Used Forest Fire Data Exploration

EDA steps

- Understanding the Problem Statement
- Data Collection
- Exploratory data analysis
- Data Cleaning
- Data Pre-Processing

1) Problem statement.

- This dataset inform about the firest fire happen cuses.
- Elpore the data and find out the causes of forest fire

2) Data Collection.

- The Dataset is collected from scrapping from archive.ics.uci.edu
- The dataset includes 244 instances that regroup a data of two regions of Algeria,namely the Bejaia region located in the northeast of Algeria and the Sidi Bel-abbes region located in the northwest of Algeria.
- 122 instances for each region.
- The period from June 2012 to September 2012.
- The dataset includes 11 attribues and 1 output attribue (class)

Abstract: The dataset includes 244 instances that regroup a data of two regions of Algeria.

Data Set Characteristics:	Multivariate	**Number of Instances:**	244	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	12	**Date Donated**	2019-10-22
Associated Tasks:	Classification, Regression	**Missing Values?**	N/A	**Number of Web Hits:**	56292

Import Data and Required Packages

In [1]:

```
# data read model
import pandas as pd
import numpy as np
import statistics as st

# graph module
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import cufflinks as cf
import warnings
import plotly.graph_objects as go
warnings.filterwarnings("ignore")
%matplotlib inline
from plotly.offline import download_plotlyjs,init_notebook_mode,plot,iplot
init_notebook_mode(connected=True)
cf.go_offline()

# for Q-Q plots
import scipy.stats as stats


from six.moves import urllib

# system module
import os

pd.set_option('display.max_rows', 500)
```

Import CSV using Pandas Dataframe

In [2]:

```
df =  pd.read_csv("E:\OneDrive - student.amity.edu\office\python &
R\DataScience_ineuron\Ineuron tutorial resources\EDA\EDA
manual\Algerian_forest_fires_dataset_UPDATE.csv")
```

Top 5 Records

In [3]:

```
df.head()
```

Out[3]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI
0	1	6	2012	29	57	18	0.0	65.7	3.4	7.6	1
1	2	6	2012	29	61	13	1.3	64.4	4.1	7.6	1
2	3	6	2012	26	82	22	13.1	47.1	2.5	7.1	0
3	4	6	2012	25	89	13	2.5	28.6	1.3	6.9	0
4	5	6	2012	27	77	16	0.0	64.8	3.0	14.2	1

In [4]:

```
#remove spaces from columna
df.columns = df.columns.str.replace(' ', '')
```

Shape of dataset

In [5]:

```
df.shape
print('data provide %s rows  and %s columns'%(df.shape[0],df.shape[1]))
```

```
data provide 244 rows  and 15 columns
```

Statistical Summary of Dataset

In [6]:

```
df.describe(include='all')
```

Out[6]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DI
count	244.000000	244.000000	244.0	244.000000	244.000000	244.000000	244.000000	244.000000	244.0
unique	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
freq	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
mean	15.754098	7.500000	2012.0	32.172131	61.938525	15.504098	0.760656	77.887705	14.6
std	8.825059	1.112961	0.0	3.633843	14.884200	2.810178	1.999406	14.337571	12.3
min	1.000000	6.000000	2012.0	22.000000	21.000000	6.000000	0.000000	28.600000	0.7

DATA Information:

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

Datatypes of Dataset

In [7]:

```
df.info()
```

```
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   day          244 non-null    int64  
 1   month        244 non-null    int64  
 2   year         244 non-null    int64  
 3   Temperature  244 non-null    int64  
 4   RH           244 non-null    int64  
 5   Ws           244 non-null    int64  
 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(5), int64(6), object(4)
memory usage: 28.7+ KB
```

Find unique value in dataset

In [8]:

```
df.nunique()
```

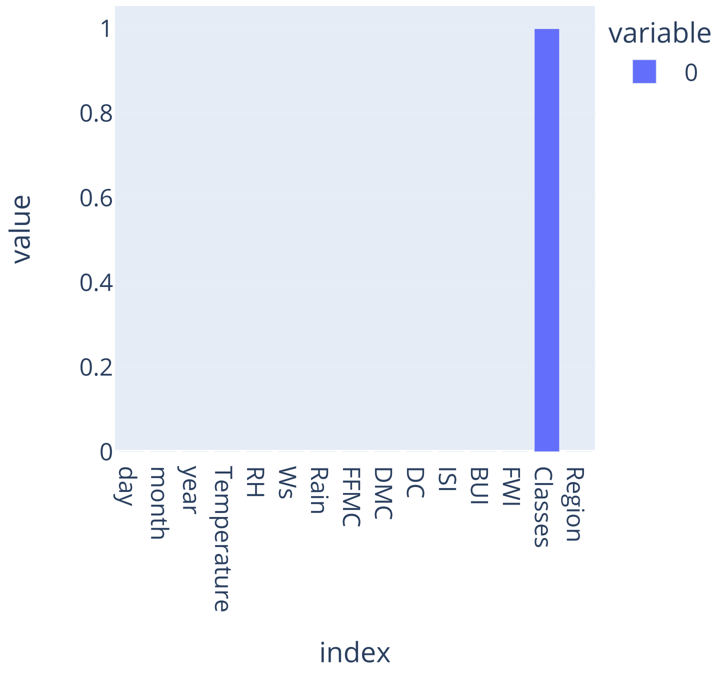
Out[8]:

```
day             31
month            4
year             1
Temperature     19
RH              62
Ws              18
Rain            39
FFMC           173
DMC            166
DC             198
ISI            106
BUI            174
FWI            126
Classes          8
Region           2
dtype: int64
```

Find null value in dataset

In [9]:

```
# df.isnull().mean().plot.bar(figsize=(6,4))
px.bar(df.isnull().sum(),width=400,height=400)
```



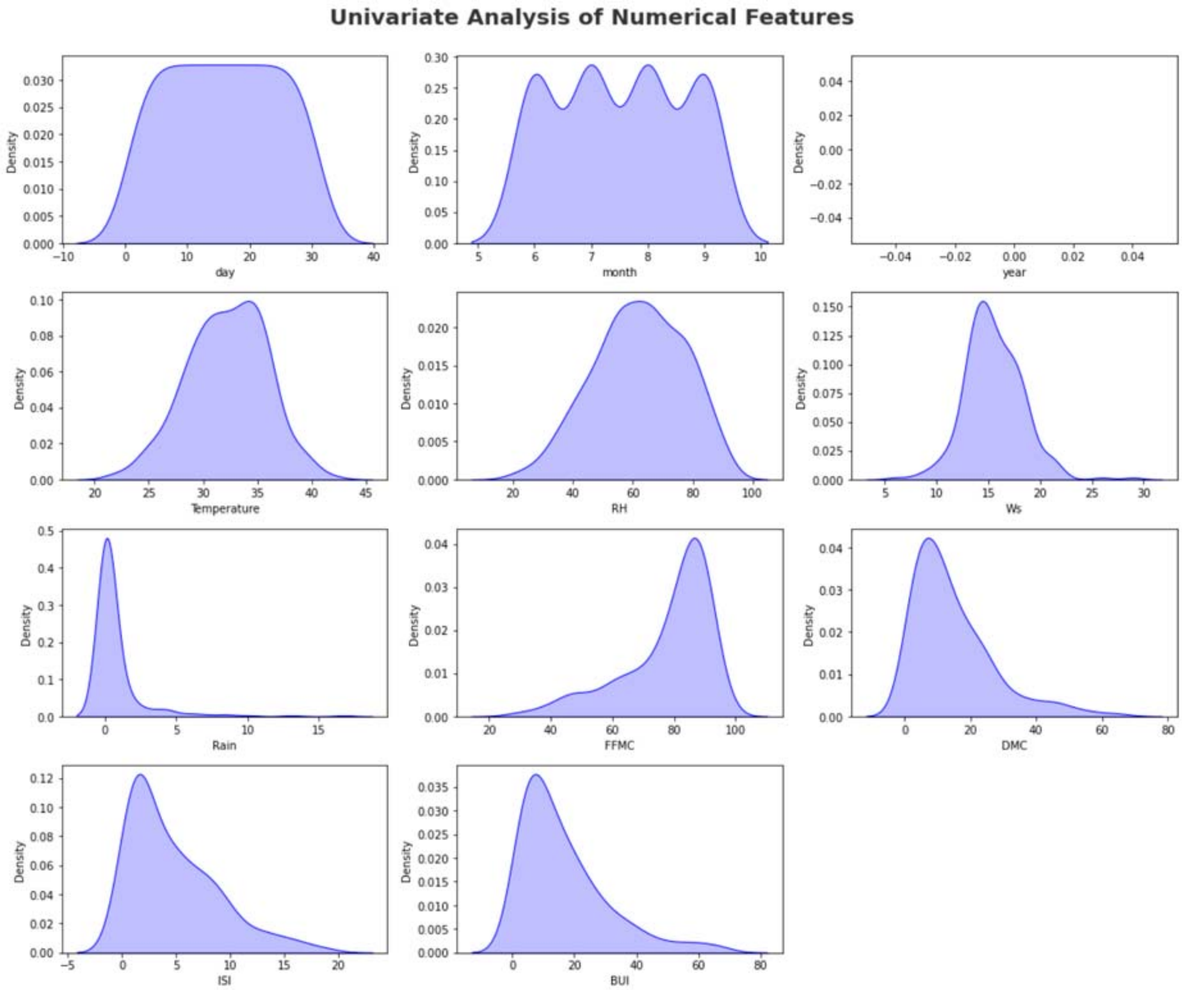
In [10]:

```
#findout the  numerical feature and categorical feature
numerical_feature=[feature for feature in df.columns if df[feature].dtype != 'O']
categorical_feature=[feature for feature in df.columns if df[feature].dtype == 'O']

print("There are %s numerical feature : %s"%(len(numerical_feature),numerical_feature))
print("There are %s categorical feature : %s"%
(len(categorical_feature),categorical_feature))
```

```
There are 11 numerical feature : ['day', 'month', 'year', 'Temperature', 'R
H', 'Ws', 'Rain', 'FFMC', 'DMC', 'ISI', 'BUI']
There are 4 categorical feature : ['DC', 'FWI', 'Classes', 'Region']
```

In [11]:

```
plt.figure(figsize=(15, 15))
plt.suptitle('Univariate Analysis of Numerical Features', fontsize=20,
fontweight='bold', alpha=0.8, y=1.)

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



Report

- rain,DMC,ISI,BUI are right skewed
- FFMC is left skewed

multivariate analysis

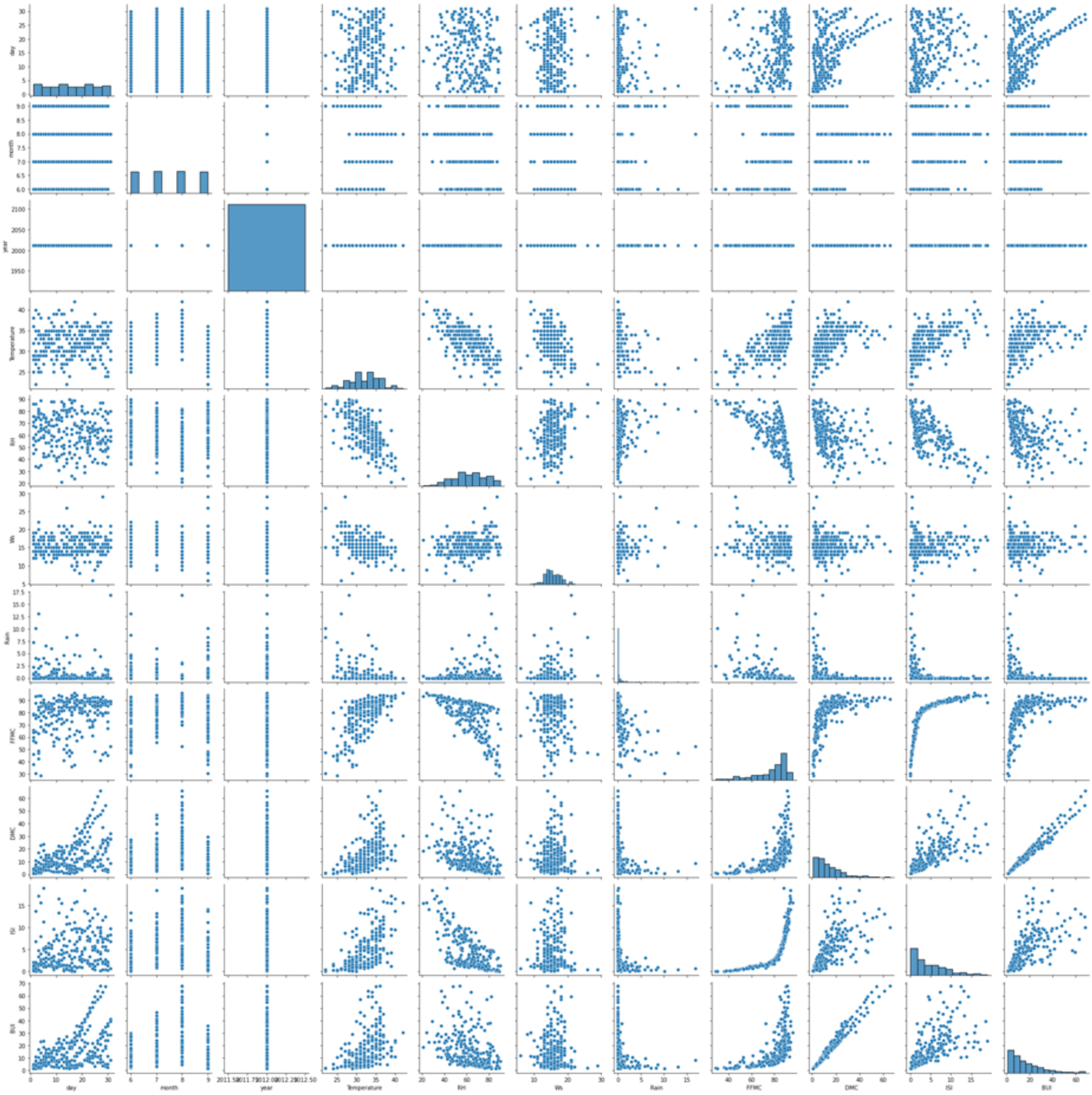
In [12]:

```
plt.figure(figsize=(15, 15))
plt.suptitle('Univariate Analysis of Numerical Features', fontsize=20,fontweight='bold',
alpha=0.8, y=1.)
sns.pairplot(df)
```

Out[12]:

```
<seaborn.axisgrid.PairGrid at 0x29f3601e100>
```

```
<Figure size 1080x1080 with 0 Axes>
```



Categorical feature

remove spaces from categorical value

In [13]:

```
df['DC'] = df['DC'].str.replace(' ','')
```

In [14]:

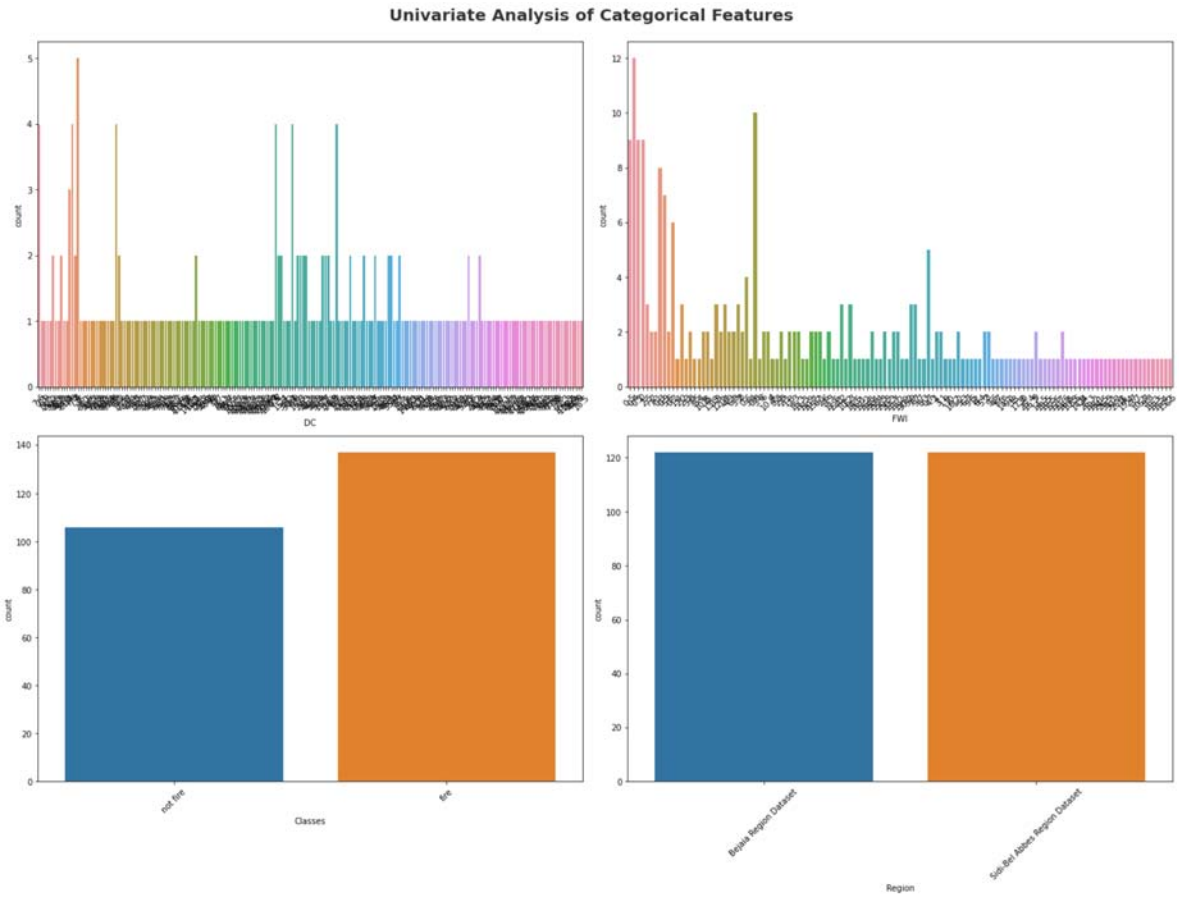
```
df['FWI'] = df['FWI'].str.replace(' ','')
```

In [15]:

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

In [16]:

```
# categorical columns
plt.figure(figsize=(20, 15))
plt.suptitle('Univariate Analysis of Categorical Features', fontsize=20,
fontweight='bold', alpha=0.8, y=1.)
cat1 = [ 'DC', 'FWI', 'Classes', 'Region']
for i in range(0, len(cat1)):
    plt.subplot(2, 2, i+1)
    sns.countplot(x=df[cat1[i]])
    plt.xlabel(cat1[i])
    plt.xticks(rotation=45)
    plt.tight_layout()
```



Report

- DC,FWI is a numerical data but in data it shows object

- Classes and Region are object data

In [17]:

```
def diagnostic_plots(df, variable):
    # function to plot a histogram and a Q-Q plot
    # side by side, for a certain variable

    plt.figure(figsize=(15,6))
    plt.subplot(1, 2, 1)
    df[variable].hist(bins=30)

    plt.subplot(1, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)

    plt.show()
```

In [18]:

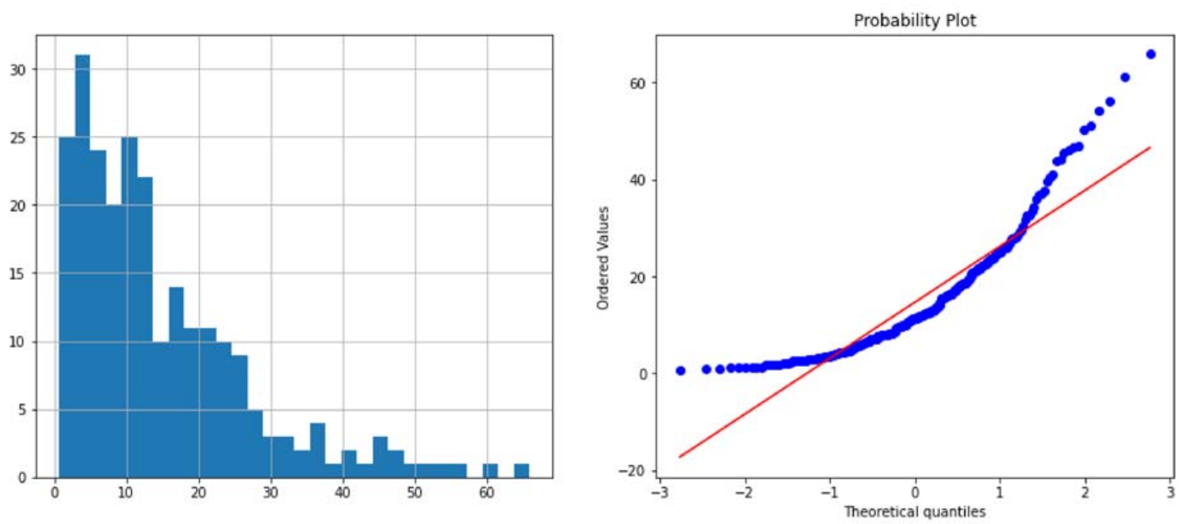
```
df.columns
```

Out[18]:

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

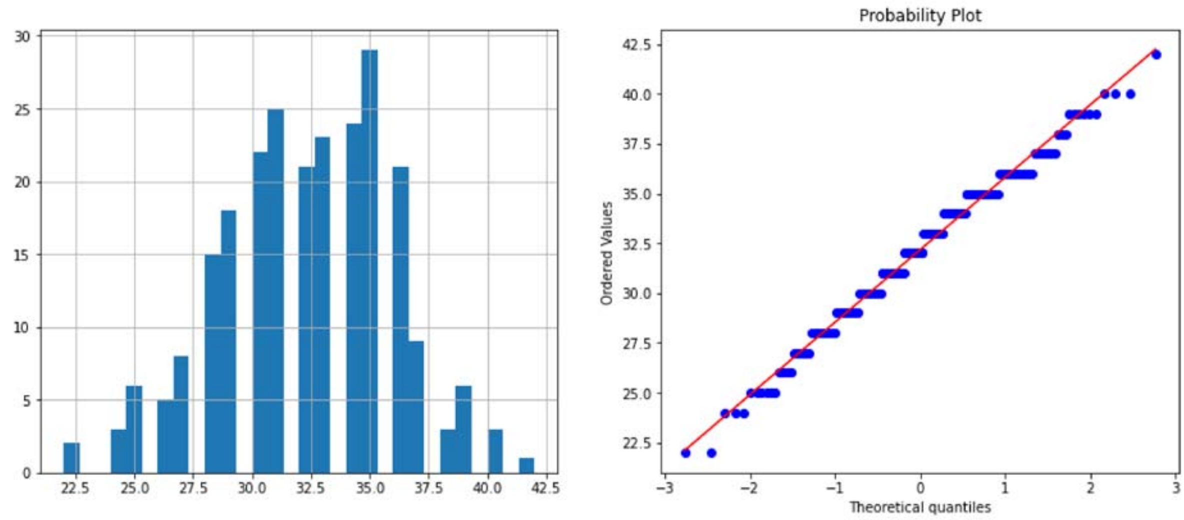
In [19]:

```
diagnostic_plots(df, 'DMC')
```



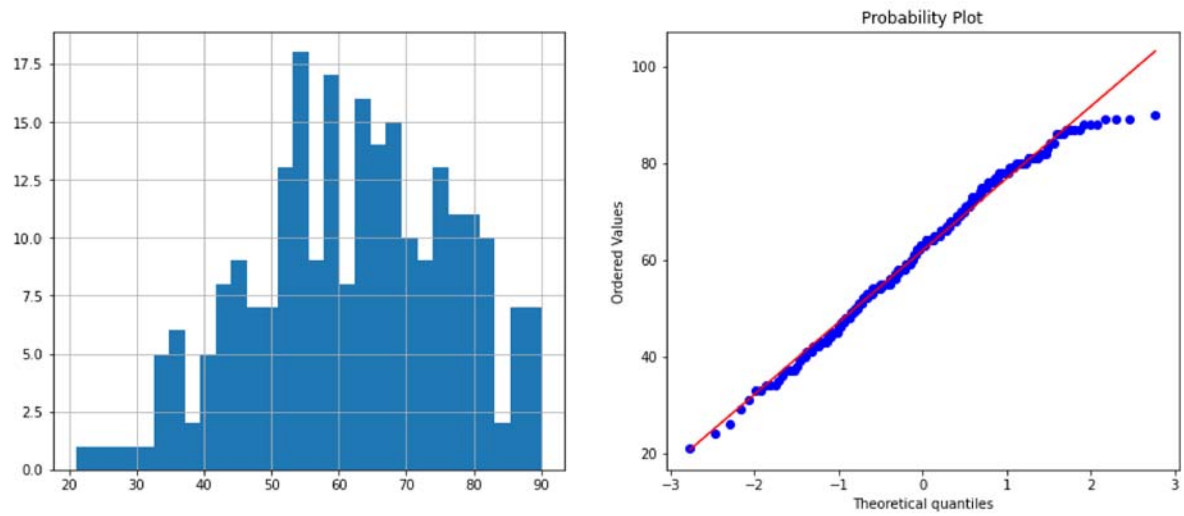
In [20]:

```
diagnostic_plots(df, 'Temperature')
```



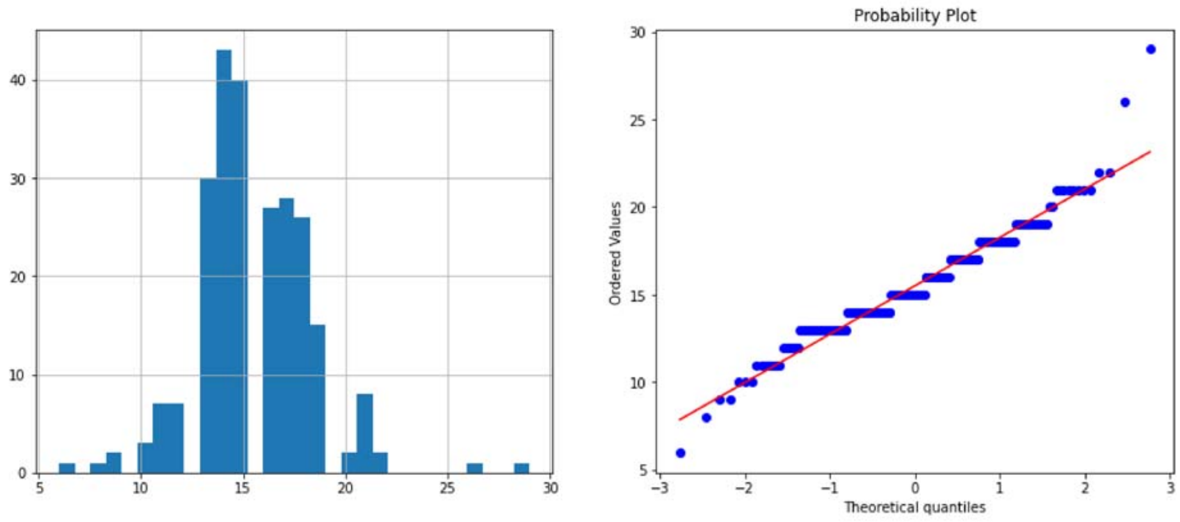
In [21]:

```
diagnostic_plots(df, 'RH')
```



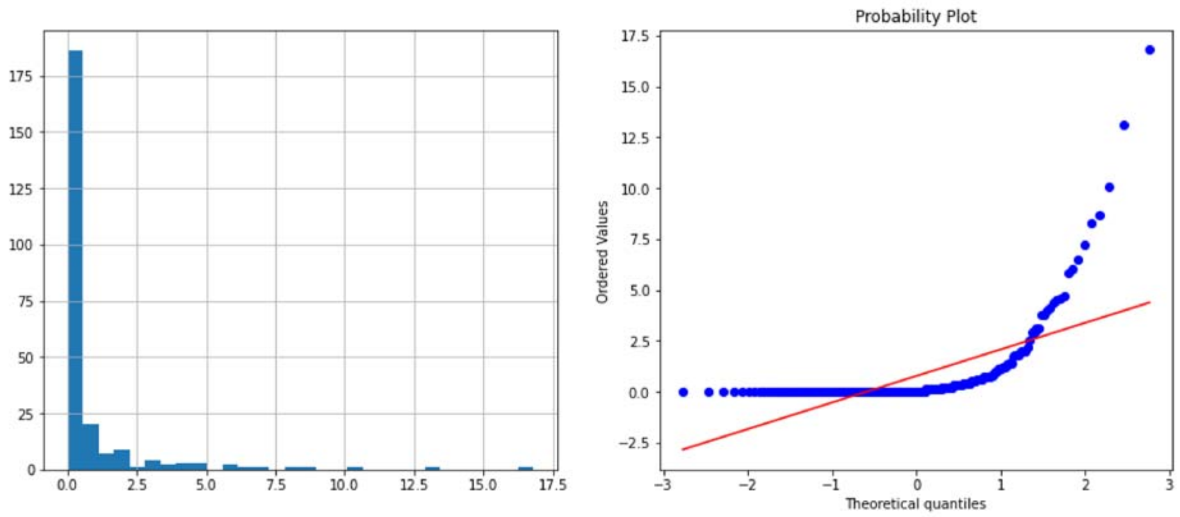
In [22]:

```
diagnostic_plots(df, 'Ws')
```



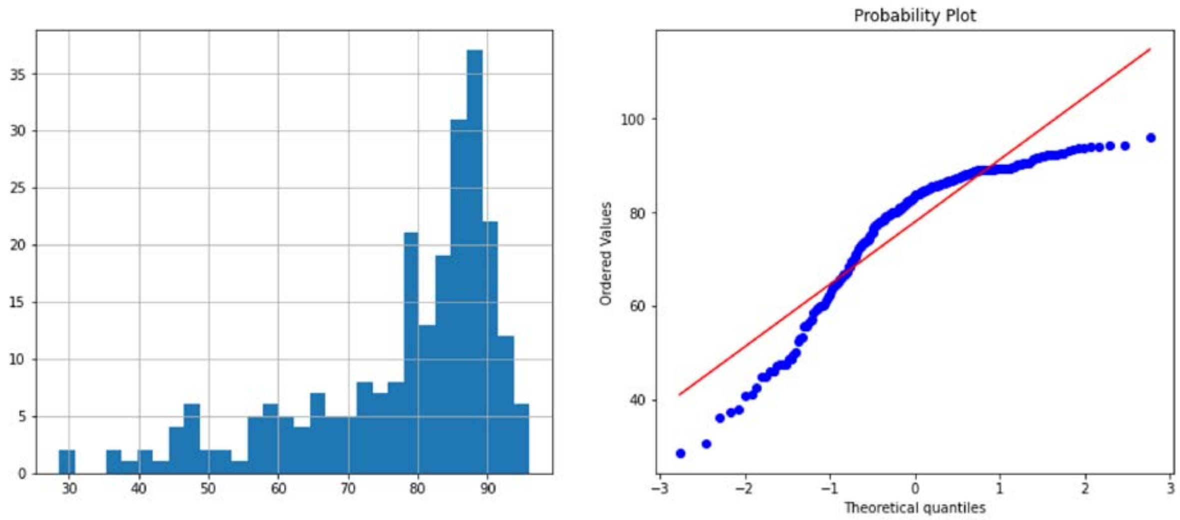
In [23]:

```
diagnostic_plots(df, 'Rain')
```



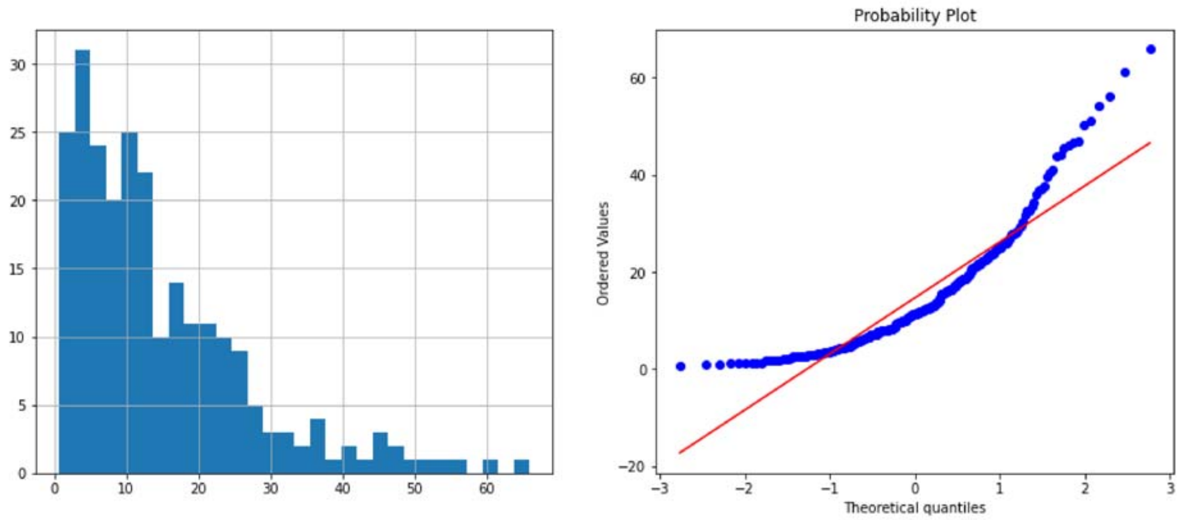
In [24]:

```
diagnostic_plots(df, 'FFMC')
```



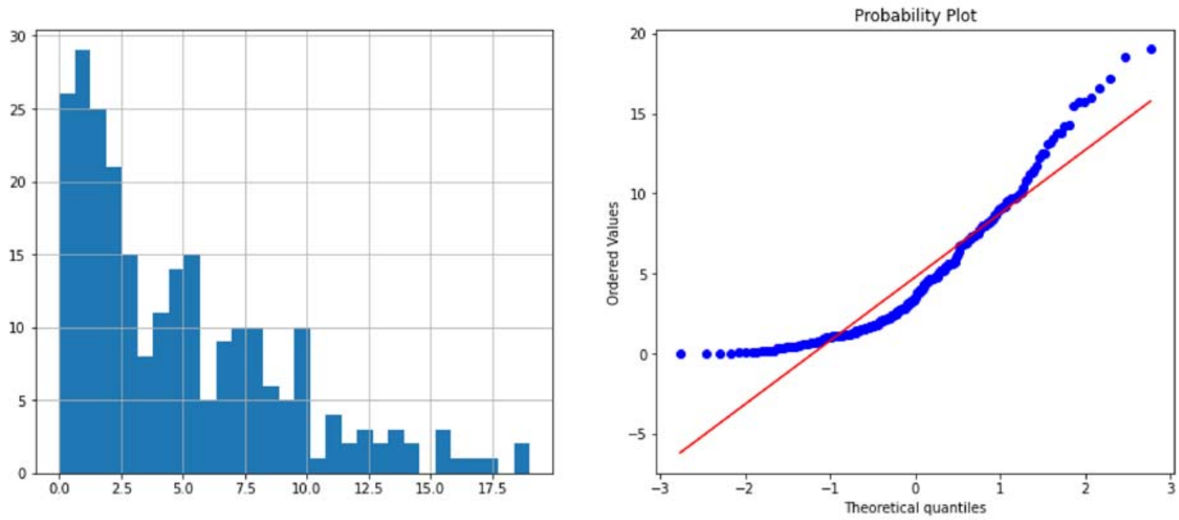
In [25]:

```
diagnostic_plots(df, 'DMC')
```



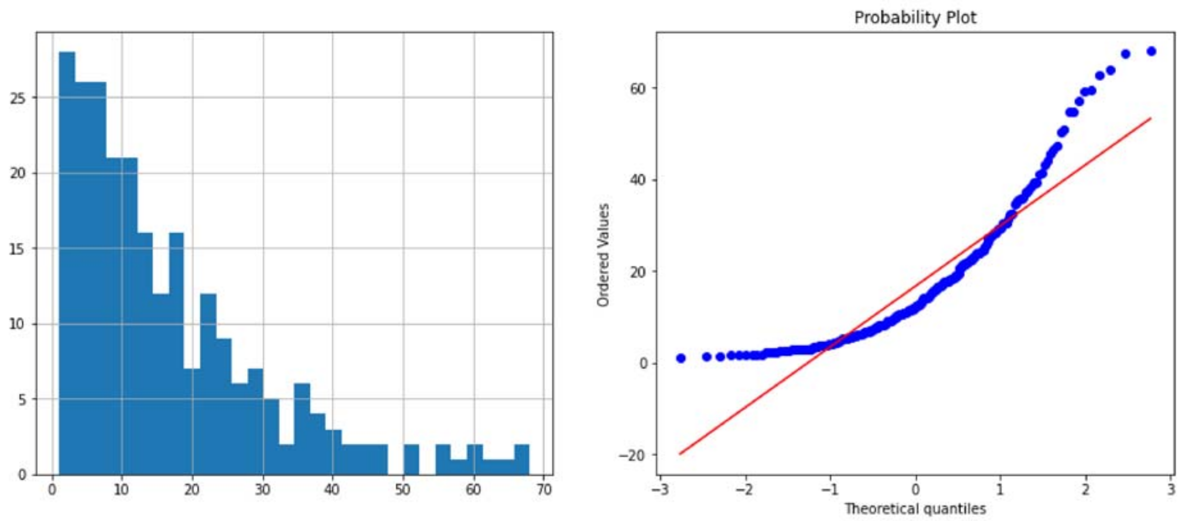
In [26]:

```
diagnostic_plots(df, 'ISI')
```



In [27]:

```
diagnostic_plots(df, 'BUI')
```



In [ ]:

```

```

In [ ]:

```

```

5) Data Cleaning

Convert object to numerical column because it is a index value which wil come in float type

DC string to float

In [28]:

```
df['DC'] = [float(x) for x in df['DC']]
```

FWI string to float

In [29]:

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

Out[29]:

```
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)
```

as we see that in the FWI column all value are float or numeric but value is string so we have to replace the value with a float or integer value

In [30]:

```
df[df['FWI'].str.isalnum()]
```

Out[30]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC
3	4	6	2012	25	89	13	2.5	28.6	1.3	6.90
15	16	6	2012	29	89	13	0.7	36.1	1.7	7.60
16	17	6	2012	30	89	16	0.6	37.3	1.1	7.80
26	27	6	2012	34	53	18	0.0	89.0	21.6	80.30
37	8	7	2012	33	68	19	0.0	85.6	12.5	49.80
47	18	7	2012	31	68	14	0.0	85.4	12.1	43.10
49	20	7	2012	33	65	15	0.1	81.4	12.3	62.10
67	7	8	2012	32	69	16	0.0	86.5	15.5	48.60
93	2	9	2012	22	86	15	10.1	30.5	0.7	7.00
94	3	9	2012	25	78	15	3.8	42.6	1.2	7.50
104	13	9	2012	25	86	21	4.6	40.9	1.3	7.50
106	15	9	2012	24	82	15	0.4	44.9	0.9	7.30
125	4	6	2012	30	64	14	0.0	79.4	5.2	15.40

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC
128	7	6	2012	35	44	17	0.2	85.6	9.9	28.90
131	10	6	2012	30	41	15	0.0	89.4	13.3	22.50
148	27	6	2012	36	55	15	0.0	89.1	20.9	43.30
156	5	7	2012	34	45	18	0.0	90.5	18.7	46.40
159	8	7	2012	35	47	18	6.0	80.8	9.8	9.70
165	14	7	2012	37	37	18	0.2	88.9	12.9	14.69
170	19	7	2012	34	58	16	0.0	88.1	27.8	61.10
172	21	7	2012	36	29	18	0.0	93.9	39.6	80.60
177	26	7	2012	35	58	10	0.2	78.3	10.8	19.70
179	28	7	2012	33	57	16	0.0	87.5	15.7	37.60
199	17	8	2012	42	24	9	0.0	96.0	30.3	76.40
202	20	8	2012	36	81	15	0.0	83.7	34.4	107.00

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC
212	30	8	2012	34	49	15	0.0	89.2	24.8	159.10
214	1	9	2012	29	86	16	0.0	37.9	0.9	8.20
238	25	9	2012	28	70	15	0.0	79.9	13.8	36.10
240	27	9	2012	28	87	15	4.4	41.1	6.5	8.00

In [31]:

```
df[df['FWI'].str.isalpha()]
```

Out[31]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI
165	14	7	2012	37	37	18	0.2	88.9	12.9	14.69	1

In [32]:

```
#fins out the median value for replace the string value
# res = []
# for r in df['FWI']:
#     try:
#         if float(r) or int(r):
#             res.append(float(r))
#     except ValueError:
#         print(r)
# median  = st.median(res)
# print(median)
```

In [33]:

```
# df['FWI'] = df['FWI'].replace('fire',median)
```

In [34]:

```
df.drop(df['FWI'].index[165],inplace=True)
```

In [35]:

```
df['FWI'] = [float(x) for x in df['FWI']]
```

In [36]:

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

Out[36]:

```
0    0.5
1    0.4
2    0.1
3    0.0
4    0.5
Name: FWI, dtype: float64
```

lable encoding

In [37]:

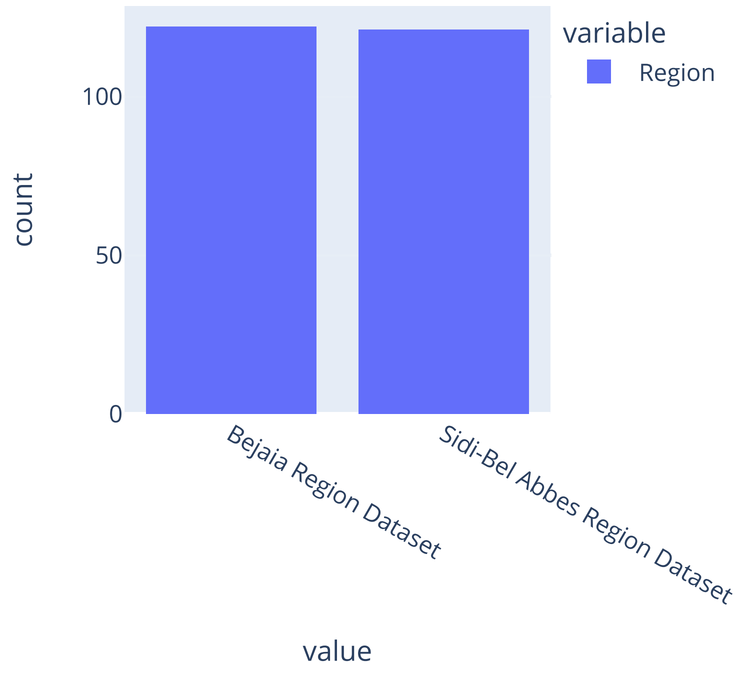
```
df['Region'].unique()
```

Out[37]:

```
array(['Bejaia Region Dataset', 'Sidi-Bel Abbes Region Dataset'],
      dtype=object)
```

In [38]:

```
px.histogram(df['Region'],width=400, height=400)
```



In [39]:

```
df['Region'] = df['Region'].map({'Bejaia Region Dataset':0,'Sidi-Bel Abbes Region Dataset':1})
```

In [40]:

```
df['Region'].value_counts().reset_index()
```

Out[40]:

	index	Region
0	0	122
1	1	121

In [41]:

```
df.head()
```

Out[41]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI
0	1	6	2012	29	57	18	0.0	65.7	3.4	7.6	1
1	2	6	2012	29	61	13	1.3	64.4	4.1	7.6	1
2	3	6	2012	26	82	22	13.1	47.1	2.5	7.1	0
3	4	6	2012	25	89	13	2.5	28.6	1.3	6.9	0
4	5	6	2012	27	77	16	0.0	64.8	3.0	14.2	1

In [42]:

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

Out[42]:

```
array(['not fire', 'fire'], dtype=object)
```

In [43]:

```
df['Classes'].isna().sum()
```

Out[43]:

```
0
```

In [44]:

```
mode = st.mode(df['Classes'])
mode
```

Out[44]:

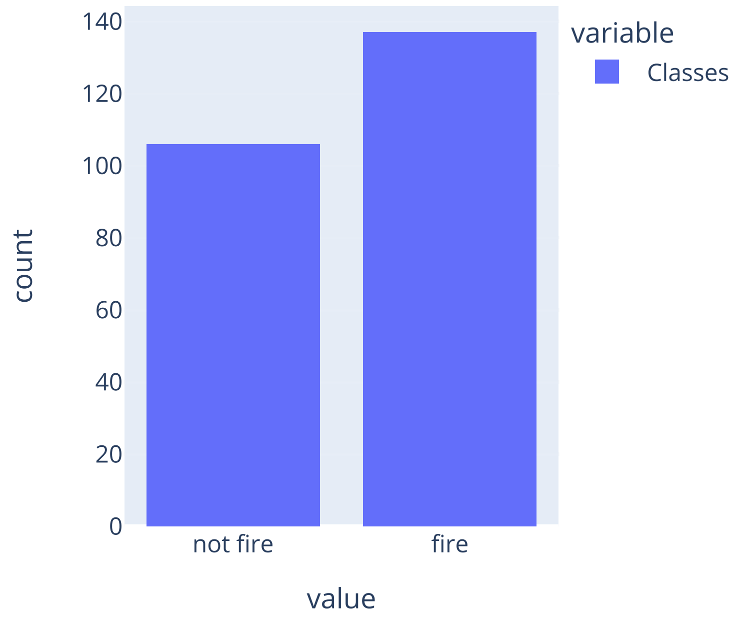
```
'fire'
```

In [45]:

```
df['Classes'].fillna(mode,inplace=True)
```

In [46]:

```
px.histogram(df['Classes'],width=400, height=400)
```



In [47]:

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

In [48]:

```
df.head()
```

Out[48]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI
0	1	6	2012	29	57	18	0.0	65.7	3.4	7.6	1
1	2	6	2012	29	61	13	1.3	64.4	4.1	7.6	1
2	3	6	2012	26	82	22	13.1	47.1	2.5	7.1	0
3	4	6	2012	25	89	13	2.5	28.6	1.3	6.9	0
4	5	6	2012	27	77	16	0.0	64.8	3.0	14.2	1

In [49]:

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

Out[49]:

	index	Classes
0	1	137
1	0	106

In [50]:

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

In [ ]:

```

```

check multicollinerity in numerical feature

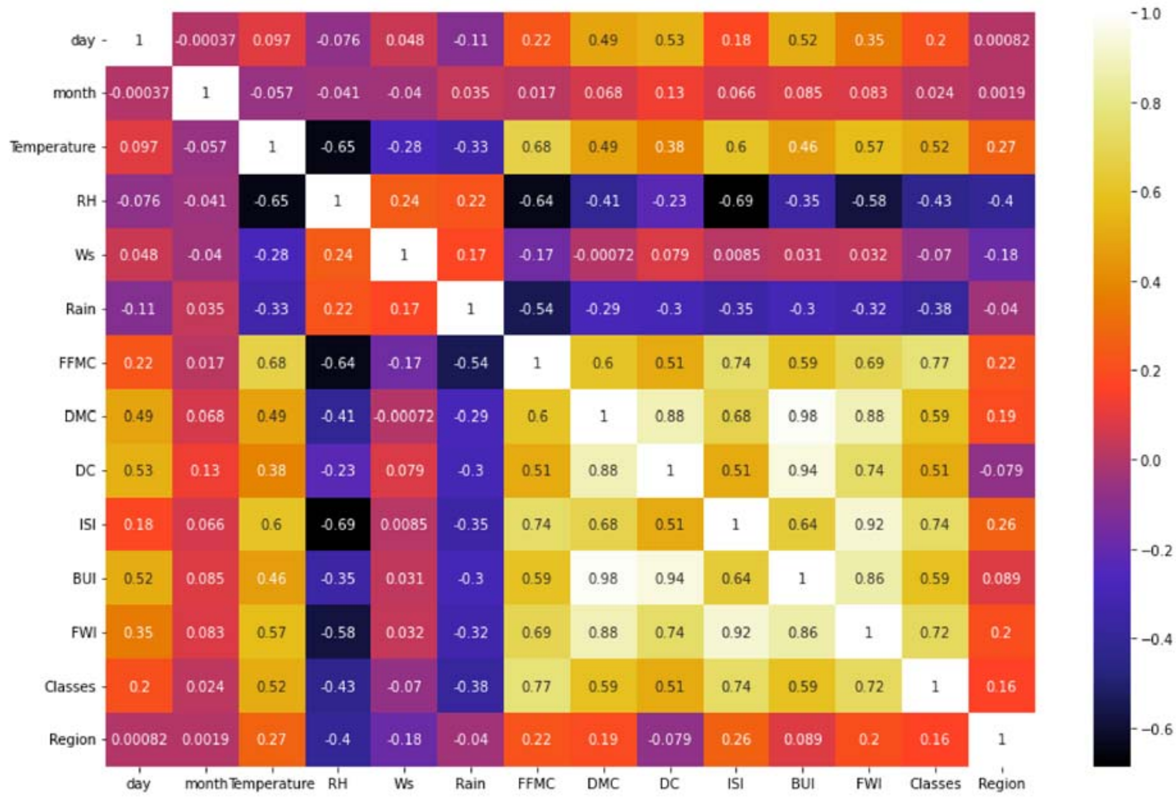
In [51]:

```
df[(list(df.columns)[1:])].corr()
```

	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI
month	1.000000	-0.056781	-0.041252	-0.039880	0.034822	0.017030	0.067943	0.126511	0.0656
Temperature	-0.056781	1.000000	-0.651400	-0.284510	-0.326492	0.676568	0.485687	0.376284	0.6038
RH	-0.041252	-0.651400	1.000000	0.244048	0.222356	-0.644873	-0.408519	-0.226941	-0.6866
Ws	-0.039880	-0.284510	0.244048	1.000000	0.171506	-0.166548	-0.000721	0.079135	0.0085
Rain	0.034822	-0.326492	0.222356	0.171506	1.000000	-0.543906	-0.288773	-0.298023	-0.3474
FFMC	0.017030	0.676568	-0.644873	-0.166548	-0.543906	1.000000	0.603608	0.507397	0.7400
DMC	0.067943	0.485687	-0.408519	-0.000721	-0.288773	0.603608	1.000000	0.875925	0.6804
DC	0.126511	0.376284	-0.226941	0.079135	-0.298023	0.507397	0.875925	1.000000	0.5086
ISI	0.065608	0.603871	-0.686667	0.008532	-0.347484	0.740007	0.680454	0.508643	1.0000

In [52]:

```
plt.figure(figsize = (15,10))
sns.heatmap(df.corr(), cmap="CMRmap", annot=True)
plt.show()
```



Report

- As we see year column has not use in dataset so we drop year column
- As we see there is negative correlation between target layer classes with RH,WS,Rain
- As we see there is less positive correlation between target layer classes with day,month

In [53]:

```
outlier = []
def detect_outlier(data):
    mean = np.mean(data)
    std = np.std(data)
    print(' min {} ,max {} ,mean {},std {}'.format(min(data),max(data),mean,std))
    print("higher value {} ".format(mean+3*std))
    print("lower value {} ".format(mean-3*std))
    threshold = 3
    for i in data:
        z_score = (i-mean)/std
        if np.abs(z_score)>threshold:
            outlier.append(i)
    return outlier
```

In [54]:

```
outlier_iqr = []
def deltect_outlier_quartile(data):
    dataset = sorted(data)
    mean = np.mean(data)
    std = np.std(data)
    q1,q3 = np.quantile(dataset,[.25,.90])
    iqr  =  q3-q1
    lowerf = q1-(1.5*iqr)
    higherf = q3+(1.5*iqr)
    print(' min {} ,max {} ,mean {},std {}'.format(min(data),max(data),mean,std))

    for i in dataset:
        if i<lowerf or i>higherf:
            outlier.append(i)
    return outlier
```

In [55]:

```
df.head()
```

Out[55]:

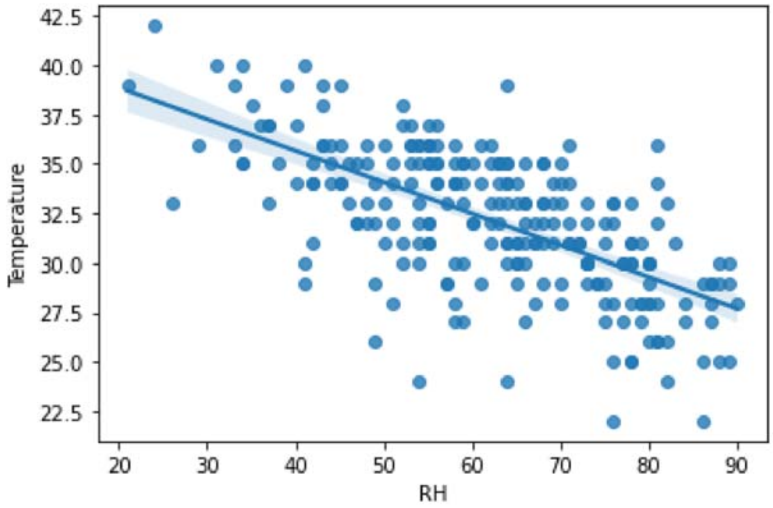
	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI
0	1	6	29	57	18	0.0	65.7	3.4	7.6	1.3	3.
1	2	6	29	61	13	1.3	64.4	4.1	7.6	1.0	3.
2	3	6	26	82	22	13.1	47.1	2.5	7.1	0.3	2.
3	4	6	25	89	13	2.5	28.6	1.3	6.9	0.0	1.
4	5	6	27	77	16	0.0	64.8	3.0	14.2	1.2	3.

In [56]:

```
sns.regplot(x="RH",y="Temperature",data=df)
```

Out[56]:

<AxesSubplot:xlabel='RH', ylabel='Temperature'>

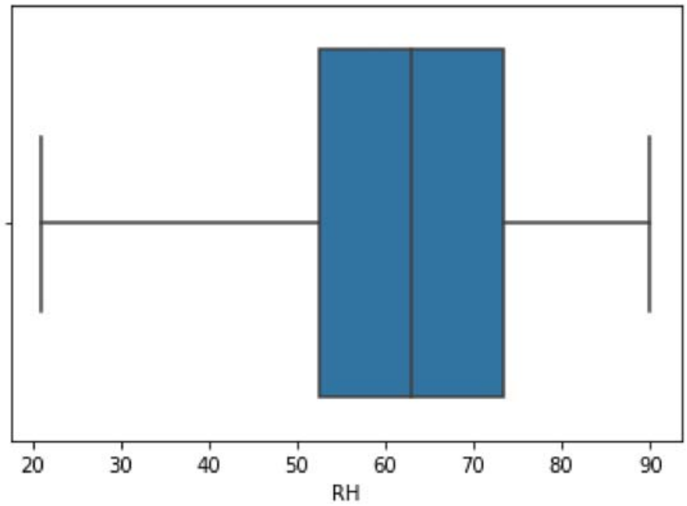


In [57]:

```
sns.boxplot(df['RH'])
```

Out[57]:

<AxesSubplot:xlabel='RH'>

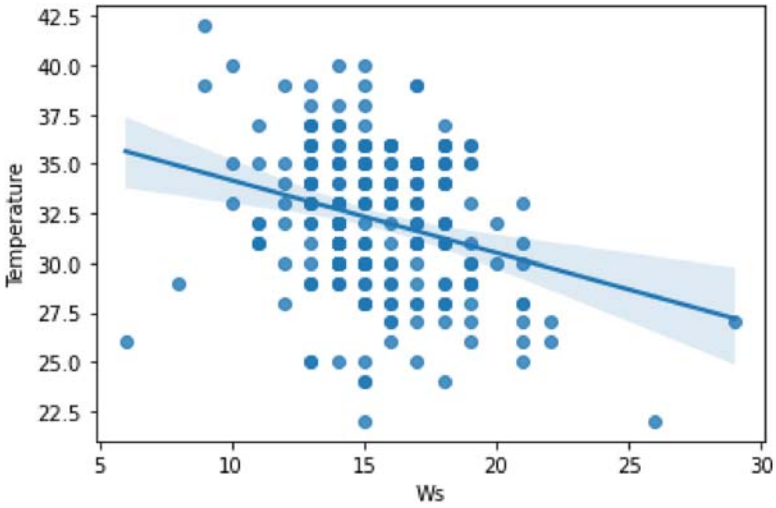


In [58]:

```python
sns.regplot(x="Ws",y="Temperature",data=df)
```

Out[58]:

<AxesSubplot:xlabel='Ws', ylabel='Temperature'>

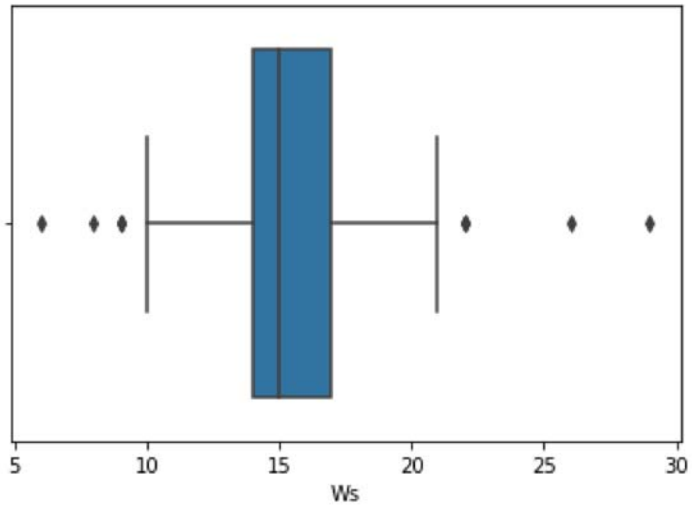


In [59]:

```python
sns.boxplot(df['Ws'])
```

Out[59]:

<AxesSubplot:xlabel='Ws'>

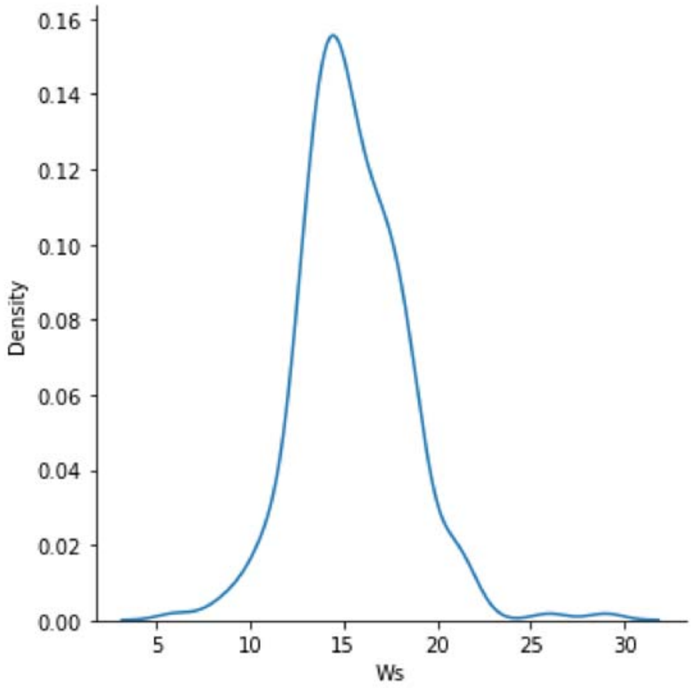


In [60]:

```
sns.displot(df['Ws'],kind='kde')
```

Out[60]:

```
<seaborn.axisgrid.FacetGrid at 0x29f3c715580>
```



In [61]:

```
# check skewness
df['Ws'].skew()
```

Out[61]:

```
0.5555858444767362
```

In [62]:

```
###find outlier
detect_outlier(df['Ws'])
```

```
 min 6 ,max 29 ,mean 15.493827160493828,std 2.8055946023984206
higher value 23.910610967689088
lower value 7.077043353298565
```

Out[62]:

```
[26, 6, 29]
```

In [63]:

```
mean_ws = np.mean(df['Ws'])
std_ws = np.std(df['Ws'])
higher_limit =  mean_ws+3*std_ws
lower_limit = mean_ws-3*std_ws

df[(df['Ws']<lower_limit) | (df['Ws']>higher_limit)]
```

Out[63]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI
105	14	9	22	76	26	8.3	47.4	1.1	7.0	0.4	1
237	24	9	26	49	6	2.0	61.3	11.9	28.1	0.6	11
241	28	9	27	87	29	0.5	45.9	3.5	7.9	0.4	3

In [64]:

```
df.shape
```

Out[64]:

(243, 14)

In [65]:

```
##triming outlier
df = df[(df['Ws']>lower_limit) & (df['Ws']<higher_limit)]
```

In [66]:

```
# caping
# df['Ws'] = np.where(df['Ws']>higher_limit
#           ,higher_limit
#           ,np.where(df['Ws']<lower_limit
#                     ,lower_limit,
#                     df['Ws']
#                    )
#             )
```

In [67]:

```
df.shape
```

Out[67]:

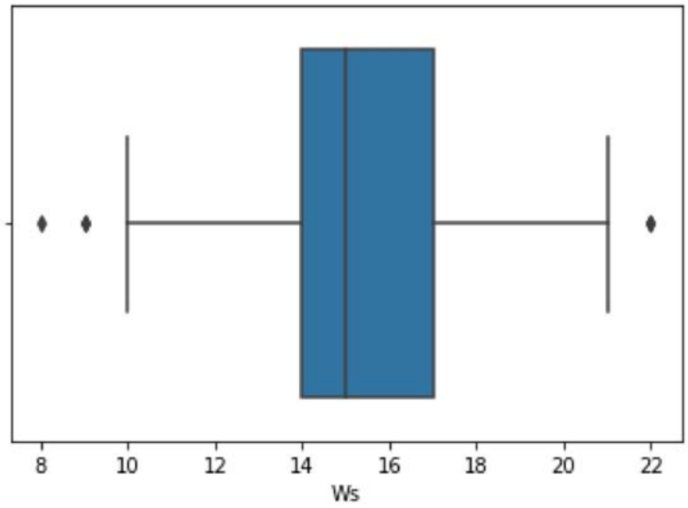
(240, 14)

In [68]:

```
sns.boxplot(df['Ws'])
```

Out[68]:

```
&lt;AxesSubplot:xlabel='Ws'&gt;
```

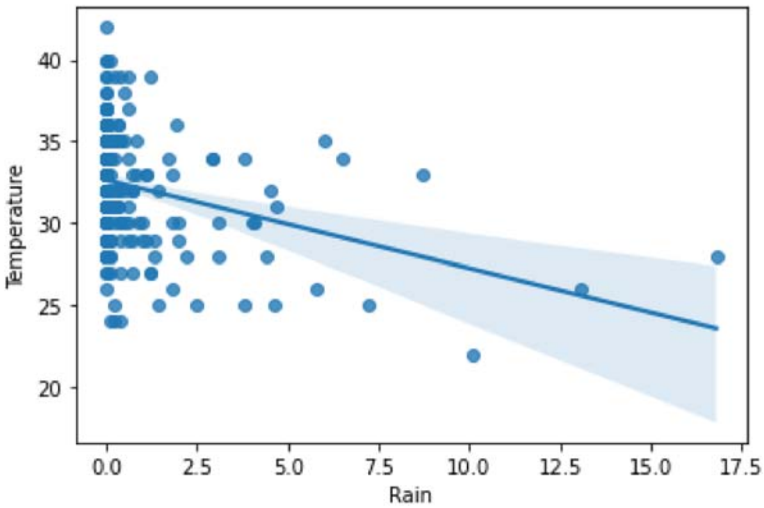


In [69]:

```
sns.regplot(x="Rain",y="Temperature",data=df)
```

Out[69]:

```
&lt;AxesSubplot:xlabel='Rain', ylabel='Temperature'&gt;
```



In [70]:

```
df['Rain'].describe()
```

Out[70]:

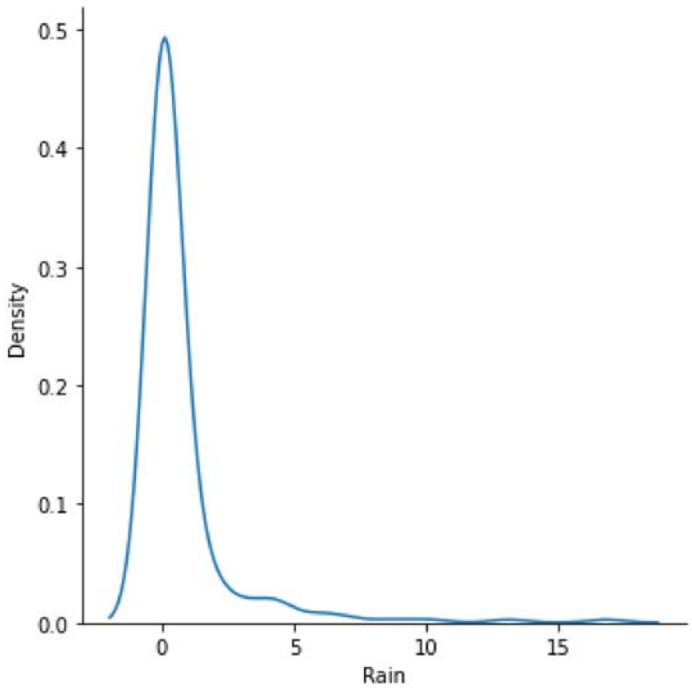
```
count    240.000000
mean       0.727500
std        1.953859
min        0.000000
25%        0.000000
50%        0.000000
75%        0.425000
max       16.800000
Name: Rain, dtype: float64
```

In [71]:

```
sns.displot(df['Rain'],kind='kde')
```

Out[71]:

<seaborn.axisgrid.FacetGrid at 0x29f3ee3d670>



In [72]:

```
# check skewness
df['Rain'].skew()
```

Out[72]:

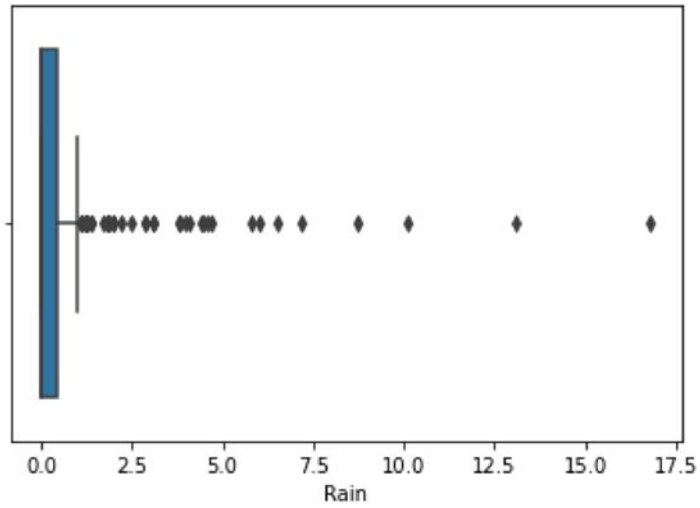
```
4.797588802426815
```

In [73]:

```
sns.boxplot(df['Rain'])
```

Out[73]:

```
<AxesSubplot:xlabel='Rain'>
```



In [74]:

```
q25 = df['Rain'].quantile(0.25)
q75 = df['Rain'].quantile(0.75)
iqr = q75-q25
upper_limit = q75 + 1.5*iqr
lower_limit = q25 - 1.5*iqr

q25,q75,iqr,lower_limit,upper_limit
```

Out[74]:

```
(0.0, 0.42500000000000004, 0.42500000000000004, -0.6375000000000001, 1.0625)
```

In [75]:

```
df[df['Rain']<lower_limit]
```

Out[75]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI

In [76]:

```
df[df['Rain']>upper_limit]
```

Out[76]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	I
1	2	6	29	61	13	1.3	64.4	4.1	7.6	1.0	
2	3	6	26	82	22	13.1	47.1	2.5	7.1	0.3	
3	4	6	25	89	13	2.5	28.6	1.3	6.9	0.0	
12	13	6	27	84	21	1.2	50.0	6.7	17.0	0.5	
14	15	6	28	80	17	3.1	49.4	3.0	7.4	0.4	
31	2	7	27	75	19	1.2	55.7	2.4	8.3	0.8	
38	9	7	32	68	14	1.4	66.6	7.7	9.2	1.1	
91	31	8	28	80	21	16.8	52.5	8.7	8.7	0.6	
92	1	9	25	76	17	7.2	46.0	1.3	7.5	0.2	
93	2	9	22	86	15	10.1	30.5	0.7	7.0	0.0	
94	3	9	25	78	15	3.8	42.6	1.2	7.5	0.1	
101	10	9	33	73	12	1.8	59.9	2.2	8.9	0.7	
102	11	9	30	77	21	1.8	58.5	1.9	8.4	1.1	
104	13	9	25	86	21	4.6	40.9	1.3	7.5	0.1	
116	25	9	26	81	21	5.8	48.6	3.0	7.7	0.4	
120	29	9	26	80	16	1.8	47.4	2.9	7.7	0.3	
121	30	9	25	78	14	1.4	45.0	1.9	7.5	0.2	
123	2	6	30	73	13	4.0	55.7	2.7	7.8	0.6	
124	3	6	29	80	14	2.0	48.7	2.2	7.6	0.3	
129	8	6	28	51	17	1.3	71.4	7.7	7.4	1.5	
134	13	6	30	52	15	2.0	72.3	11.4	7.8	1.4	
138	17	6	31	69	17	4.7	62.2	3.9	8.0	1.1	
139	18	6	33	62	10	8.7	65.5	4.6	8.3	0.9	
140	19	6	32	67	14	4.5	64.6	4.4	8.2	1.0	
143	22	6	33	46	14	1.1	78.3	8.1	8.3	1.9	
151	30	6	34	42	15	1.7	79.7	12.0	8.5	2.2	
152	1	7	28	58	18	2.2	63.7	3.2	8.5	1.2	
159	8	7	35	47	18	6.0	80.8	9.8	9.7	3.1	
160	9	7	36	43	15	1.9	82.3	9.4	9.9	3.2	
161	10	7	34	51	16	3.8	77.5	8.0	9.5	2.0	
162	11	7	34	56	15	2.9	74.8	7.1	9.5	1.6	
175	24	7	33	63	17	1.1	72.8	20.9	56.6	1.6	

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI
176	25	7	39	64	9	1.2	73.8	11.7	15.9	1.1
188	6	8	30	54	14	3.1	70.5	11.0	9.1	1.3
189	7	8	34	63	13	2.9	69.7	7.2	9.8	1.2
218	5	9	30	58	12	4.1	66.1	4.0	8.4	1.0
219	6	9	34	71	14	6.5	64.5	3.3	9.1	1.0
223	10	9	29	74	15	1.1	59.5	4.7	8.2	0.8
240	27	9	28	87	15	4.4	41.1	6.5	8.0	0.1

In [77]:

```
# caping
df['Rain'] = np.where(df['Rain']>upper_limit
         ,upper_limit
         ,np.where(df['Rain']<lower_limit
                   ,lower_limit,
                   df['Rain']
                  )
            )
```

In [78]:

```
df.shape
```

Out[78]:

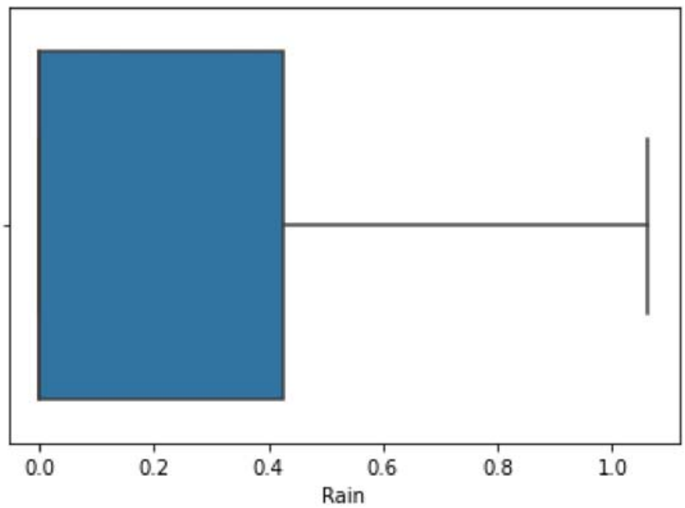
```
(240, 14)
```

In [79]:

```
sns.boxplot(df['Rain'])
```

Out[79]:

```
<AxesSubplot:xlabel='Rain'>
```

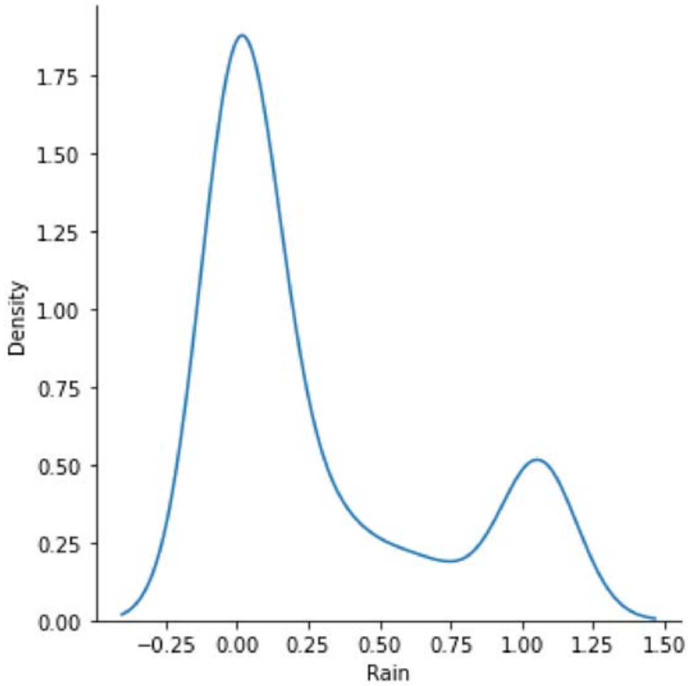


In [80]:

```
sns.displot(df['Rain'],kind='kde')
```

Out[80]:

```
&lt;seaborn.axisgrid.FacetGrid at 0x29f3ee3d640&gt;
```

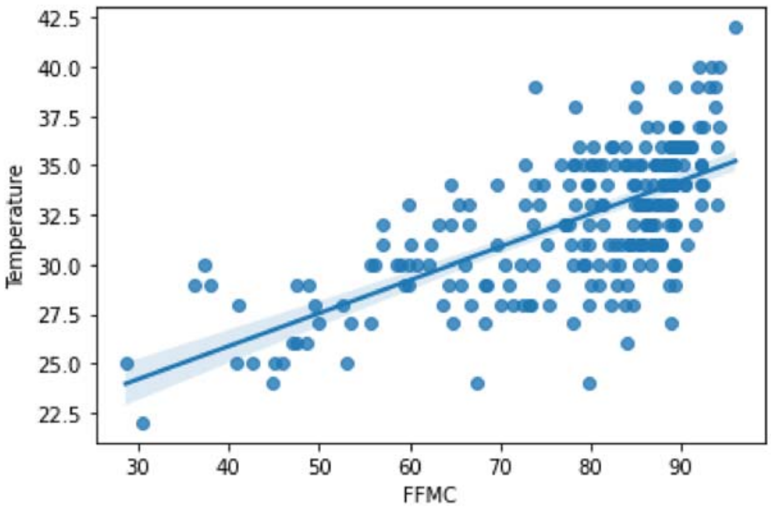


In [81]:

```
sns.regplot(x="FFMC",y="Temperature",data=df)
```

Out[81]:

```
&lt;AxesSubplot:xlabel='FFMC', ylabel='Temperature'&gt;
```

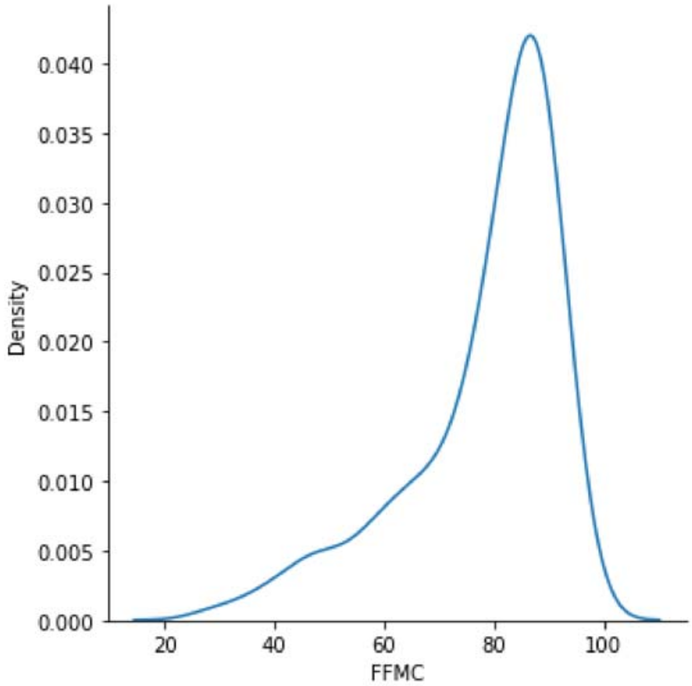


In [82]:

```
sns.displot(df['FFMC'],kind='kde')
```

Out[82]:

```
<seaborn.axisgrid.FacetGrid at 0x29f3f0e85e0>
```



In [83]:

```
df['FFMC'].skew()
```

Out[83]:

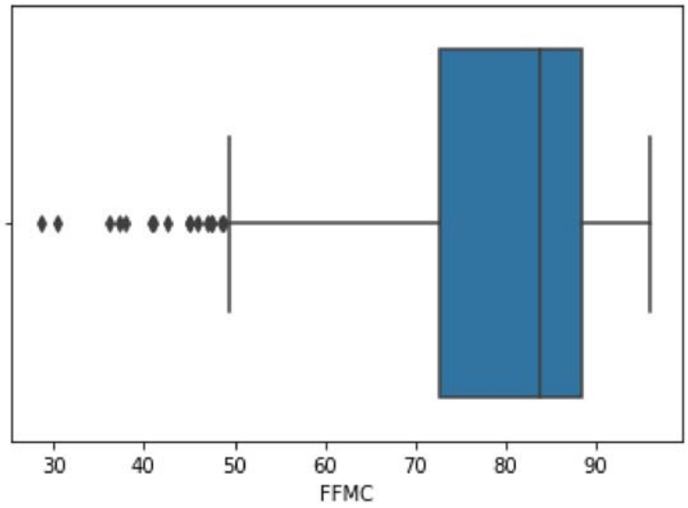
```
-1.3784701843619993
```

In [84]:

```
sns.boxplot(df['FFMC'])
```

Out[84]:

```
&lt;AxesSubplot:xlabel='FFMC'&gt;
```



In [85]:

```
df['FFMC'].describe()
```

Out[85]:

```
count    240.00000
mean      78.17125
std       14.11016
min       28.60000
25%       72.67500
50%       83.70000
75%       88.30000
max       96.00000
Name: FFMC, dtype: float64
```

In [86]:

```
q3_ffmc = df['FFMC'].quantile(0.75)
q1_ffmc = df['FFMC'].quantile(0.25)
iqr_ffmc = q3_ffmc-q1_ffmc
lower_limit_ffmc = q1_ffmc - 1.5 * iqr_ffmc
upper_limit_ffmc = q3_ffmc + 1.5 * iqr_ffmc
q1_ffmc, q3_ffmc,iqr_ffmc,lower_limit_ffmc,upper_limit_ffmc
```

Out[86]:

```
(72.675, 88.3, 15.625, 49.2375, 111.7375)
```

In [87]:

```
df[df['FFMC']< lower_limit_ffmc]
```

Out[87]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Cl
2	3	6	26	82	22	1.0625	47.1	2.5	7.1	0.3	2.7	0.1	
3	4	6	25	89	13	1.0625	28.6	1.3	6.9	0.0	1.7	0.0	
15	16	6	29	89	13	0.7000	36.1	1.7	7.6	0.0	2.2	0.0	
16	17	6	30	89	16	0.6000	37.3	1.1	7.8	0.0	1.6	0.0	
92	1	9	25	76	17	1.0625	46.0	1.3	7.5	0.2	1.8	0.1	
93	2	9	22	86	15	1.0625	30.5	0.7	7.0	0.0	1.1	0.0	
94	3	9	25	78	15	1.0625	42.6	1.2	7.5	0.1	1.7	0.0	
104	13	9	25	86	21	1.0625	40.9	1.3	7.5	0.1	1.8	0.0	
106	15	9	24	82	15	0.4000	44.9	0.9	7.3	0.2	1.4	0.0	

In [88]:

```
df[df['FFMC']> upper_limit_ffmc]
```

Out[88]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI

In [89]:

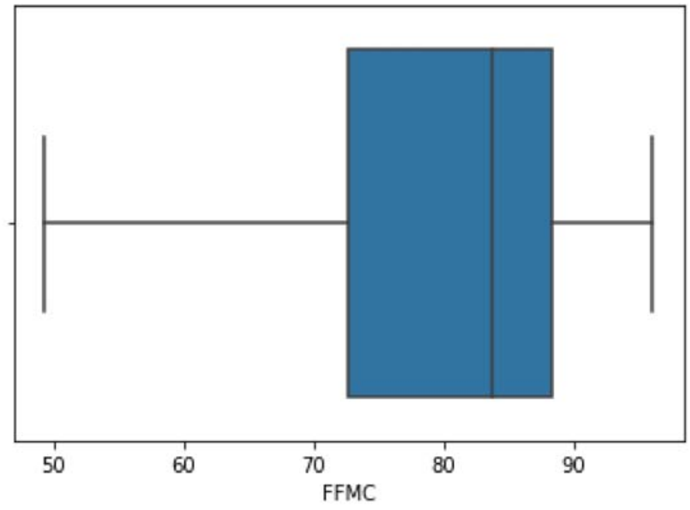
```
df['FFMC'] = np.where(
   df['FFMC']< lower_limit_ffmc
   ,lower_limit_ffmc
   ,np.where(
     df['FFMC']> upper_limit_ffmc,
     upper_limit_ffmc,
       df['FFMC']
   )
    )
```

In [90]:

```
sns.boxplot(df['FFMC'])
```

Out[90]:

<AxesSubplot:xlabel='FFMC'>

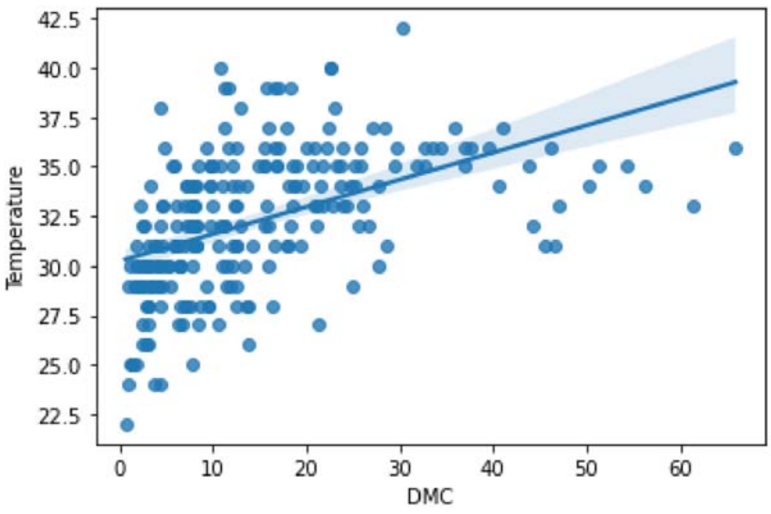


In [91]:

```
sns.regplot(x="DMC",y="Temperature",data=df)
```

Out[91]:

<AxesSubplot:xlabel='DMC', ylabel='Temperature'>

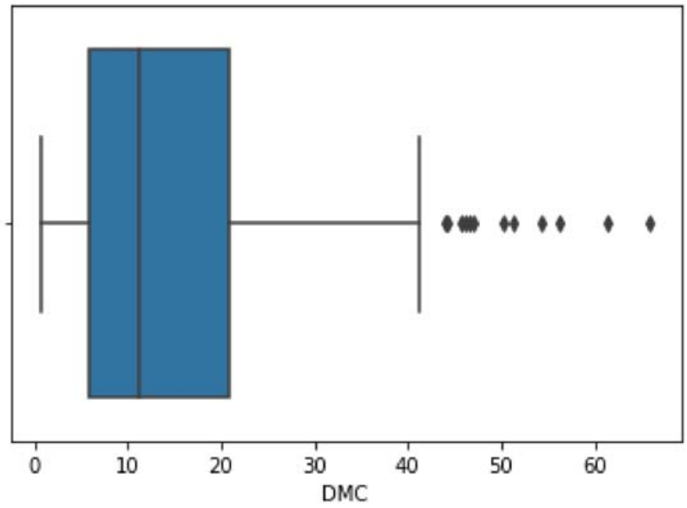


In [92]:

```
sns.boxplot(df['DMC'])
```

Out[92]:

```
&lt;AxesSubplot:xlabel='DMC'&gt;
```

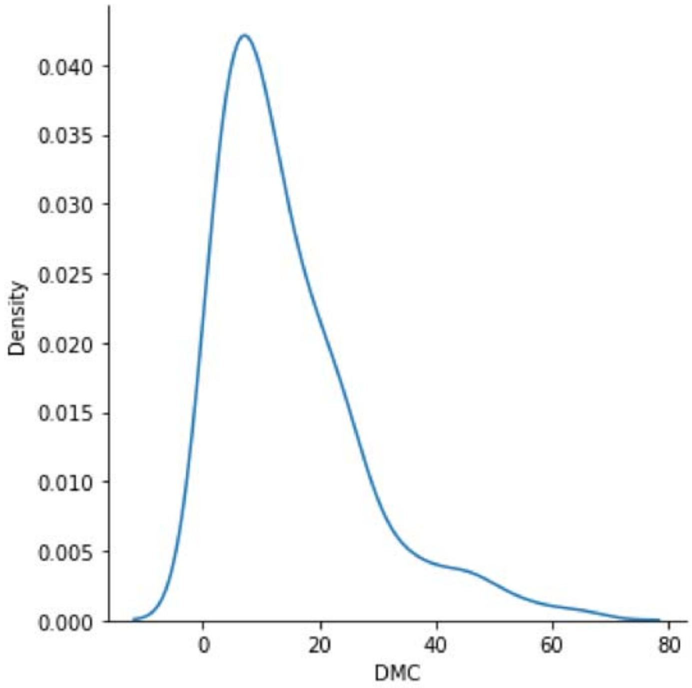


In [93]:

```
sns.displot(df['DMC'],kind='kde')
df['DMC'].skew()
```

Out[93]:

```
1.5141397270837338
```



In [94]:

```
df['DMC'].describe()
```

Out[94]:

```
count    240.000000
mean      14.795417
std       12.416723
min        0.700000
25%        5.800000
50%       11.300000
75%       20.900000
max       65.900000
Name: DMC, dtype: float64
```

In [95]:

```
q3_dmc = df['DMC'].quantile(0.75)
q1_dmc = df['DMC'].quantile(0.25)
iqr_dmc = q3_dmc-q1_dmc
lower_limit_dmc = q1_dmc - 1.5 * iqr_dmc
upper_limit_dmc = q3_dmc + 1.5 * iqr_dmc
q1_dmc, q3_dmc,iqr_dmc,lower_limit_dmc,upper_limit_dmc
```

Out[95]:

```
(5.8, 20.9, 15.099999999999998, -16.849999999999998, 43.55)
```

In [96]:

```
df[df['DMC']<lower_limit_dmc]
```

Out[96]:

day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI

In [97]:

```
df[df['DMC']>upper_limit_dmc]
```

Out[97]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI
85	25	8	35	60	15	0.0	88.9	43.9	181.3	8.2	54
86	26	8	31	78	18	0.0	85.8	45.6	190.6	4.7	57
87	27	8	33	82	21	0.0	84.9	47.0	200.2	4.4	59
88	28	8	34	64	16	0.0	89.4	50.2	210.4	7.3	62
89	29	8	35	48	18	0.0	90.1	54.2	220.4	12.5	67
173	22	7	32	48	18	0.0	91.5	44.2	90.1	13.2	44
174	23	7	31	71	17	0.0	87.3	46.6	99.0	6.9	46
205	23	8	36	43	16	0.0	91.2	46.1	137.7	11.5	50
206	24	8	35	38	15	0.0	92.1	51.3	147.7	12.2	54
207	25	8	34	40	18	0.0	92.1	56.3	157.5	14.3	59
208	26	8	33	37	16	0.0	92.2	61.3	167.2	13.1	64
209	27	8	36	54	14	0.0	91.0	65.9	177.3	10.0	68

In [98]:

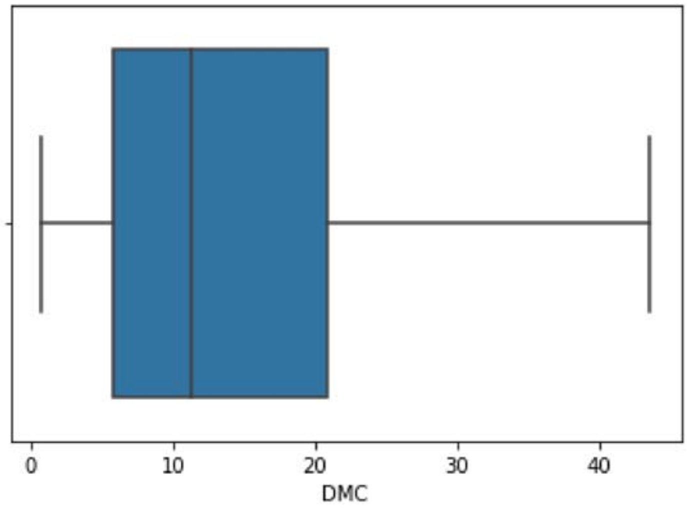
```
df['DMC'] = np.where(df['DMC']>upper_limit_dmc,upper_limit_dmc,np.where(df['DMC']
<lower_limit_dmc,lower_limit_dmc,df['DMC']))
```

In [99]:

```
sns.boxplot(df['DMC'])
```

Out[99]:

<AxesSubplot:xlabel='DMC'>

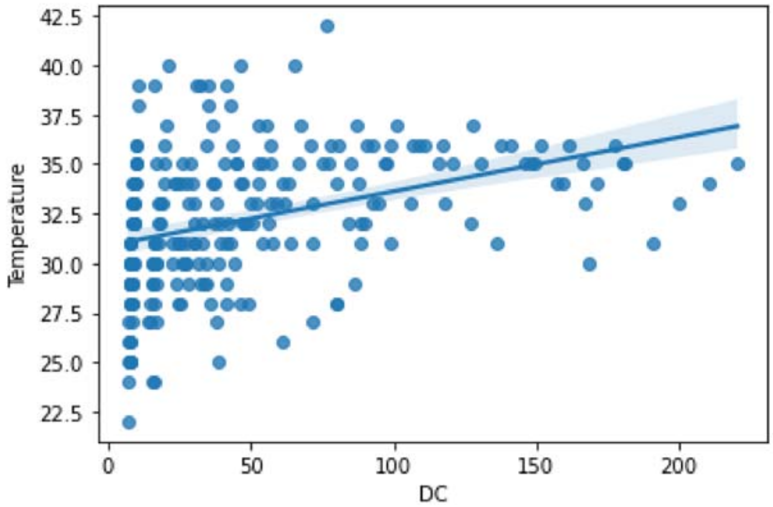


In [100]:

```
sns.regplot(x="DC",y="Temperature",data=df)
```

Out[100]:

<AxesSubplot:xlabel='DC', ylabel='Temperature'>

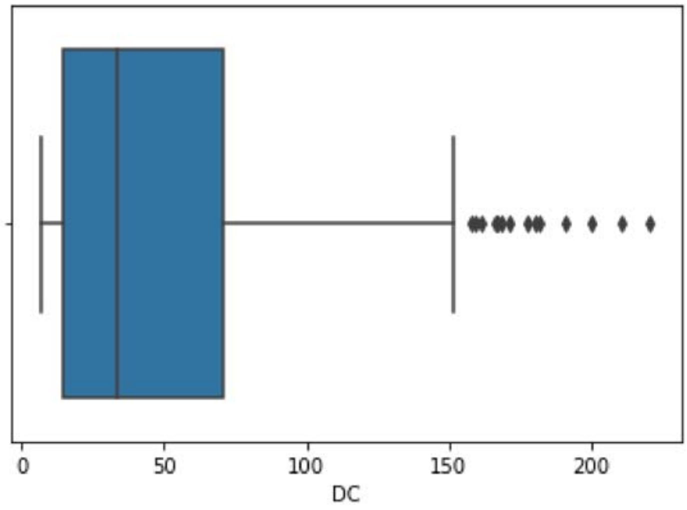


In [101]:

```
sns.boxplot(df['DC'])
```

Out[101]:

<AxesSubplot:xlabel='DC'>

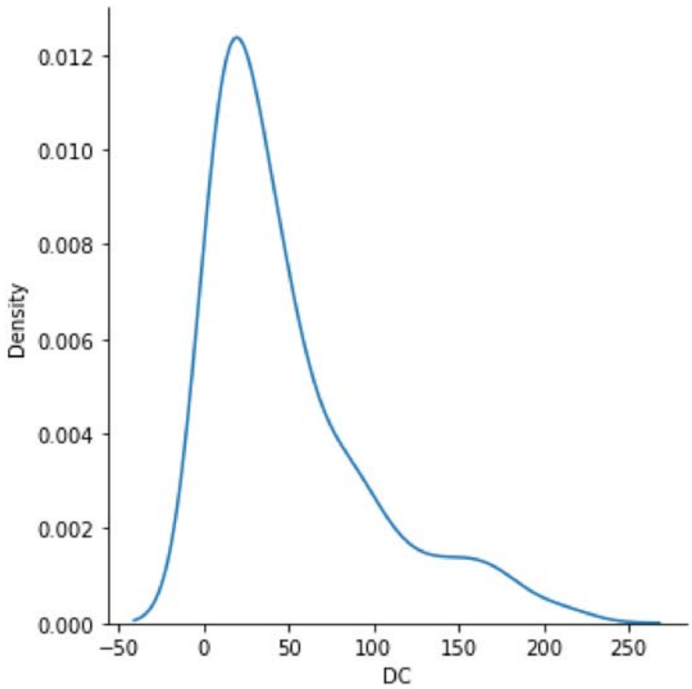


In [102]:

```
sns.displot(df['DC'],kind='kde')
```

Out[102]:

<seaborn.axisgrid.FacetGrid at 0x29f3f47eee0>



In [103]:

```
df['DC'].describe()
```

Out[103]:

```
count    240.000000
mean      49.869583
std       47.787887
min        6.900000
25%       14.575000
50%       33.750000
75%       71.075000
max      220.400000
Name: DC, dtype: float64
```

In [104]:

```
q3_dc = df['DC'].quantile(0.75)
q1_dc = df['DC'].quantile(0.25)
iqr_dc = q3_dc-q1_dc
lower_limit_dc = q1_dc - 1.5 * iqr_dc
upper_limit_dc = q3_dc + 1.5 * iqr_dc
q1_dc, q3_dc,iqr_dc,lower_limit_dc,upper_limit_dc
```

Out[104]:

```
(14.575, 71.075, 56.5, -70.175, 155.825)
```

In [105]:

```
df[df['DC']<lower_limit_dc]
```

Out[105]:

day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI

In [106]:

```
df[df['DC']>upper_limit_dc]
```

Out[106]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI
83	23	8	36	53	16	0.0	89.5	37.60	161.5	10.4	47
84	24	8	34	64	14	0.0	88.9	40.50	171.3	9.0	50
85	25	8	35	60	15	0.0	88.9	43.55	181.3	8.2	54
86	26	8	31	78	18	0.0	85.8	43.55	190.6	4.7	57
87	27	8	33	82	21	0.0	84.9	43.55	200.2	4.4	59
88	28	8	34	64	16	0.0	89.4	43.55	210.4	7.3	62
89	29	8	35	48	18	0.0	90.1	43.55	220.4	12.5	67
90	30	8	35	70	17	0.8	72.7	25.20	180.4	1.7	37
207	25	8	34	40	18	0.0	92.1	43.55	157.5	14.3	59
208	26	8	33	37	16	0.0	92.2	43.55	167.2	13.1	64
209	27	8	36	54	14	0.0	91.0	43.55	177.3	10.0	68
210	28	8	35	56	14	0.4	79.2	37.00	166.0	2.1	30
212	30	8	34	49	15	0.0	89.2	24.80	159.1	8.1	35
213	31	8	30	59	19	0.0	89.1	27.80	168.2	9.8	39

In [107]:

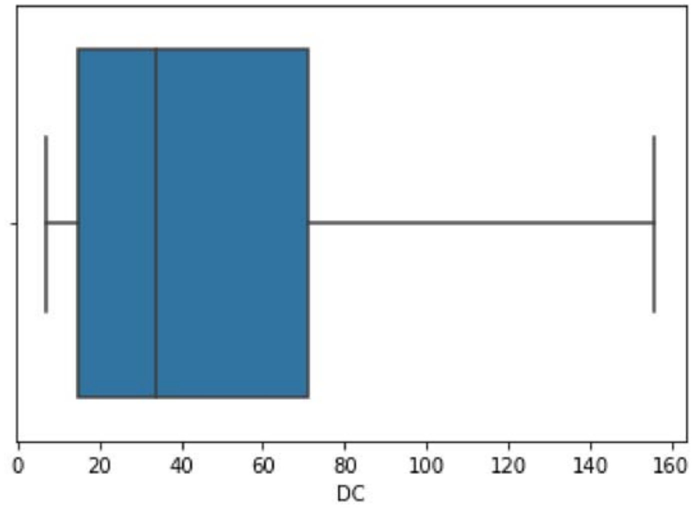
```
df['DC'] = np.where(df['DC']>upper_limit_dc,upper_limit_dc,np.where(df['DC']
<lower_limit_dc,lower_limit_dc,df['DC']))
```

In [108]:

```
sns.boxplot(df['DC'])
```

Out[108]:

```
&lt;AxesSubplot:xlabel='DC'&gt;
```

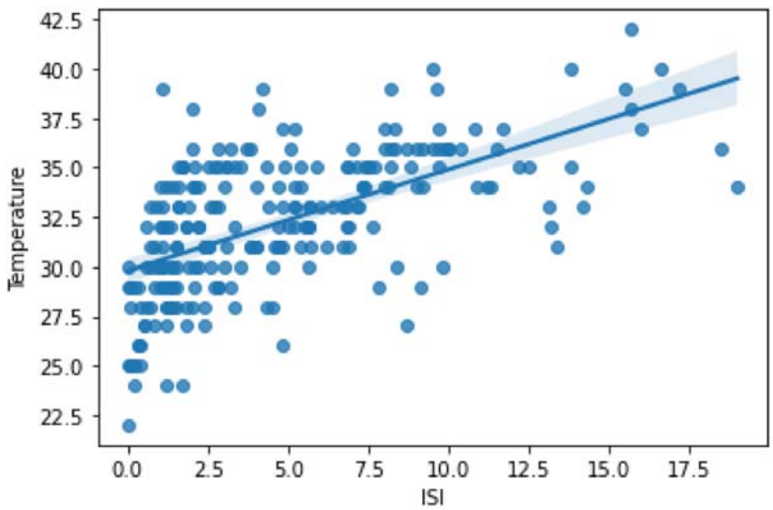


In [109]:

```
sns.regplot(x="ISI",y="Temperature",data=df)
```

Out[109]:

```
&lt;AxesSubplot:xlabel='ISI', ylabel='Temperature'&gt;
```

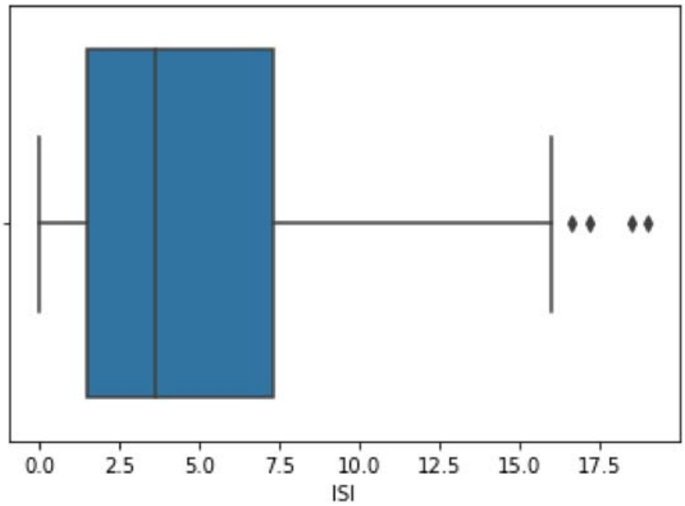


In [110]:

```
sns.boxplot(df['ISI'])
```

Out[110]:

<AxesSubplot:xlabel='ISI'>

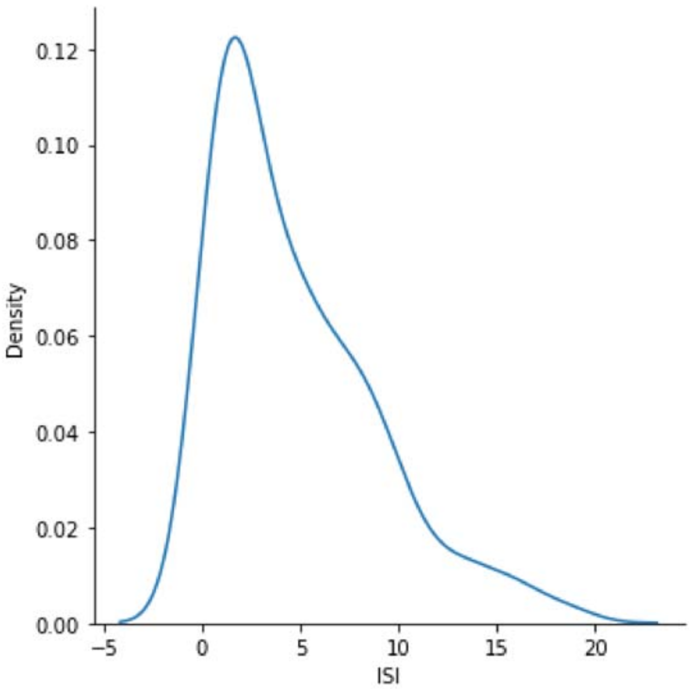


In [111]:

```
sns.displot(df['ISI'],kind='kde')
df['ISI'].skew()
```

Out[111]:

1.1311652043469416



In [112]:

```
df['ISI'].describe()
```

Out[112]:

```
count    240.000000
mean       4.795833
std        4.152327
min        0.000000
25%        1.475000
50%        3.600000
75%        7.300000
max       19.000000
Name: ISI, dtype: float64
```

In [113]:

```
q3_isi = df['ISI'].quantile(0.75)
q1_isi = df['ISI'].quantile(0.25)
iqr_isi = q3_isi-q1_isi
lower_limit_isi = q1_isi - 1.5 * iqr_isi
upper_limit_isi = q3_isi + 1.5 * iqr_isi
q1_isi, q3_isi,iqr_isi,lower_limit_isi,upper_limit_isi
```

Out[113]:

```
(1.475, 7.3, 5.824999999999999, -7.262499999999999, 16.037499999999998)
```

In [114]:

```
df[df['ISI']<lower_limit_isi]
```

Out[114]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI

In [115]:

```
df[df['ISI']>upper_limit_isi]
```

Out[115]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI
172	21	7	36	29	18	0.0	93.9	39.6	80.6	18.5	39
185	3	8	39	33	17	0.0	93.7	17.1	32.1	17.2	16
187	5	8	34	42	17	0.1	88.3	23.6	52.5	19.0	23
193	11	8	40	31	15	0.0	94.2	22.5	46.3	16.6	22

In [116]:

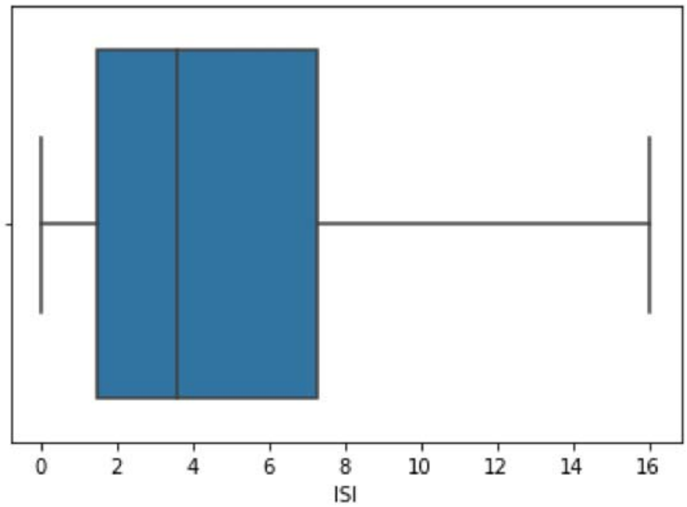
```
df['ISI'] = np.where(df['ISI']>upper_limit_isi,upper_limit_isi,np.where(df['ISI']
<lower_limit_isi,lower_limit_isi,df['ISI']))
```

In [117]:

```
sns.boxplot(df['ISI'])
```

Out[117]:

<AxesSubplot:xlabel='ISI'>

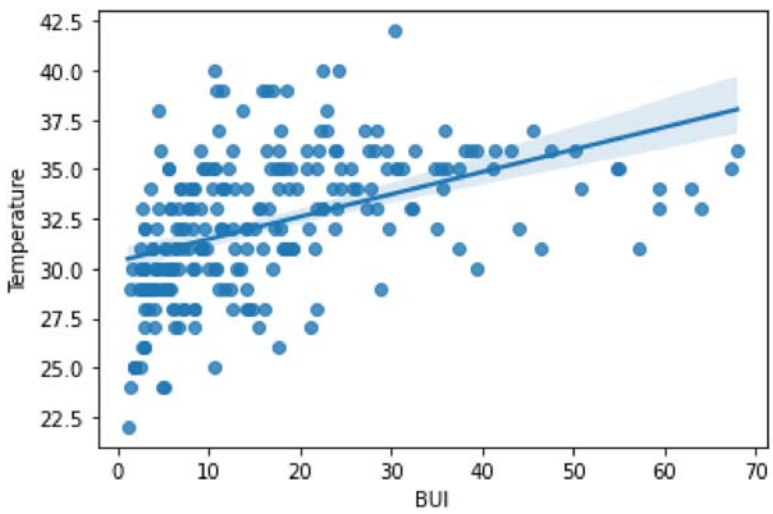


In [118]:

```
sns.regplot(x="BUI",y="Temperature",data=df)
```

Out[118]:

<AxesSubplot:xlabel='BUI', ylabel='Temperature'>

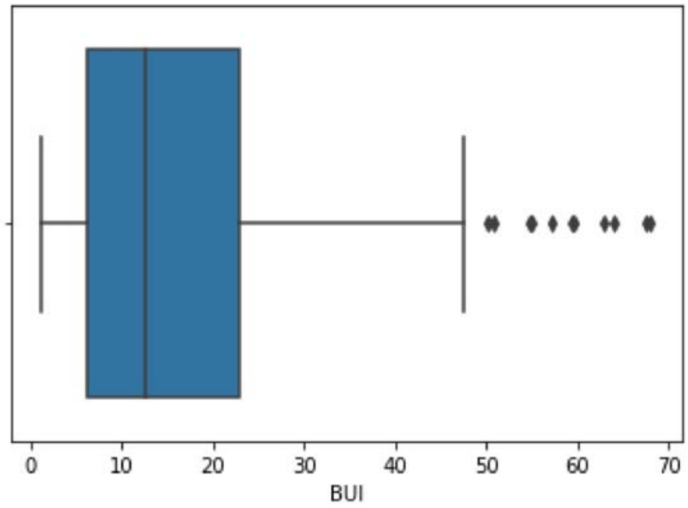


In [119]:

```
sns.boxplot(df['BUI'])
```

Out[119]:

<AxesSubplot:xlabel='BUI'>

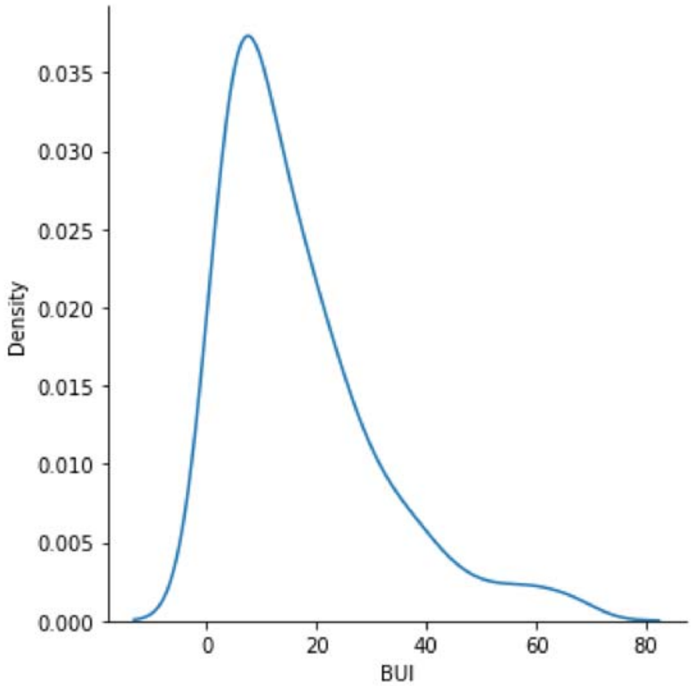


In [120]:

```
sns.displot(df['BUI'],kind='kde')
```

Out[120]:

<seaborn.axisgrid.FacetGrid at 0x29f3c97d0a0>



In [121]:

```
df['BUI'].describe()
```

Out[121]:

```
count    240.000000
mean      16.828750
std       14.254194
min        1.100000
25%        6.075000
50%       12.500000
75%       22.900000
max       68.000000
Name: BUI, dtype: float64
```

In [122]:

```
q3_bui = df['BUI'].quantile(0.75)
q1_bui = df['BUI'].quantile(0.25)
iqr_bui = q3_bui-q1_bui
lower_limit_bui = q1_bui - 1.5 * iqr_bui
upper_limit_bui = q3_bui + 1.5 * iqr_bui
q1_bui, q3_bui,iqr_bui,lower_limit_bui,upper_limit_bui
```

Out[122]:

```
(6.074999999999999, 22.9, 16.825, -19.162499999999998, 48.137499999999996)
```

In [123]:

```
df[df['BUI']<lower_limit_bui]
```

Out[123]:

day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI

In [124]:

```
df[df['BUI']>upper_limit_bui]
```

Out[124]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BU
84	24	8	34	64	14	0.0	88.9	40.50	155.825	9.0	
85	25	8	35	60	15	0.0	88.9	43.55	155.825	8.2	
86	26	8	31	78	18	0.0	85.8	43.55	155.825	4.7	
87	27	8	33	82	21	0.0	84.9	43.55	155.825	4.4	
88	28	8	34	64	16	0.0	89.4	43.55	155.825	7.3	
89	29	8	35	48	18	0.0	90.1	43.55	155.825	12.5	
205	23	8	36	43	16	0.0	91.2	43.55	137.700	11.5	
206	24	8	35	38	15	0.0	92.1	43.55	147.700	12.2	
207	25	8	34	40	18	0.0	92.1	43.55	155.825	14.3	
208	26	8	33	37	16	0.0	92.2	43.55	155.825	13.1	
209	27	8	36	54	14	0.0	91.0	43.55	155.825	10.0	

In [125]:

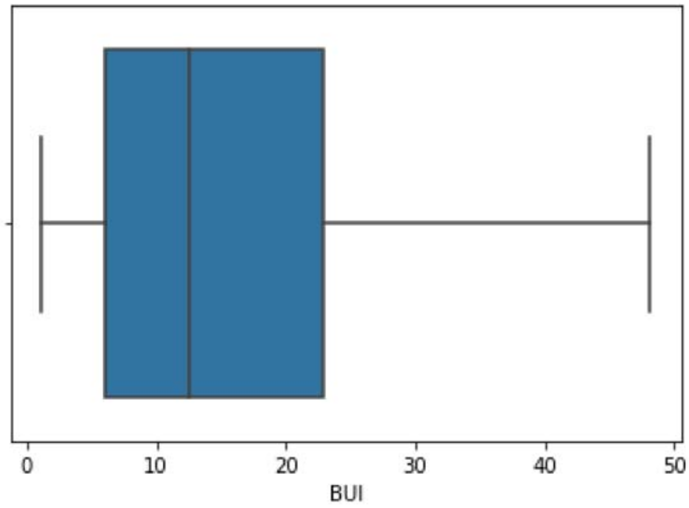
```
df['BUI'] = np.where(df['BUI']>upper_limit_bui,upper_limit_bui,np.where(df['BUI']
<lower_limit_bui,lower_limit_bui,df['BUI']))
```

In [126]:

```
sns.boxplot(df['BUI'])
```

Out[126]:

```
<AxesSubplot:xlabel='BUI'>
```

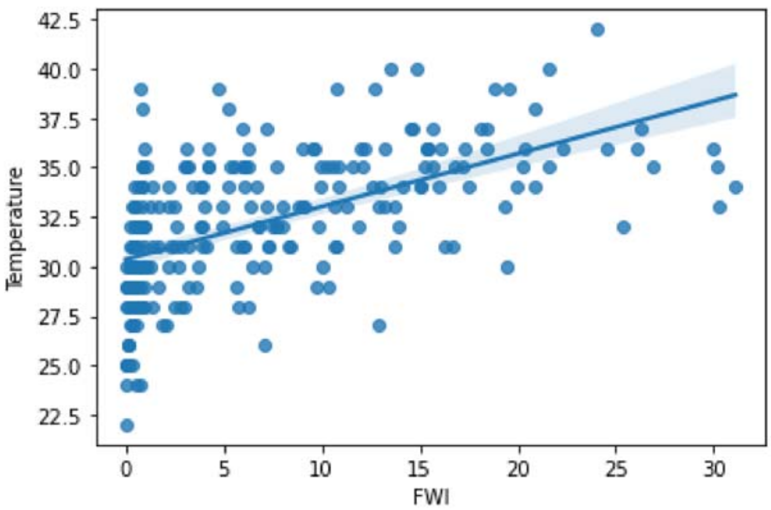


In [127]:

```
sns.regplot(x="FWI",y="Temperature",data=df)
```

Out[127]:

```
<AxesSubplot:xlabel='FWI', ylabel='Temperature'>
```

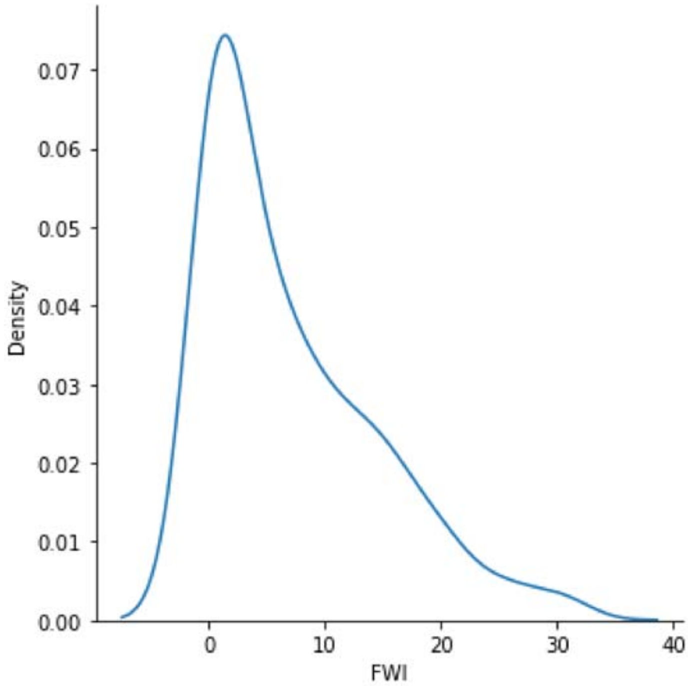


In [128]:

```
sns.displot(df['FWI'],kind='kde')
df['FWI'].skew()
```

Out[128]:

1.1338808198663723

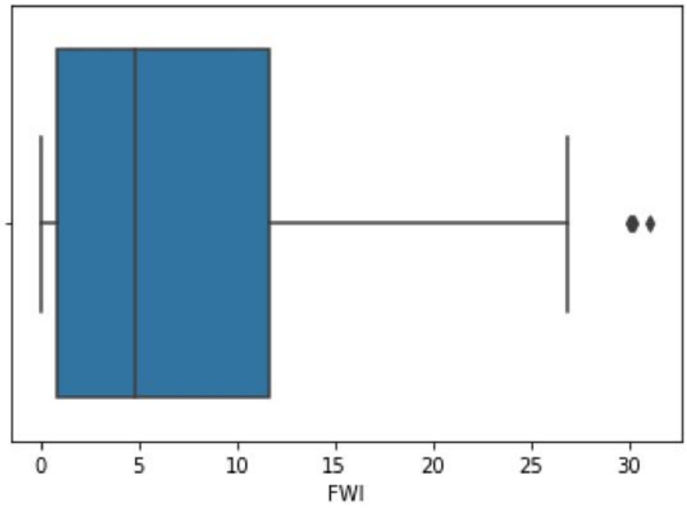


In [129]:

```
sns.boxplot(df['FWI'])
```

Out[129]:

```
<AxesSubplot:xlabel='FWI'>
```



In [130]:

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

Out[130]:

```
count    240.000000
mean       7.120417
std        7.447734
min        0.000000
25%        0.800000
50%        4.800000
75%       11.675000
max       31.100000
Name: FWI, dtype: float64
```

In [131]:

```
q3_FWI = df['FWI'].quantile(0.75)
q1_FWI = df['FWI'].quantile(0.25)
iqr_FWI = q3_FWI-q1_FWI
lower_limit_FWI = q1_FWI - 1.5 * iqr_FWI
upper_limit_FWI = q3_FWI + 1.5 * iqr_FWI
q1_FWI, q3_FWI,iqr_FWI,lower_limit_FWI,upper_limit_FWI
```

Out[131]:

```
(0.8, 11.675, 10.875, -15.5125, 27.9875)
```

In [132]:

```
df[df['FWI']<lower_limit_FWI]
```

Out[132]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI

In [133]:

```
df[df['FWI']>upper_limit_FWI]
```

Out[133]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI
89	29	8	35	48	18	0.0	90.1	43.55	155.825	12.5000
172	21	7	36	29	18	0.0	93.9	39.60	80.600	16.0375
207	25	8	34	40	18	0.0	92.1	43.55	155.825	14.3000
208	26	8	33	37	16	0.0	92.2	43.55	155.825	13.1000

In [134]:

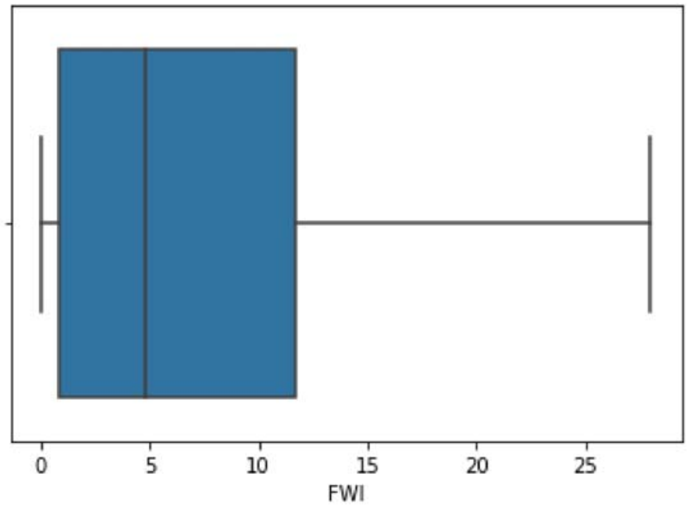
```
df['FWI'] = np.where(df['FWI']>upper_limit_FWI,upper_limit_FWI,np.where(df['FWI']
<lower_limit_FWI,lower_limit_FWI,df['FWI']))
```

In [135]:

```
sns.boxplot(df['FWI'])
```

Out[135]:

<AxesSubplot:xlabel='FWI'>



In [136]:

```
df.head()
```

Out[136]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI
0	1	6	29	57	18	0.0000	65.7000	3.4	7.6	1.3	3.
1	2	6	29	61	13	1.0625	64.4000	4.1	7.6	1.0	3.
2	3	6	26	82	22	1.0625	49.2375	2.5	7.1	0.3	2.
3	4	6	25	89	13	1.0625	49.2375	1.3	6.9	0.0	1.
4	5	6	27	77	16	0.0000	64.8000	3.0	14.2	1.2	3.

In [137]:

```
## Independent And Dependent Features
X=df.drop('Temperature',axis=1)
Y=df['Temperature']
```

In [138]:

```
X.head()
```

Out[138]:

	day	month	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Class
0	1	6	57	18	0.0000	65.7000	3.4	7.6	1.3	3.4	0.5	
1	2	6	61	13	1.0625	64.4000	4.1	7.6	1.0	3.9	0.4	
2	3	6	82	22	1.0625	49.2375	2.5	7.1	0.3	2.7	0.1	
3	4	6	89	13	1.0625	49.2375	1.3	6.9	0.0	1.7	0.0	
4	5	6	77	16	0.0000	64.8000	3.0	14.2	1.2	3.9	0.5	

In [139]:

```
Y.head()
```

Out[139]:

```
0    29
1    29
2    26
3    25
4    27
Name: Temperature, dtype: int64
```

In [140]:

```
from sklearn.model_selection import train_test_split
```

In [141]:

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.30,
random_state=10)
```

In [142]:

```
X_train.shape
```

Out[142]:

```
(168, 13)
```

In [143]:

```
y_train.shape
```

Out[143]:

```
(168,)
```

In [144]:

```
X_test.shape
```

Out[144]:

```
(72, 13)
```

In [145]:

```
y_test.shape
```

Out[145]:

```
(72,)
```

In [146]:

```
## Standardize or feature scaling the datasets
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

In [147]:

```
X_train=scaler.fit_transform(X_train)
```

In [148]:

```
X_test=scaler.transform(X_test)
```

In [149]:

```
X_train
```

Out[149]:

```
array([[-1.69100208, -1.36054109,  0.55435372, ..., -0.90607543,
        -1.10023921,  1.02409984],
       [ 0.17953849, -0.47802795,  1.20290125, ...,  0.11685567,
         0.90889326,  1.02409984],
       [ 0.29644728,  1.28699833, -1.91012689, ...,  1.16706494,
         0.90889326,  1.02409984],
       ...,
       [ 0.17953849,  1.28699833, -1.19672461, ...,  1.45348565,
         0.90889326,  1.02409984],
       [-1.22336694, -1.36054109, -0.15904856, ..., -0.81060186,
        -1.10023921,  1.02409984],
       [-0.63882301, -1.36054109,  1.07319174, ..., -0.81060186,
        -1.10023921, -0.97646729]])
```

In [150]:

```
X_test
```

```
        -5.40295374e-01, -5.10542072e-01,  9.08893259e-01,
         1.02409984e+00],
       [ 1.34862635e+00, -1.36054109e+00, -6.13031830e-01,
         9.99532040e-01, -7.13039111e-01,  8.36474601e-01,
         6.34272712e-01,  7.57697519e-01,  1.13999886e+00,
         7.51101337e-01,  1.11250861e+00,  9.08893259e-01,
        -9.76467292e-01],
       [ 6.47173637e-01,  1.28699833e+00, -4.83322324e-01,
        -1.79250995e+00, -7.13039111e-01,  7.44480477e-01,
         1.95125351e-01,  2.44709437e-01,  1.95556766e-01,
         2.43766915e-01,  1.98690159e-01,  9.08893259e-01,
        -9.76467292e-01],
       [-1.10645815e+00, -4.78027949e-01,  3.55156980e-02,
        -5.95920527e-01, -7.13039111e-01,  6.83151060e-01,
        -2.87075674e-01, -2.33926764e-01,  2.45264244e-01,
        -2.71254393e-01, -5.89606122e-03,  9.08893259e-01,
        -9.76467292e-01],
       [ 2.96447279e-01, -4.78027949e-01,  3.59789462e-01,
        -5.95920527e-01, -7.13039111e-01,  5.60492228e-01,
        -1.83746883e-01, -9.42291166e-02, -3.27314870e-03,
```

Model Training

In [151]:

```
from sklearn.linear_model import LinearRegression
```

In [152]:

```
regression=LinearRegression()
```

In [153]:

```
regression.fit(X_train,y_train)
```

Out[153]:

```
LinearRegression()
```

In [154]:

```
## print the coefficients and the intercept
print(regression.coef_)
```

```
[-0.37760471 -0.70345652 -0.26448755 -0.67637119  1.47910241  2.85813506
  0.91275964  1.18583025  0.77071416 -1.50830788 -0.01360104 -0.25762326
 -0.10293394]
```

In [155]:

```
print(regression.intercept_)
```

```
32.029761904761905
```

In [156]:

```
## PRediction for the test data
reg_pred=regression.predict(X_test)
```

In [157]:

```
reg_pred
```

Out[157]:

```
array([36.77108949, 28.85886868, 31.29354683, 35.67934625, 34.37416854,
       33.40842784, 34.00056275, 32.90728075, 25.61870671, 31.87630401,
       33.5425386 , 36.57531672, 33.52404662, 30.09349553, 36.70744534,
       34.96718201, 25.63249428, 32.72775439, 29.77171778, 34.20223326,
       35.06141672, 31.22472954, 35.74580416, 31.77878723, 33.6025462 ,
       33.53786861, 36.05631225, 35.34535436, 33.76084189, 33.06107696,
       25.52876311, 28.29988877, 33.47912972, 36.79811783, 32.65936553,
       32.96484584, 33.02788332, 33.05150391, 31.18023607, 34.08781502,
       28.32966964, 33.5839263 , 34.49737373, 34.25622573, 31.78829049,
       33.43945476, 32.95269599, 25.7046844 , 31.88777907, 33.3686528 ,
       31.84411936, 32.01285202, 33.33731847, 27.04895774, 35.80032371,
       31.64860651, 33.72726936, 34.26077274, 35.77212716, 31.98651423,
       35.34720796, 34.69055617, 22.76109168, 33.01893914, 28.74204027,
       34.42263448, 34.55429167, 32.46384871, 34.63077359, 34.01724563,
       27.08393783, 31.38042204])
```

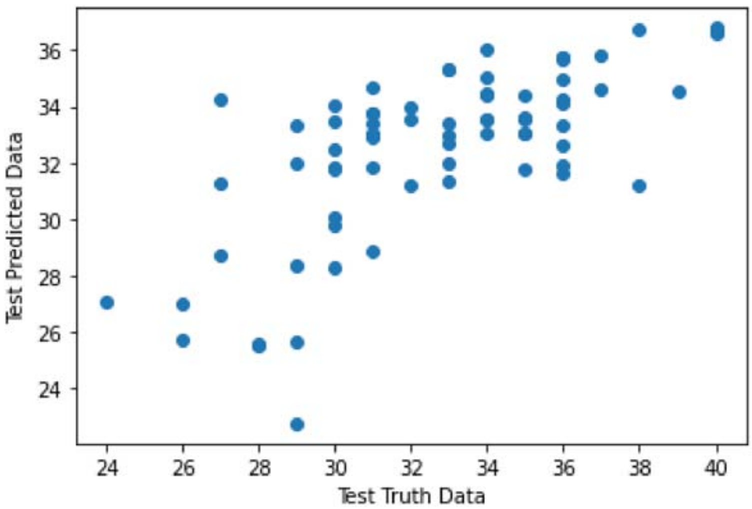
Assumptions Of Linear Regression

In [158]:

```
plt.scatter(y_test,reg_pred)
plt.xlabel("Test Truth Data")
plt.ylabel("Test Predicted Data")
```

Out[158]:

```
Text(0, 0.5, 'Test Predicted Data')
```



In [159]:

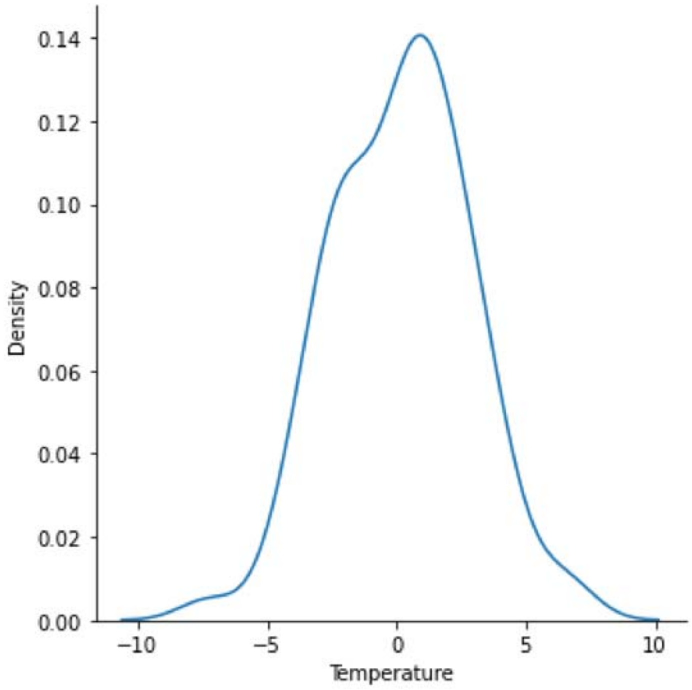
```
## residuals
residuals=y_test-reg_pred
# residuals
```

In [160]:

```
sns.displot(residuals,kind="kde")
```

Out[160]:

```
<seaborn.axisgrid.FacetGrid at 0x29f3c6c8520>
```

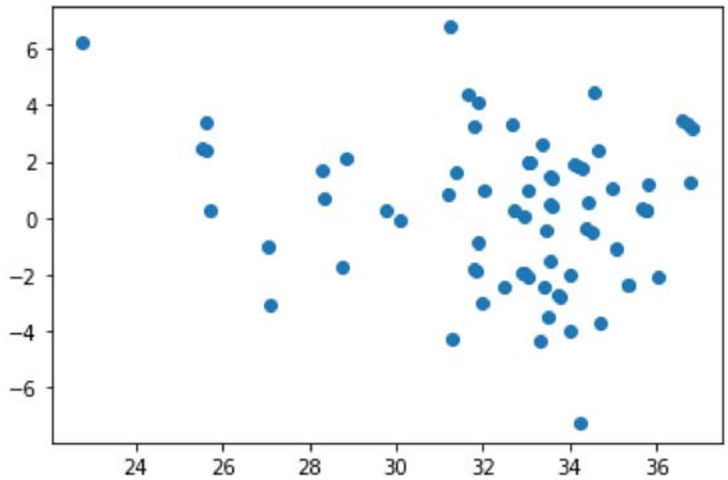


In [161]:

```
## SCatter plot with predictions and residual
##uniform distribution
plt.scatter(reg_pred,residuals)
```

Out[161]:

```
<matplotlib.collections.PathCollection at 0x29f3c675dc0>
```



In [162]:

```
## Performance Metrics
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
print(mean_squared_error(y_test,reg_pred))
print(mean_absolute_error(y_test,reg_pred))
print(np.sqrt(mean_squared_error(y_test,reg_pred)))
```

```
6.861205190879424
2.1217427713544543
2.6193902326456486
```

R square and adjusted R square

In [163]:

```
from sklearn.metrics import r2_score
score=r2_score(y_test,reg_pred)
print(score)
```

```
0.4618581177166362
```

In [164]:

```
## Adjusted R square
#display adjusted R-squared
1 - (1-score)*(len(y_test)-1)/(len(y_test)-X_test.shape[1]-1)
```

Out[164]:

```
0.34124010961864093
```

In [ ]:

```

```