-null    float64
 7   FFMC         420 non-null    float64
 8   DMC          420 non-null    float64
 9   DC           420 non-null    float64
 10  ISI          420 non-null    float64
 11  BUI          420 non-null    float64
 12  FWI          420 non-null    float64
 13  region       420 non-null    int32  
 14  Classes      420 non-null    int32  
dtypes: float64(7), int32(8)
memory usage: 36.2 KB
```

Statistical analysis on Balanced Dataset

In [99]:
```
data_bal.describe()
```

Out[99]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FW
count	420.000000	420.000000	420.0	420.000000	420.000000	420.000000	420.000000	420.000000	420.000000	420.000000	420.000000	420.000000	420.0000
mean	15.092857	7.495238	2012.0	32.159524	62.333333	15.407143	0.548422	78.817661	15.895861	55.682177	4.795001	18.357948	7.4072
std	8.654793	1.014208	0.0	3.285286	14.405426	2.614161	1.566067	12.663700	13.603118	50.871848	3.857084	15.648845	7.2995
min	1.000000	6.000000	2012.0	22.000000	21.000000	6.000000	0.000000	28.600000	0.700000	6.900000	0.000000	1.100000	0.0000
25%	7.000000	7.000000	2012.0	30.000000	52.000000	14.000000	0.000000	72.589067	5.065853	15.796873	1.500000	5.740116	0.7622
50%	15.000000	7.000000	2012.0	32.000000	64.000000	15.000000	0.000789	83.769216	12.586255	38.621881	3.964255	15.400000	5.2000
75%	23.000000	8.000000	2012.0	34.000000	73.000000	17.000000	0.400000	88.002343	21.375283	85.363592	7.097870	25.064714	13.2030
max	31.000000	9.000000	2012.0	42.000000	90.000000	29.000000	16.800000	96.000000	65.900000	220.400000	19.000000	68.000000	30.3000

In [100]:
```
data_bal.corr()
```

Out[100]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	region
day	1.000000	-0.124342	NaN	0.176670	-0.131013	0.140204	-0.086283	0.285885	0.593130	0.631302	0.284739	0.620686	0.465136	-0.003697
month	-0.124342	1.000000	NaN	-0.129777	0.098939	-0.034824	0.019090	-0.010194	-0.037364	0.024498	0.020579	-0.021258	0.016443	0.013299
year	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Temperature	0.176670	-0.129777	NaN	1.000000	-0.681877	-0.277972	-0.271193	0.660612	0.409921	0.334932	0.636623	0.393433	0.561881	0.259914
RH	-0.131013	0.098939	NaN	-0.681877	1.000000	0.216113	0.176387	-0.608730	-0.319004	-0.189483	-0.674833	-0.277885	-0.539103	-0.383485
Ws	0.140204	-0.034824	NaN	-0.277972	0.216113	1.000000	0.108506	-0.064579	0.144791	0.238695	0.052685	0.186180	0.111443	-0.147211
Rain	-0.086283	0.019090	NaN	-0.271193	0.176387	0.108506	1.000000	-0.525023	-0.252531	-0.277126	-0.328615	-0.266252	-0.305373	0.024886
FFMC	0.285885	-0.010194	NaN	0.660612	-0.608730	-0.064579	-0.525023	1.000000	0.596327	0.543537	0.770326	0.594986	0.721458	0.178082
DMC	0.593130	-0.037364	NaN	0.409921	-0.319004	0.144791	-0.252531	0.596327	1.000000	0.894667	0.635752	0.985236	0.856695	0.083091
DC	0.631302	0.024498	NaN	0.334932	-0.189483	0.238695	-0.277126	0.543537	0.894667	1.000000	0.531455	0.952730	0.771875	-0.120451
ISI	0.284739	0.020579	NaN	0.636623	-0.674833	0.052685	-0.328615	0.770326	0.635752	0.531455	1.000000	0.616407	0.912625	0.220542
BUI	0.620686	-0.021258	NaN	0.393433	-0.277885	0.186180	-0.266252	0.594986	0.985236	0.952730	0.616407	1.000000	0.849671	0.003151
FWI	0.465136	0.016443	NaN	0.561881	-0.539103	0.111443	-0.305373	0.721458	0.856695	0.771875	0.912625	0.849671	1.000000	0.124718
region	-0.003697	0.013299	NaN	0.259914	-0.383485	-0.147211	0.024886	0.178082	0.083091	-0.120451	0.220542	0.003151	0.124718	1.000000
Classes	0.103286	-0.028205	NaN	-0.029749	-0.029786	0.031916	0.194322	-0.117539	-0.134454	-0.178847	-0.031582	-0.158627	-0.098437	0.361158

In [101]:
```
data_bal.cov()
```

Out[101]:

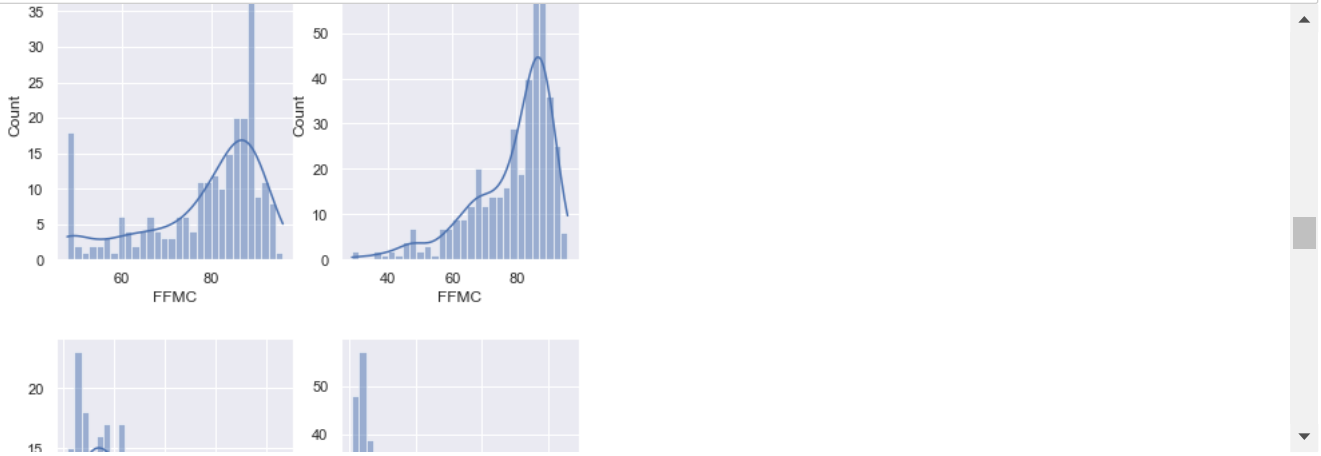
	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	
day	74.905438	-1.091442	0.0	5.023338	-16.334129	3.172128	-1.169473	31.333458	69.830492	277.953054	9.505219	84.064192	29.3
month	-1.091442	1.028617	0.0	-0.432413	1.445505	-0.092329	0.030321	-0.130925	-0.515484	1.263941	0.080502	-0.337392	0.1
year	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
Temperature	5.023338	-0.432413	0.0	10.793107	-32.270485	-2.387300	-1.395280	27.484003	18.319407	55.976757	8.067047	20.226782	13.4
RH	-16.334129	1.445505	0.0	-32.270485	207.516309	8.138425	3.979261	-111.048243	-62.511711	-138.858801	-37.495717	-62.643082	-56.6
Ws	3.172128	-0.092329	0.0	-2.387300	8.138425	6.833839	0.444217	-2.137885	5.148874	31.743375	0.531223	7.616371	2.1
Rain	-1.169473	0.030321	0.0	-1.395280	3.979261	0.444217	2.452564	-10.412369	-5.379762	-22.078298	-1.984982	-6.525084	-3.4
FFMC	31.333458	-0.130925	0.0	27.484003	-111.048243	-2.137885	-10.412369	160.369292	102.726804	350.160736	37.626563	117.909672	66.6
DMC	69.830492	-0.515484	0.0	18.319407	-62.511711	5.148874	-5.379762	102.726804	185.044831	619.123635	33.356876	209.730260	85.0
DC	277.953054	1.263941	0.0	55.976757	-138.858801	31.743375	-22.078298	350.160736	619.123635	2587.944967	104.280538	758.454658	286.6
ISI	9.505219	0.080502	0.0	8.067047	-37.495717	0.531223	-1.984982	37.626563	33.356876	104.280538	14.877100	37.205633	25.6
BUI	84.064192	-0.337392	0.0	20.226782	-62.643082	7.616371	-6.525084	117.909672	209.730260	758.454658	37.205633	244.886355	97.0
FWI	29.385554	0.121730	0.0	13.474557	-56.688500	2.126575	-3.490910	66.691279	85.067146	286.630183	25.694997	97.057835	53.2
region	-0.015206	0.006410	0.0	0.405796	-2.625298	-0.182884	0.018521	1.071730	0.537149	-2.912009	0.404255	0.023436	0.4
Classes	0.447494	-0.014320	0.0	-0.048926	-0.214797	0.041766	0.152342	-0.745129	-0.915586	-4.554575	-0.060979	-1.242647	-0.3

In [102]:
```
num_bal_col=[feature for feature in data_bal.columns if data_bal[feature].dtype != 'O']
num_bal_col
```

Out[102]:
```
['day',
 'month',
 'year',
 'Temperature',
 'RH',
 'Ws',
 'Rain',
 'FFMC',
 'DMC',
 'DC',
 'ISI',
 'BUI',
 'FWI',
 'region',
 'Classes']
```

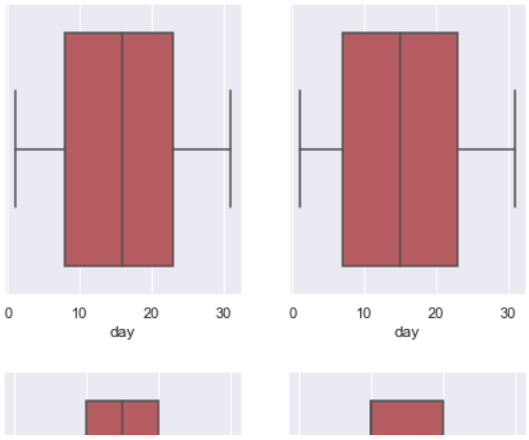
Compairing the feature for Original and Balanced Dataset

In [103]:
```
for i in num_col:
    plt.figure(figsize=(7,4))
    plt.subplot(121)
    sns.histplot(data=data,x=i,kde=True,bins=30)


    plt.subplot(122)
    sns.histplot(data=data_bal,x=i,kde=True,bins=30)
```



Checking the Outliers for Original and Balanced Dataset

In [104]:
```
for i in num_col:
    plt.figure(figsize=(7,4))
    plt.subplot(121)
    sns.boxplot(data=data,x=i,color='r')


    plt.subplot(122)
    sns.boxplot(data=data_bal,x=i,color='r')
```



Test train Split

In [105]:
```
from sklearn.model_selection import train_test_split
X_train1,X_test1,y_train1,y_test1=train_test_split(X_bal,y_bal,test_size=0.30,random_state=16)
```

In [106]:
```
X_train1
```

Out[106]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	region
135	29	8	2012	35	48	18	0.000000	90.100000	54.200000	220.400000	12.500000	67.400000	30.200000	0
46	30	9	2012	24	64	15	0.200000	67.300000	3.800000	16.500000	1.200000	4.800000	0.500000	1
7	3	6	2012	29	80	14	2.000000	48.700000	2.200000	7.600000	0.300000	2.600000	0.100000	1
290	2	8	2012	28	67	18	0.051431	75.076719	3.076335	16.039171	1.966937	4.124904	0.792653	0
199	10	7	2012	33	69	13	0.700000	66.600000	6.000000	9.300000	1.100000	5.800000	0.500000	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
321	26	6	2012	31	59	16	0.086622	83.029549	14.203559	87.458423	4.049279	20.030316	6.318288	0
69	30	7	2012	36	56	16	0.000000	88.900000	23.800000	57.100000	8.200000	23.800000	13.200000	1
121	6	7	2012	32	63	14	0.000000	87.000000	10.900000	37.000000	5.600000	12.500000	6.800000	0
238	21	7	2012	34	60	17	0.525411	80.792008	16.251054	54.615057	4.786356	19.871763	6.887769	0
169	28	6	2012	32	55	14	0.000000	89.100000	25.500000	88.500000	7.600000	29.700000	13.900000	0

294 rows × 14 columns

In [107]:
```
X_test1
```

Out[107]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	region
64	1	6	2012	32	71	12	0.700000	57.100000	2.500000	8.200000	0.600000	2.800000	0.200000	1
54	17	7	2012	29	70	14	0.000000	82.800000	9.400000	34.100000	3.200000	11.100000	3.600000	0
26	14	7	2012	37	37	18	0.200000	88.900000	12.900000	14.690000	12.500000	10.400000	0.400000	1
355	20	7	2012	31	69	16	0.000000	87.658523	41.020478	94.070302	6.944815	41.794379	15.358876	0
167	21	7	2012	33	70	17	0.000000	85.400000	18.500000	71.500000	5.200000	22.400000	8.800000	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
212	10	9	2012	33	73	12	1.800000	59.900000	2.200000	8.900000	0.700000	2.700000	0.300000	0
140	22	6	2012	31	67	17	0.100000	79.100000	7.000000	39.500000	2.400000	9.700000	2.300000	0
264	4	8	2012	29	73	16	0.000000	82.976676	6.250317	30.125072	3.916772	8.147282	3.756388	0
12	1	7	2012	28	58	18	2.200000	63.700000	3.200000	8.500000	1.200000	3.300000	0.500000	1
375	6	7	2012	35	41	14	0.299257	84.727129	15.526386	45.174326	4.319325	16.727872	6.331588	1

126 rows × 14 columns

In [108]:
```
y_train1
```

Out[108]:
```
135    1
46     1
7      1
290    0
199    1
      ..
321    0
69     1
121    1
238    0
169    1
Name: Classes, Length: 294, dtype: int32
```

In [109]:
```
y_test1
```

Out[109]:
```
64     1
54     1
26     1
355    0
167    1
      ..
212    0
140    1
264    0
12     1
375    0
Name: Classes, Length: 126, dtype: int32
```

Logistic Regression Model

In [111]:
```
from sklearn.linear_model import LogisticRegression
classifier_bal=LogisticRegression()
classifier_bal
```

Out[111]:
```
LogisticRegression
LogisticRegression()
```

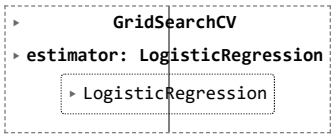
In [112]:
```
from sklearn.model_selection import GridSearchCV
parameter_bal={'penalty':['l1','l2','elasticnet'],'C':[1,2,3,4,5,6,10,20,30,40,50],'max_iter':[100,200,300]}
```

In [113]:
```
classifier_regressor_bal=GridSearchCV(classifier,param_grid=parameter,scoring='accuracy',cv=5)
```

Standarizing or Feature Scaling

In [114]:
```
classifier_regressor_bal.fit(X_train1,y_train1)
```

Out[114]:



In [115]:
```
print(classifier_regressor_bal.best_params_)
```

```
{'C': 40, 'max_iter': 300, 'penalty': 'l2'}
```

In [116]:
```
print(classifier_regressor_bal.best_score_)
```

```
0.683927527761543
```

Prediction

In [117]:
```
y_bal_pred = classifier_regressor_bal.predict(X_test1)
```

In [118]:
```
y_bal_pred
```

Out[118]:
```
array([1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1,
       0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0])
```

Acurracy

In [119]:
```
from sklearn.metrics import accuracy_score,classification_report
bal_score=accuracy_score(y_bal_pred,y_test1)
print(bal_score)
```

```
0.7142857142857143
```

Classification Report

In [120]:
```
print(classification_report(y_bal_pred,y_test1))
```

```
              precision    recall  f1-score   support

           0       0.79      0.71      0.75        75
           1       0.63      0.73      0.67        51

    accuracy                           0.71       126
   macro avg       0.71      0.72      0.71       126
weighted avg       0.72      0.71      0.72       126
```

Performance Metrics

Confusion Metrics

In [121]:
```
conf_mat_bal=confusion_matrix(y_bal_pred,y_test1)
```

In [122]:
```
conf_mat_bal
```

Out[122]:
```
array([[53, 22],
       [14, 37]], dtype=int64)
```

In [123]:
```
true_positive = conf_mat_bal[0][0]
false_positive = conf_mat_bal[0][1]
false_negative = conf_mat_bal[1][0]
true_negative = conf_mat_bal[1][1]
```

Precision

In [124]:
```
bal_Precision = true_positive/(true_positive+false_positive)
bal_Precision
```

Out[124]: 0.7066666666666667

Recall

In [125]:
```
bal_recall = true_positive/(true_positive+false_negative)
bal_recall
```

Out[125]: 0.7910447761194029

F1 score

In [126]:
```
F1_Score_bal = 2*(bal_recall * bal_Precision) / (bal_recall + bal_Precision)
F1_Score_bal
```

Out[126]: 0.7464788732394366

Conclusion

Performance of Logistic Model on Original Dataset

In [127]:
```
print(classification_report(y_pred,y_test))
```

```
              precision    recall  f1-score   support

         0.0       0.96      0.92      0.94        25
         1.0       0.95      0.97      0.96        36

    accuracy                           0.95        61
   macro avg       0.95      0.95      0.95        61
weighted avg       0.95      0.95      0.95        61
```

Observation

It seems that model is good when we predict from original dataset

Performance of Logistic Model on Balanced Dataset which are created from imbalanced dataset

In [128]:
```
print(classification_report(y_bal_pred,y_test1))
```

```
              precision    recall  f1-score   support

           0       0.79      0.71      0.75        75
           1       0.63      0.73      0.67        51

    accuracy                           0.71       126
   macro avg       0.71      0.72      0.71       126
weighted avg       0.72      0.71      0.72       126
```

It seems that model is very bad when we try to predict from balanced(created from an imbalanced dataset)

In [ ]:
```

```