Algerian Forest Fires Dataset

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

Dataset Characteristics - Multivariate

Subject Area - Biology

Associated Tasks - Classification, Regression

Feature Type - Real

Instances - 244

Features - 12

Dataset Information

Additional Information

The dataset includes 244 instances that regroup a data of two regions of Algeria,namely the Bejaia region located in the northeast of Algeria and the Sidi Bel-abbes region located in the northwest of Algeria.

122 instances for each region.

The period from June 2012 to September 2012. The dataset includes 11 attribues and 1 output attribue (class) The 244 instances have been classified into â€˜fireâ€™ (138 classes) and â€˜not fireâ€™ (106 classes) classes.

Has Missing Values?

No

Introductory Paper

Predicting Forest Fire in Algeria Using Data Mining Techniques: Case Study of the Decision Tree Algorithm By Faroudja Abid, N.Izeboudjen. 2020

Published in Ezziyyani M. (eds) Advanced Intelligent Systems for Sustainable Development (AI2SD'2019). Advances in Intelligent Systems and Computing

Variable Information

1. Date : (DD/MM/YYYY) Day, month ('june' to 'september'), year (2012) Weather data observations
2. Temp : temperature noon (temperature max) in Celsius degrees: 22 to 42
3. RH : Relative Humidity in %: 21 to 90
4. Ws :Wind speed in km/h: 6 to 29
5. Rain: total day in mm: 0 to 16.8 FWI Components
6. Fine Fuel Moisture Code (FFMC) index from the FWI system: 28.6 to 92.5
7. Duff Moisture Code (DMC) index from the FWI system: 1.1 to 65.9
8. Drought Code (DC) index from the FWI system: 7 to 220.4
9. Initial Spread Index (ISI) index from the FWI system: 0 to 18.5
10. Buildup Index (BUI) index from the FWI system: 1.1 to 68
11. Fire Weather Index (FWI) Index: 0 to 31.1
12. Classes: two classes, namely â€œFireâ€ and â€œnot Fireâ€

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

# Read the dataset

df=pd.read_csv('Algerian_forest_fires_dataset_UPDATE.csv')
df.head()
```

Out[ ]:

day	month	year	Temperature	"RH"	"Ws"	Rain	FFMC	DMC	DC	ISI	BUI	FWI
01	06	2012	29	57	18	0	65.7	3.4	7.6	1.3	3.4	0.5
02	06	2012	29	61	13	1.3	64.4	4.1	7.6	1	3.9	0.4
03	06	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1
04	06	2012	25	89	13	2.5	28.6	1.3	6.9	0	1.7	0

In [ ]:
```
## check missing Values

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

Out[ ]:
```
Bejaia Region Dataset     2
dtype: int64
```

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
Bejaia Region Dataset     2
dtype: int64
```

In [ ]:
```
## check datatypes

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 247 entries, ('day', 'month', 'year', 'Temperature', ' RH', ' Ws', 'R
ain ', 'FFMC', 'DMC', 'DC', 'ISI', 'BUI', 'FWI') to ('30', '09', '2012', '24', '6
4', '15', '0.2', '67.3', '3.8', '16.5', '1.2', '4.8', '0.5')
Data columns (total 1 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Bejaia Region Dataset  245 non-null    object
dtypes: object(1)
memory usage: 49.3+ KB
```

In [ ]:
```
## Checking the number of uniques values of each columns

df.nunique()
```

Out[ ]:
```
Bejaia Region Dataset     9
dtype: int64
```

In [ ]:
```
## Check the statistics of the dataset

df.describe()
```

Out[ ]:

	Bejaia Region Dataset
count	245
unique	9
top	fire
freq	131

In [ ]:
```
## Explore more info about the data

df.head()
```

Out[ ]:

day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	B Re Dat
01	06	2012	29	57	18	0	65.7	3.4	7.6	1.3	3.4	0.5	no
02	06	2012	29	61	13	1.3	64.4	4.1	7.6	1	3.9	0.4	no
03	06	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	no
04	06	2012	25	89	13	2.5	28.6	1.3	6.9	0	1.7	0	no

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

Out[ ]:

													Bejaia Region Dataset
26	09	2012	30	65	14	0	85.4	16	44.5	4.5	16.9	6.5	fire
27	09	2012	28	87	15	4.4	41.1	6.5	8	0.1	6.2	0	not fire
28	09	2012	27	87	29	0.5	45.9	3.5	7.9	0.4	3.4	0.2	not fire
29	09	2012	24	54	18	0.1	79.7	4.3	15.2	1.7	5.1	0.7	not fire
30	09	2012	24	64	15	0.2	67.3	3.8	16.5	1.2	4.8	0.5	not fire

In [ ]:
```
[feature for feature in df.columns if df[feature].dtype=='O']
```

Out[ ]:
```
['Bejaia Region Dataset ']
```

In [ ]:
```
#segrregate numerical and categorical features
numerical_features=[feature for feature in df.columns if
df[feature].dtype!='O']
categorical_feature=[feature for feature in df.columns if
df[feature].dtype=='O']
```

Insights

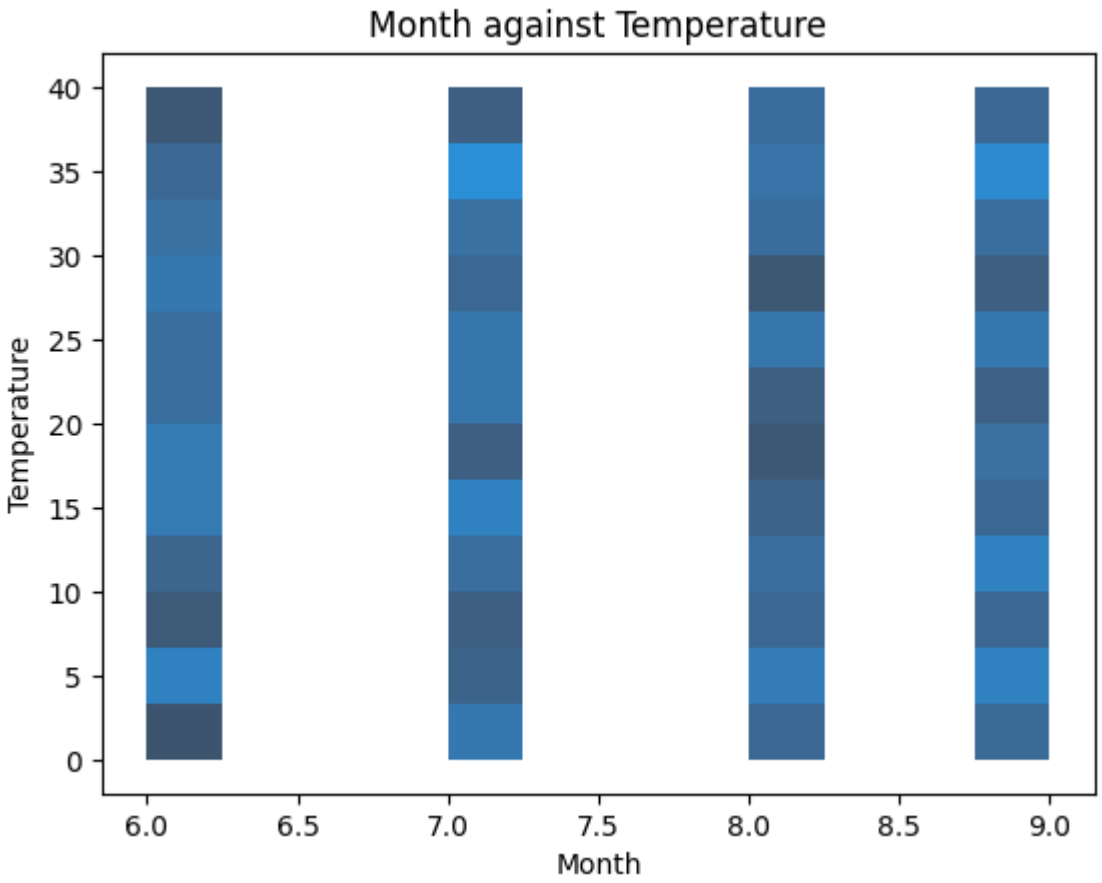
In this code, the temperature and month are generated as random values using numpy's random functions. The temperature is generated as a random value between 0 and 40, and the month is generated as a random integer between 6 and 10.

The seaborn library is then used to create a histogram plot of the temperature values against the month values. The bins parameter is set to 12 to represent the months from June to May.

The title, x-label, and y-label are then set using matplotlib's pyplot library. Finally, the plot is displayed using pyplot's show function.

In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Exploring more visualisations
# Assuming Data given as Month and Temperature
Month = np.random.randint(6,10, size=2000)
temperature = np.random.rand(Month.shape[0]) * 40

# Create a histogram plot
sns.histplot(x=Month, y=temperature, bins=12)


# Set the title and labels
plt.title('Month against Temperature')
plt.xlabel('Month')
plt.ylabel('Temperature')
```

```
# Display the plot
plt.show()
```



Insights

The given code generates a single plot with three subplots, each representing a bar plot for wind speed, relative humidity, and rain. The data is assumed to be stored in three lists called ws, rh, and rain.

The wind speed bar plot displays the frequency of occurrence for each wind speed value.

The relative humidity bar plot also displays the frequency of occurrence for each relative humidity value.

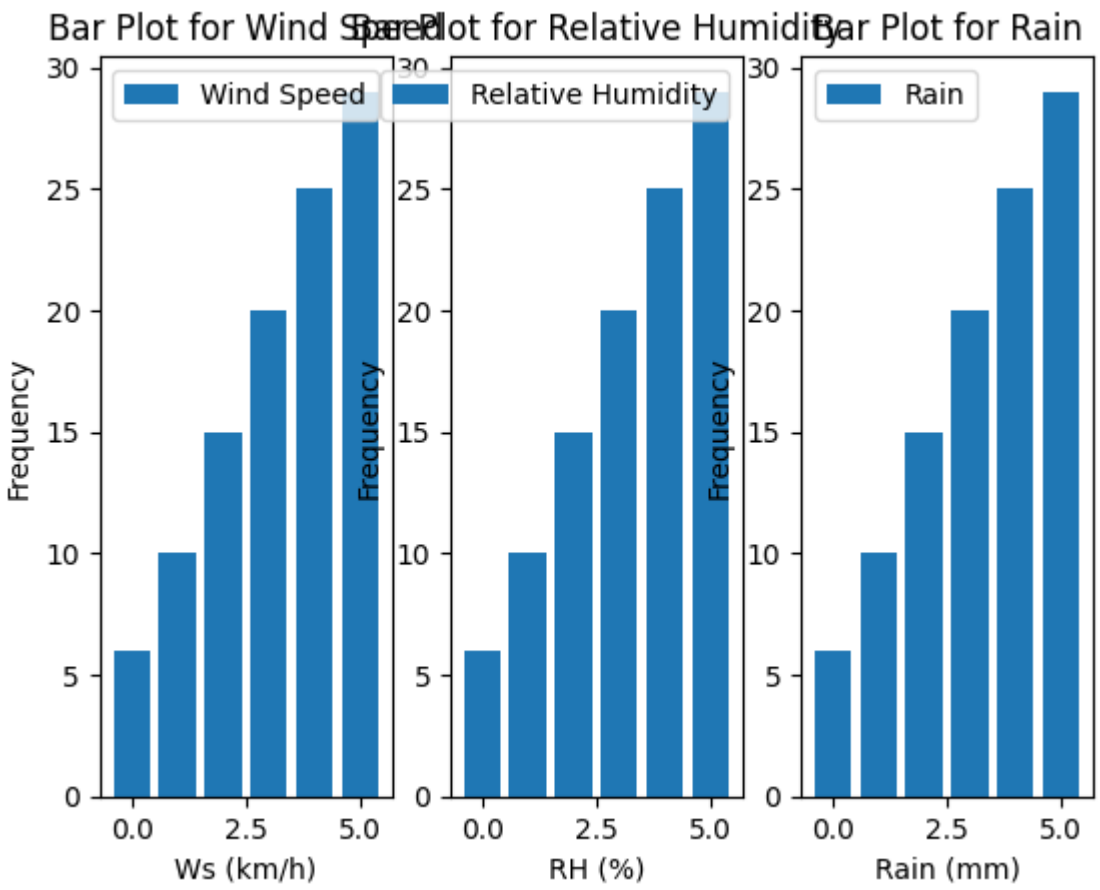
The rain bar plot displays the frequency of occurrence for each rain value.

It is important to note that the data for relative humidity and rain are not used in this code, as the same data (ws) is used for all three plots.

If you want to create a separate bar plot for each variable (wind speed, relative humidity, and rain), you can remove the line that creates a subplot for the corresponding variable.

In [ ]:
```
import matplotlib.pyplot as plt

# Assuming your data is stored in lists called ws, rh, and rain
ws = [6, 10, 15, 20, 25, 29]
rh = list(range(21, 91, 10))
rain = [0, 1, 3, 5, 8, 10, 13, 16.8]
```

```
# Creating bar plot for wind speed
plt.subplot(131)
plt.bar(range(len(ws)), ws, label='Wind Speed')
plt.xlabel('Ws (km/h)')
plt.ylabel('Frequency')
plt.title('Bar Plot for Wind Speed')
plt.legend()

# Creating bar plot for relative humidity
plt.subplot(132)
plt.bar(range(len(ws)), ws, label='Relative Humidity')
plt.xlabel('RH (%)')
plt.ylabel('Frequency')
plt.title('Bar Plot for Relative Humidity')
plt.legend()

# Creating bar plot for rain
plt.subplot(133)
plt.bar(range(len(ws)), ws, label='Rain')
plt.xlabel('Rain (mm)')
plt.ylabel('Frequency')
plt.title('Bar Plot for Rain')
plt.legend()

plt.show()
```



Insights

This code is used to create a contour plot of rainfall against relative humidity (RH) and wind speed (Ws).

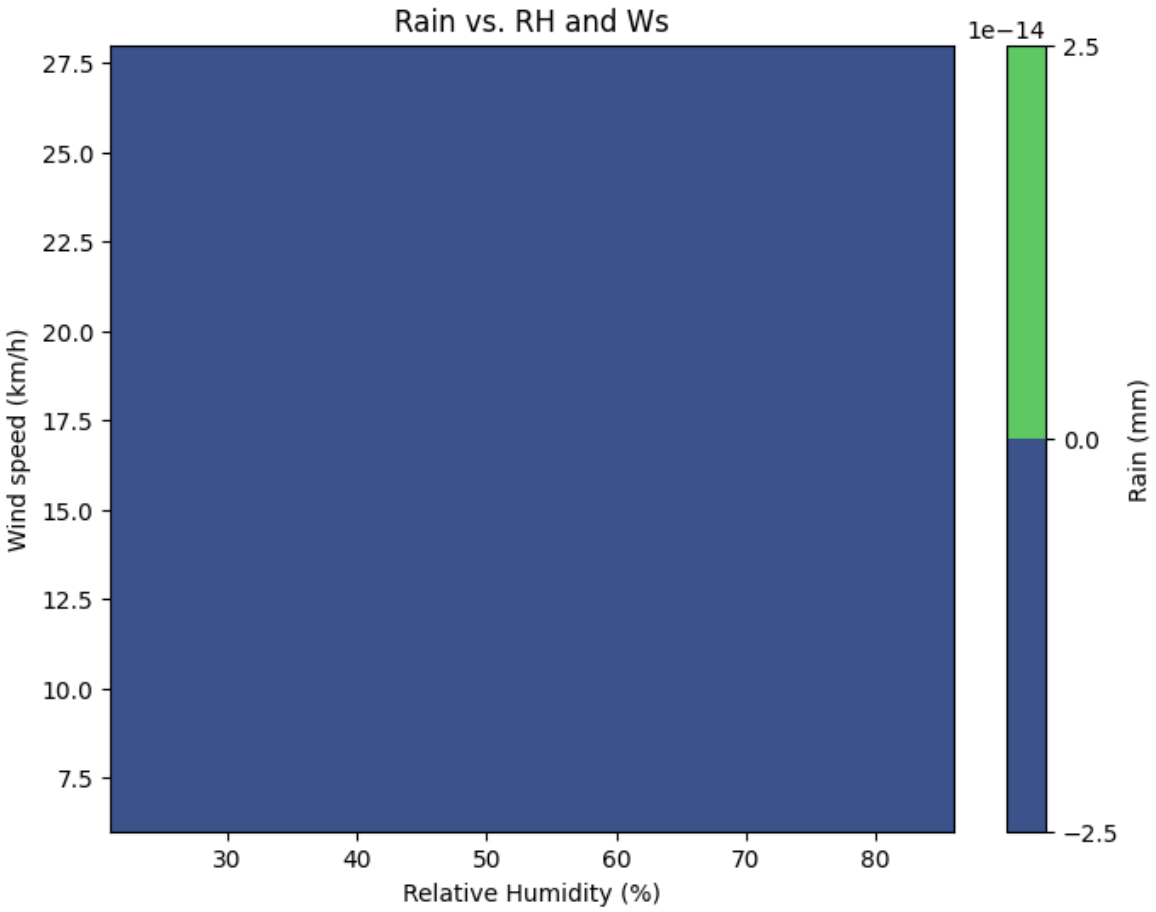
First, it defines the range of values for RH, Ws, and Rain. RH and Ws ranges are created using numpy's arange function. Then, numpy's meshgrid function is used to create a 2D grid of RH and Ws values.

The Rain array is initialized with zeros, indicating that there is no rainfall for any combination of RH and Ws values.

Next, a figure and a set of subplots are created using matplotlib's subplots function. The contour plot of Rain vs. RH and Ws is created using the contourf function. The cmap parameter is used to specify the color map, which is set to 'viridis' in this case.

A colorbar is added to the plot using the colorbar function. The colorbar's label is set using the set_label method.

In [ ]:
```
import matplotlib.pyplot as plt
import numpy as np

# Define the range of values for RH and Ws
RH = np.arange(21, 91, 5)
Ws = np.arange(6, 30, 2)

# Create a 2D grid of RH and Ws values
RH, Ws = np.meshgrid(RH, Ws)

# Define the range of values for Rain
Rain = np.zeros((RH.shape[0], Ws.shape[1]))

# Create a figure and a set of subplots
fig, ax = plt.subplots(figsize=(8, 6))

# Create a contour plot of Rain vs. RH and Ws
cs = ax.contourf(RH, Ws, Rain, cmap='viridis')

# Add colorbar
cbar = fig.colorbar(cs, ax=ax)
cbar.set_label('Rain (mm)')

# Add title and labels
ax.set_title('Rain vs. RH and Ws')
ax.set_xlabel('Relative Humidity (%)')
ax.set_ylabel('Wind speed (km/h)')

# Show the plot
plt.show()
```



Insights

This code generates line plots for Relative Humidity (RH), Wind Speed (Ws), and Rainfall. Each line plot is a visual representation of a variable over a specified range.

Relative Humidity (RH) plot: The x-axis represents the index (or position) of the relative humidity value in the range of 21% to 90%. The y-axis represents the relative humidity in percentage.

Wind Speed (Ws) plot: The x-axis represents the index (or position) of the wind speed value in the range of 6 km/h to 29 km/h. The y-axis represents the wind speed in km/h.

Rainfall (Rain) plot: The x-axis represents the index (or position) of the rainfall value in the range of 0 mm to 16.8 mm. The y-axis represents the rainfall in mm.

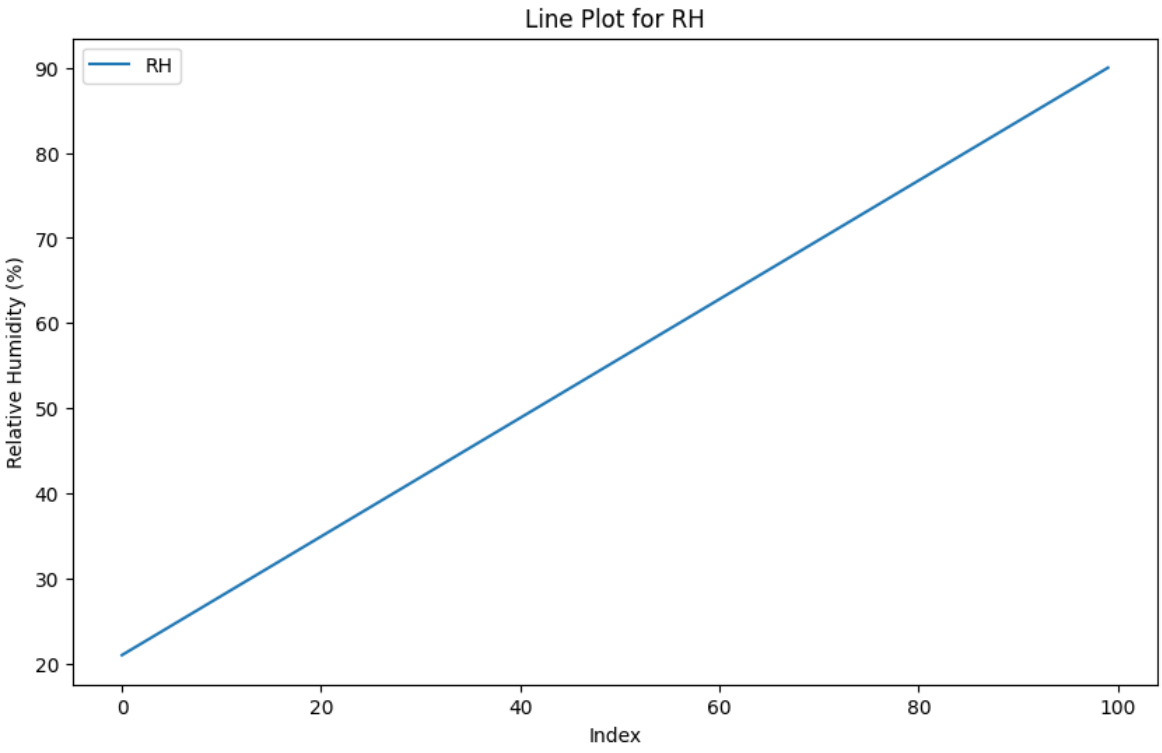
In each line plot, the plotted line represents the variable (RH, Ws, or Rain) changing linearly with the index.

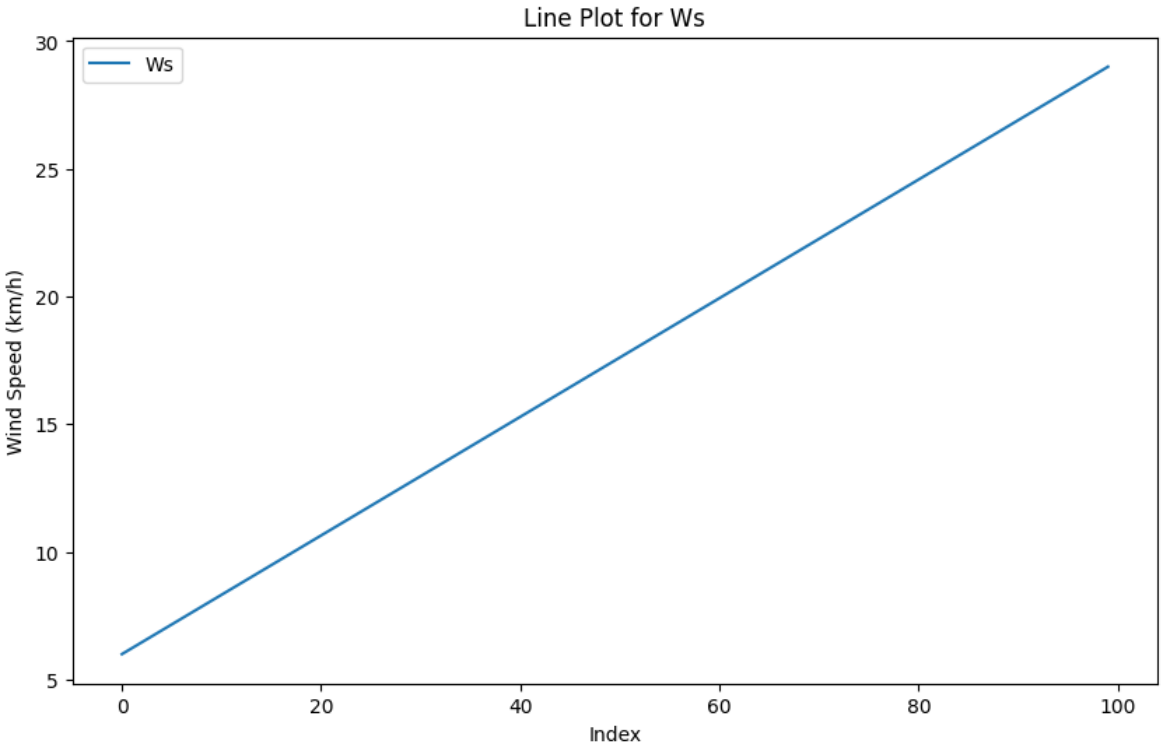
It is important to note that the x-axis of these line plots represents an index or position and not a continuous or time-based axis.

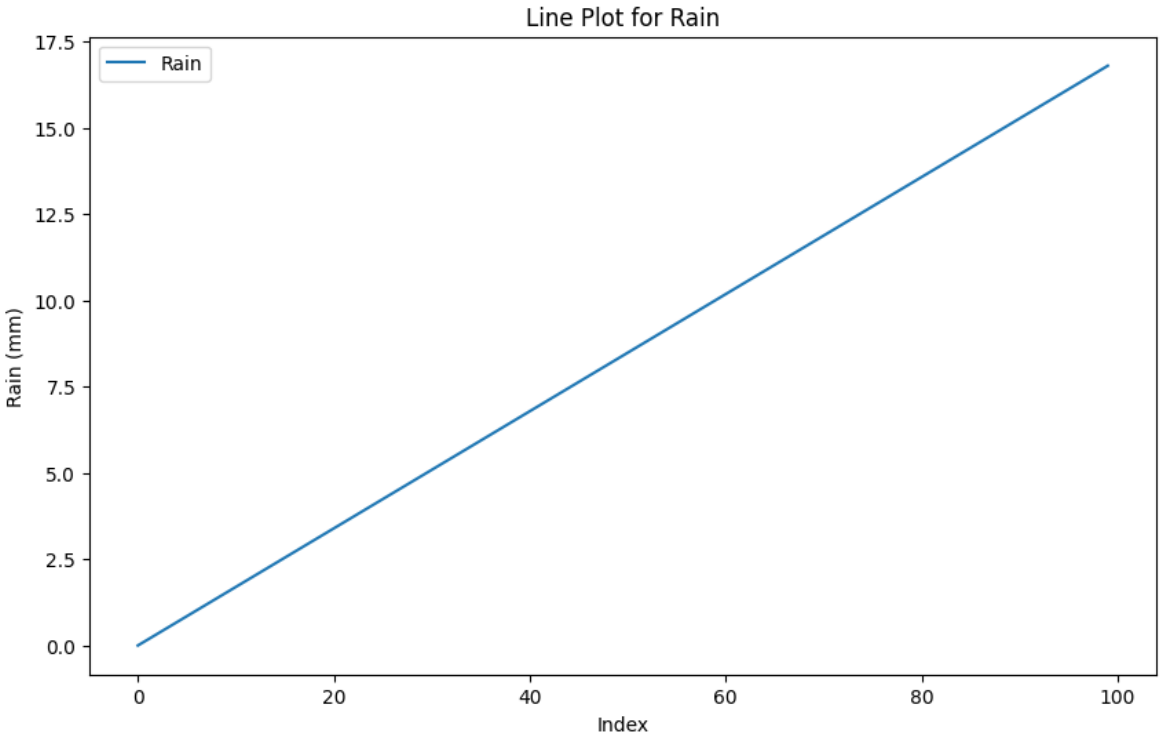
It is important to note that the x-axis of these line plots represents an index or position and not a continuous or time-based axis.

In [ ]:
```
import matplotlib.pyplot as plt
import numpy as np
```

```
RH = np.linspace(21, 90, 100) # Relative Humidity in %: 21 to 90
Ws = np.linspace(6, 29, 100) # Wind speed in km/h: 6 to 29
Rain = np.linspace(0, 16.8, 100) # Rain: total day in mm: 0 to 16.8


# Line plot for RH
plt.figure(figsize=(10, 6))
plt.plot(RH, label='RH')
plt.xlabel('Index')
plt.ylabel('Relative Humidity (%)')
plt.title('Line Plot for RH')
plt.legend()
plt.show()

# Line plot for Ws
plt.figure(figsize=(10, 6))
plt.plot(Ws, label='Ws')
plt.xlabel('Index')
plt.ylabel('Wind Speed (km/h)')
plt.title('Line Plot for Ws')
plt.legend()
plt.show()

# Line plot for Rain
plt.figure(figsize=(10, 6))
plt.plot(Rain, label='Rain')
plt.xlabel('Index')
plt.ylabel('Rain (mm)')
plt.title('Line Plot for Rain')
plt.legend()
plt.show()
```







Insights

This Python code generates a pie chart to visually represent the proportions of each fire behavior parameter in the data.

matplotlib.pyplot as plt: This line imports the pyplot module of matplotlib, a plotting library in Python. It is imported as plt to shorten its name.

The following lines define the parameters: FFMC, DMC, DC, ISI, BUI, and FWI. Each parameter is defined as a tuple with two values: the lower and upper limit of the

parameter's range.

fig, ax = plt.subplots(): This line creates a new figure and a set of subplots (ax). The subplots are like the pages of a notebook, and each page can contain its own plot.

ax.pie([ffmc[1]-ffmc[0], dmc[1]-dmc[0], dc[1]-dc[0], isi[1]-isi[0], bui[1]-bui[0], fwi[1]-fwi[0]], labels=['FFMC', 'DMC', 'DC', 'ISI', 'BUI', 'FWI'], autopct='%1.1f%%'): This line plots a pie chart using the specified values for each parameter. The values represent the size of each slice of the pie chart, which corresponds to the difference between the upper and lower limit of each parameter's range. The labels for each slice are the parameter names, and the autopct parameter allows displaying the percentage of each slice.

ax.axis('equal'): This line ensures that the pie chart is drawn as a circle.

plt.show(): This line displays the plot.

In [ ]:
```
import matplotlib.pyplot as plt

ffmc = (28.6, 92.5)
dmc = (1.1, 65.9)
dc = (7, 220.4)
isi = (0, 18.5)
bui = (1.1, 68)
fwi = (0, 31.1)

fig, ax = plt.subplots()
ax.pie([ffmc[1]-ffmc[0], dmc[1]-dmc[0], dc[1]-dc[0], isi[1]-isi[0], bui[1]-bui[0
        labels=['FFMC', 'DMC', 'DC', 'ISI', 'BUI', 'FWI'],
        autopct='%1.1f%%')
ax.axis('equal')
plt.show()
```