TITLE

Name	Venkat Yadav Moolamalla
Email:	vmf8b@umsystem.edu
Course:	CS 5402
Assignment:	Programming Assignment 02
Date:	2022-02-17


```
# Imported for data management (dataframes)
import pandas as pd
import numpy as np

# Imported for visualization
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

# Import to create train or test set in the data
from sklearn.model_selection import train_test_split
```

Concept Description:

Supplied a "forest fires" comma separated value file, determine and evaluate the properties in the given dataset. To divide the data set into two parts: one for training and one for testing. Take both data sets and compare three of the attributes using visuals and, where necessary, statistics to demonstrate that the attributes in the training set are identical to the same attribute in the test set. When choosing three characteristics, one must be Nominal or Ordinal data with no data issues, one must be Interval or Ratio with no data issues, and one must have data difficulties.

Data Collection:

The dataset has been provided as a part of our assignment.

Example Description:

coord_X
x-axis spatial coordinate within a topographical map of the area of interest.

coord_Y
y-axis spatial coordinate within a topographical map of the area of interest

y-axis spatial coordinate within a topographical map of the area of interest.

month
the month in which the forest fire happened

day
the day of the week in which the forest fire happened

FFMC
Fine Fuel Moisture Code from the Fire Weather Index (FWI) System

DMC
Duff Moisture Code from the Fire Weather Index (FWI) System

DC
Draught Code from the Fire Weather Index (FWI) System

ISI
Initial Spread Index from the Fire Weather Index (FWI) System

temp
temperature in Celsius degrees

RH
relative humidity in %

wind
wind speed in km/h

rain
outside rain in mm/m2

area
the burned area of the forest in hectares

Data Import and Wrangling:

The results of each search are read into distinct dataframes from the corresponding comma separated value file (csv). A great deal of care is taken to ensure that the data is read in as character strings.

```
# Loading the forest fire dataset

df_forest = pd.read_csv("/Users/venkatyadav/DM2/src/forestfires.csv", dtype=str)
df_forest
```

	coord_X	coord_Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0	0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18	33	0.9	0	0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0	0
3	8	6	mar	fri	91.7	33.3	77.5	9	8.3	97	4	0.2	0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
512	4	3	aug	sun	81.6	56.7	665.6	1.9	27.8	32	2.7	0	6.44
513	2	4	aug	sun	81.6	56.7	665.6	1.9	21.9	71	5.8	0	54.29
514	7	4	aug	sun	81.6	56.7	665.6	1.9	21.2	70	6.7	0	11.16
515	1	4	aug	sat	94.4	146	614.7	11.3	25.6	42	4	0	0
516	6	3	nov	tue	79.5	3	106.7	1.1	11.8	31	4.5	0	0

517 rows × 13 columns

```
df_forest.columns
```

```
Index(['coord_X', 'coord_Y', 'month', 'day', 'FFMC', 'DMC', 'DC', 'ISI',
       'temp', 'RH', 'wind', 'rain', 'area'],
      dtype='object')
```

```
# checks the datatypes of each column

df_forest.dtypes
```

```
coord_X    object
coord_Y    object
month      object
day        object
FFMC       object
DMC        object
DC         object
ISI        object
temp       object
RH         object
wind       object
rain       object


area       object
dtype: object
```

In this case, I've used characteristics to transform them to Numeric data types so that I can produce graphs.

```
# converting the given datatype to numeric

df_forest['coord_X'] = pd.to_numeric(df_forest['coord_X'], errors='coerce')
df_forest['coord_X'].dtypes
```

```
dtype('int64')
```

```
df_forest['coord_Y'] = pd.to_numeric(df_forest['coord_Y'], errors='coerce')
df_forest['coord_Y'].dtypes
```

```
dtype('int64')
```

```
df_forest['temp'] = pd.to_numeric(df_forest['temp'], errors='coerce')
df_forest['temp'].dtypes
```

```
dtype('float64')
```

```
df_forest['rain'] = pd.to_numeric(df_forest['rain'], errors='coerce')
df_forest['rain'].dtypes
```

```
dtype('float64')
```

```
df_forest['wind'] = pd.to_numeric(df_forest['wind'], errors='coerce')
df_forest['wind'].dtypes
```

```
dtype('float64')
```

```
df_forest['area'] = pd.to_numeric(df_forest['area'], errors='coerce')
df_forest['area'].dtypes
```

```
dtype('float64')
```

```
df_forest['RH'] = pd.to_numeric(df_forest['RH'], errors='coerce')
df_forest['RH'].dtypes
```

```
dtype('int64')
```

```
df_forest['FFMC'] = pd.to_numeric(df_forest['FFMC'], errors='coerce')
df_forest['FFMC'].dtypes
```

```
dtype('float64')
```

```
df_forest['DMC'] = pd.to_numeric(df_forest['DMC'], errors='coerce')
df_forest['DMC'].dtypes
```

```
dtype('float64')
```

```
df_forest['DC'] = pd.to_numeric(df_forest['DC'], errors='coerce')
df_forest['DC'].dtypes
```

```
dtype('float64')
```

```
df_forest['ISI'] = pd.to_numeric(df_forest['ISI'], errors='coerce')
df_forest['ISI'].dtypes
```

```
dtype('float64')
```

General Discussion:

Count the number of instances and attributes in the data collection.

```
# Groupby single category to find the count

df_forest.groupby('coord_X').count()
```

coord_X	coord_Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
1	48	48	48	48	48	48	48	47	48	48	48	48
2	73	73	73	73	73	73	73	73	73	73	73	73
3	55	55	55	55	55	55	55	54	55	55	55	55
4	91	91	91	91	91	91	91	91	91	91	91	91
5	30	30	30	30	30	30	30	30	30	30	30	30
6	86	86	86	86	86	86	86	86	86	86	86	86
7	60	60	60	60	60	60	60	60	60	60	60	60
8	61	61	61	61	61	61	61	61	61	61	61	61
9	13	13	13	13	13	13	13	13	13	13	13	13

```
# Groupby multiple categories to find count

df_forest.groupby(['day','coord_X','coord_Y','FFMC','DMC','DC','ISI','RH','month','temp','rai
```

day	coord_X	coord_Y	FFMC	DMC	DC	ISI	RH	month	temp	rain	area	wind
fri	**1**	**2**	**90.1**	**108.0**	**529.8**	**12.5**	**66**	**aug**	**14.7**	**0.0**	**0.00**	**2.7**

			90.7	80.9	368.3	16.8	78	jul	14.8	0.0	0.00	8.0
			91.0	166.9	752.6	7.1	41	aug	25.9	0.0	0.00	3.6
							73	aug	18.5	0.0	0.00	8.5
		3	91.1	91.3	738.1	7.2	46	sep	19.1	0.0	0.33	2.2
...	...	...	...	...	...	...	...	...	...	...	...	...
wed	8	6	93.1	157.3	666.7	13.5	28	aug	28.7	0.0	0.00	2.7
			95.2	217.7	690.0	18.0	29	aug	28.2	0.0	5.86	1.8
		8	91.7	191.4	635.9	7.8	36	aug	26.2	0.0	185.76	4.5
	9	5	93.3	49.5	297.7	14.0	34	jun	28.0	0.0	0.00	4.5
											8.16	4.5

511 rows × 0 columns

```
# Example
# Returns the burnt area in hectares

pd.DataFrame(df_forest.area.value_counts())
```

	area
0.00	247
1.94	3
0.52	2
3.71	2
0.68	2
...	...
105.66	1
154.88	1
196.48	1
200.94	1
11.16	1

251 rows × 1 columns

```
df_forest.nunique()
```

```
coord_X      9
coord_Y      7
month       12
```

```
month      11
day         7
FFMC      108
DMC       215
DC        219
ISI       119
temp      192
RH         75
wind       21
rain        7
area      251
dtype: int64
```

1. Determine whether the data is Nominal, Ordinal, Interval, or Ratio:

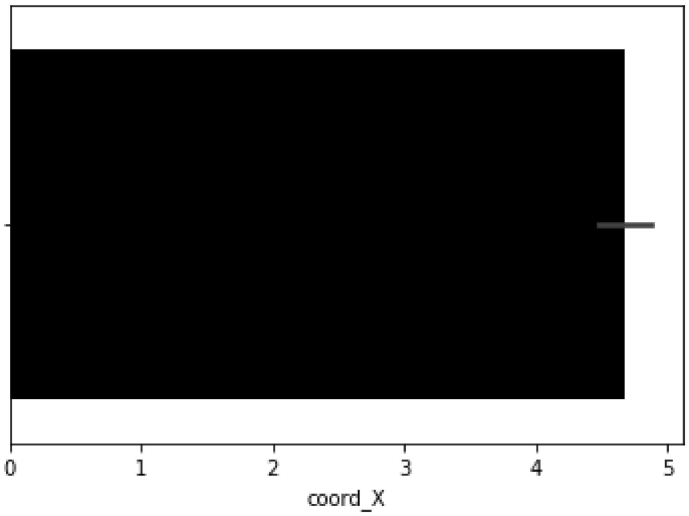
According to my understanding, I have integrated the qualities Nominal, Ordinal, Interval, and Ratio data from the given dataset.

image-3.png

2. Create a visual representation of the data for each attribute.

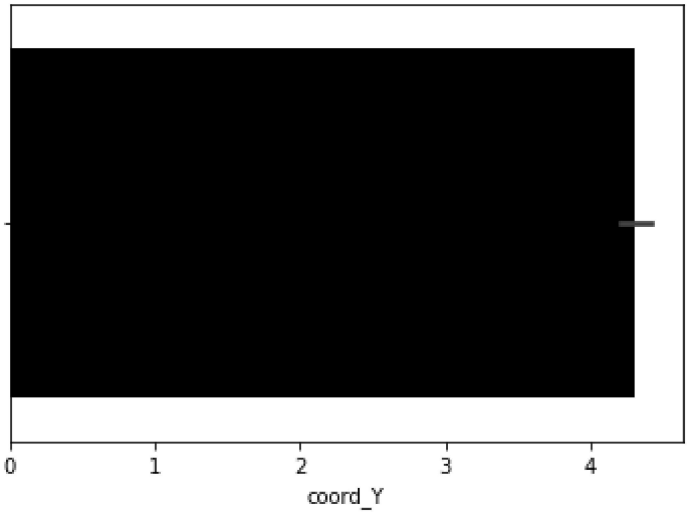
```
# The following are the details of a bar graph for a single attribute.
# Coord_X
sns.barplot(x='coord_X',data=df_forest,color='Black')
```

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



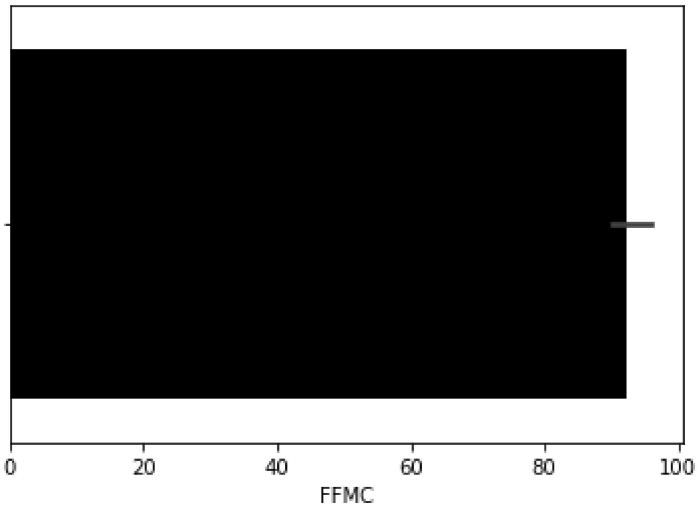

```
#coord_Y

sns.barplot(x='coord_Y',data=df_forest,color='Black')
```

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



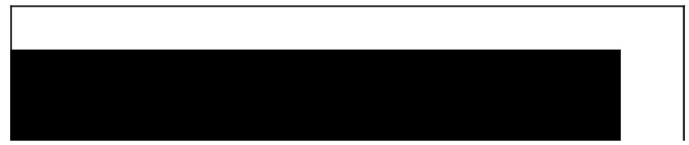

```
# FFMC

sns.barplot(x='FFMC',data=df_forest,color='Black')
```

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



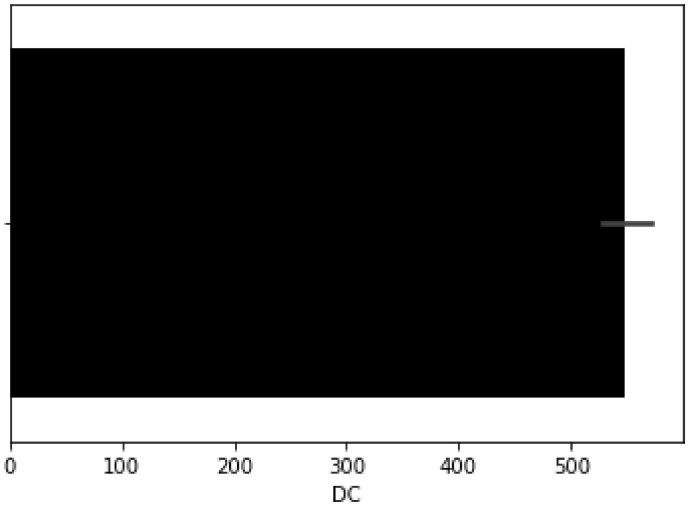

```
#DMC

sns.barplot(x='DMC',data=df_forest,color='Black')
```

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



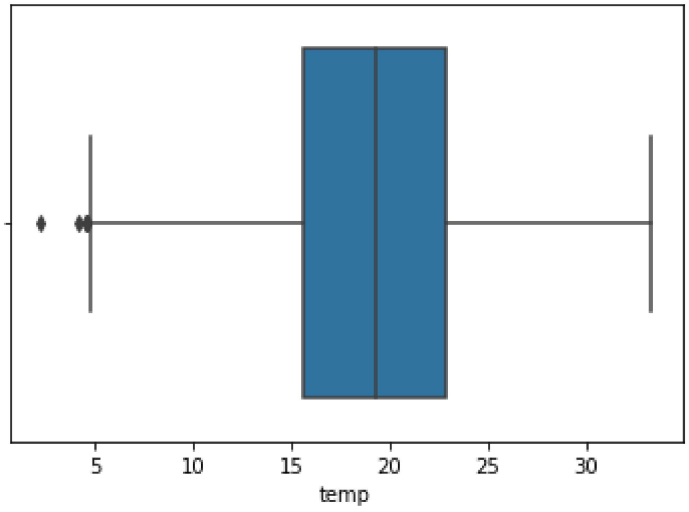
```
# DC

sns.barplot(x='DC',data=df_forest,color='Black')
```

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



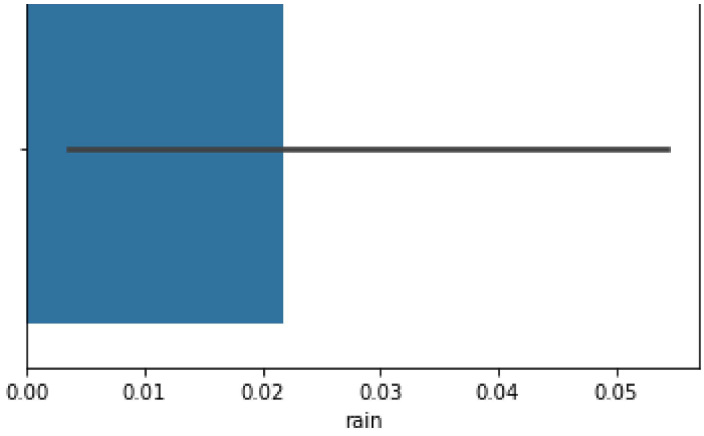

```
# Temperature

sns.boxplot(x=df_forest["temp"],)
```

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



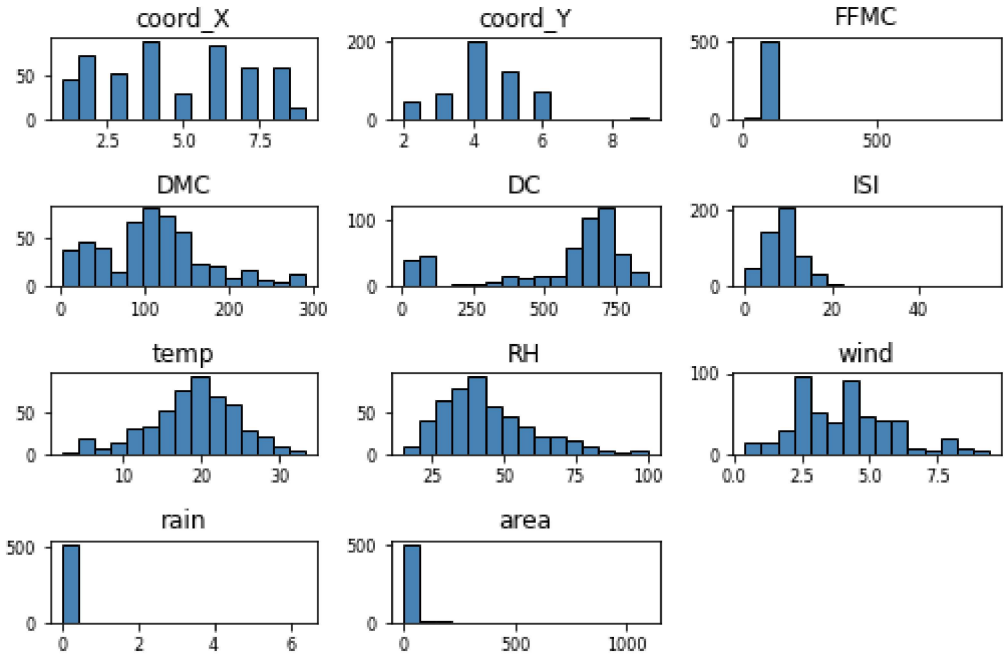

```
# Rain

sns.barplot(x=df_forest['rain'])
```

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



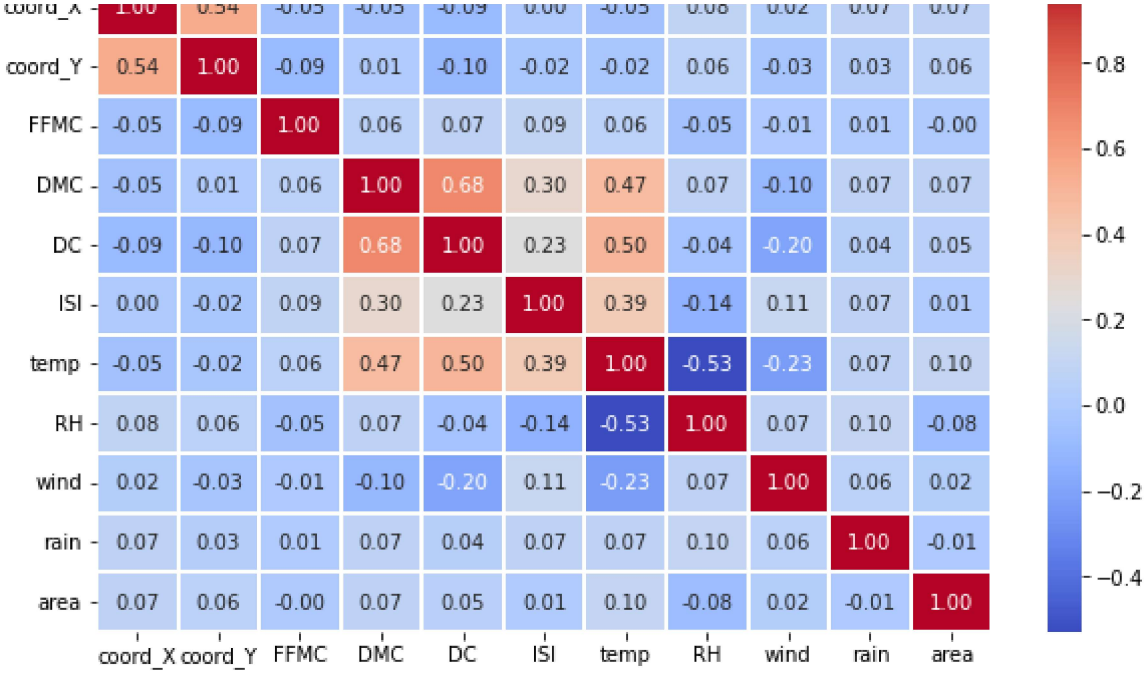

```
#sns.catplot(x="temp", y="area",hue="wind", col="temp",data=df_forest, kind="bar",height=4, a
```

```
# Histogram plot for various attributes

df_forest.hist(bins=15, color='steelblue', edgecolor='black', linewidth=1.0,
           xlabelsize=8, ylabelsize=8, grid=False)
plt.tight_layout(rect=(0, 0, 1.2, 1.2))
```




```
# Heatmap to find the relation between parameters

f, ax = plt.subplots(figsize=(10, 6))
corr = df_forest.corr()

sns.heatmap(round(corr,2), annot=True, ax=ax, cmap="coolwarm",fmt='.2f',
                 linewidths=.05)
```

```
<AxesSubplot:>
```



3. Determine the possible values for each attribute, often known as the "range of values."

```
# To determine the value range for each attribute

df_forest.coord_X.unique()

#df_forest.nunique()
```

```
array([7, 8, 6, 5, 4, 2, 9, 1, 3])
```

```
# To determine the range of values for the attribute, I utilized the lambda function.

print(df_forest.apply(lambda col: col.unique()))
```

```
coord_X                         [7, 8, 6, 5, 4, 2, 9, 1, 3]
coord_Y                               [5, 4, 6, 3, 2, 9, 8]
month      [mar, oct, aug, sep, apr, jun, jul, feb, jan, ...
day                      [fri, tue, sat, sun, mon, wed, thu]
FFMC       [86.2, 90.6, 91.7, 89.3, 92.3, 91.5, 91.0, 92....
DMC        [26.2, 35.4, 43.7, 33.3, 51.3, 85.3, 88.9, 145...
DC         [94.3, 669.1, 686.9, 77.5, 102.2, 488.0, 495.6...
ISI        [5.1, 6.7, 9.0, 9.6, 14.7, 8.5, 10.7, 7.0, 7.1...
temp       [8.2, 18.0, 14.6, 8.3, 11.4, 22.2, 24.1, 8.0, ...
RH         [51, 33, 97, 99, 29, 27, 86, 63, 40, 38, 72, 4...
wind       [6.7, 0.9, 1.3, 4.0, 1.8, 5.4, 3.1, 2.2, 7.2, ...
rain                     [0.0, 0.2, 1.0, 6.4, 0.8, 0.4, 1.4]
area       [0.0, 0.36, 0.43, 0.47, 0.55, 0.61, 0.71, 0.77...
dtype: object
```

4. Determine if there are any data values that we maybe concerned about and state whythey are of concern. If/when concerns are discovered,suggest what can be done to address those concerns.

```
# To check if there are null values

df_forest.isna().any()
```

```
coord_X    False
coord_Y    False
month      False
day        False
FFMC       False
DMC        False
DC         False
ISI        False
temp       False
RH         False
wind       False
rain       False
area       False
dtype: bool
```

By using isna().any(), we can see that "temp" returns boolean object as True. This denotes that the attribute "temp" has null values.

```
# Suggestion is to remove the "Nan" values present in temperature attribute

df_forest.dropna(inplace=True)
```

Partition the data set into training data set and test data set.

80% train data and 20% test data

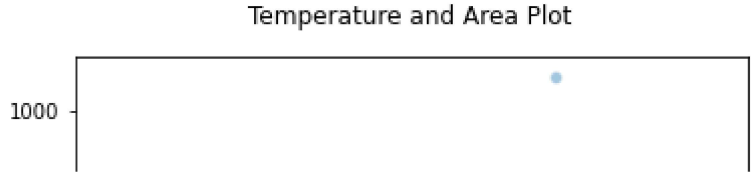
```
#  defines the values contained in the array correspond to the numeric parameters start, stop

X,y = np.arange(10).reshape((5, 2)), range(5)

X_train, X_test,y_train,y_test = train_test_split( X,y, train_size = 0.80, random_state=42)

X_train
```

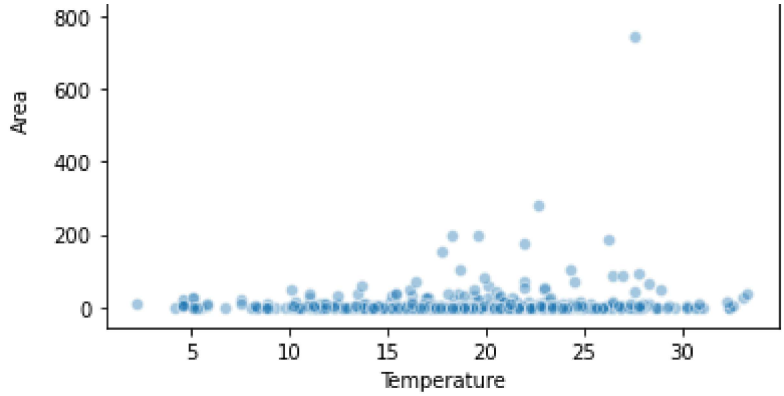
```
array([[8, 9],
       [4, 5],
```

```
            [0, 1],
            [6, 7]])


X_test

     array([[2, 3]])
```

1. Here from the train and test data , I considered "temp", "area" attributes to plot the graph. As temp is an interval measurement and area is ratio measurement.

```
#Scatter plot for Two Attributes

#sns.barplot(x="temp", y="area", data=df_forest,color="blue", saturation=.5)

plt.scatter(df_forest['temp'], df_forest['area'],
            alpha=0.4, edgecolors='w')

plt.xlabel('Temperature')
plt.ylabel('Area')
plt.title('Temperature and Area Plot',y=1.05)


# Joint Plot
jp = sns.jointplot(x='temp', y='area', data=df_forest,kind='reg', space=0, height=5, ratio=4)
```





2. Visualizing for three attributes

```
# Considering 3 attributes - Temp (data issue), Area(Ratio) and Wind (Nominal) data

fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')

xs = df_forest['temp']
ys = df_forest['area']
zs = df_forest['wind']
ax.scatter(xs, ys, zs, s=50, alpha=0.6, edgecolors='b')

ax.set_xlabel('Temperature in Celsius')
ax.set_ylabel('The burned area of the forest in hectares')
ax.set_zlabel('Wind speed in km/h')
```

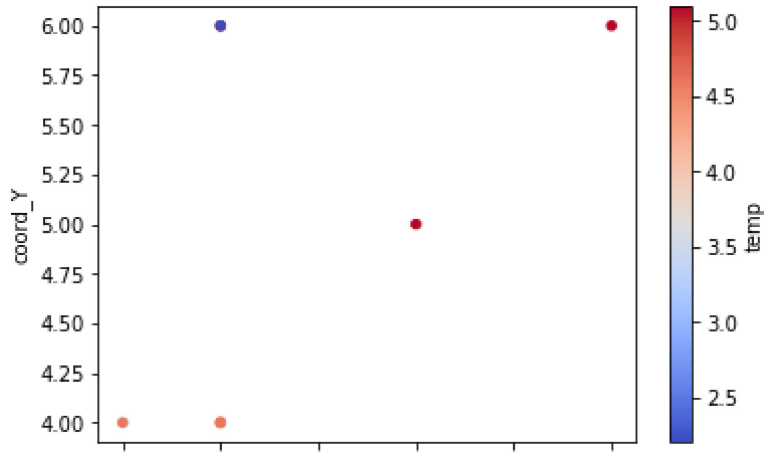
```
Text(0.5, 0, 'Wind speed in km/h')
```

Exploratory Data Analysis:

```
# summarizing the data excluding the Nan values

df_forest.describe()
```

	coord_X	coord_Y	FFMC	DMC	DC	ISI	temp
count	515.000000	515.000000	515.000000	515.000000	515.000000	515.000000	515.000000
mean	4.679612	4.302913	92.101165	111.001553	548.591845	9.035728	18.895922
std	2.311423	1.227716	37.182684	64.055503	247.560609	4.561636	5.815985
min	1.000000	2.000000	9.900000	1.100000	7.900000	0.000000	2.200000
25%	3.000000	4.000000	90.200000	69.150000	439.300000	6.500000	15.550000
50%	4.000000	4.000000	91.600000	108.300000	664.200000	8.400000	19.300000
75%	7.000000	5.000000	92.900000	142.400000	713.900000	10.900000	22.800000
max	9.000000	9.000000	921.000000	291.300000	860.600000	56.100000	33.300000

```
# scatter plot for single month

df_forest[df_forest.month=='dec'].plot(kind='scatter', x='coord_X', y='coord_Y', c='temp', cm
plt.show()
```




```
# Correlation Matrix between attributes using Spearman Method


corr_matrix = df_forest.corr(method='spearman')
corr_matrix
```

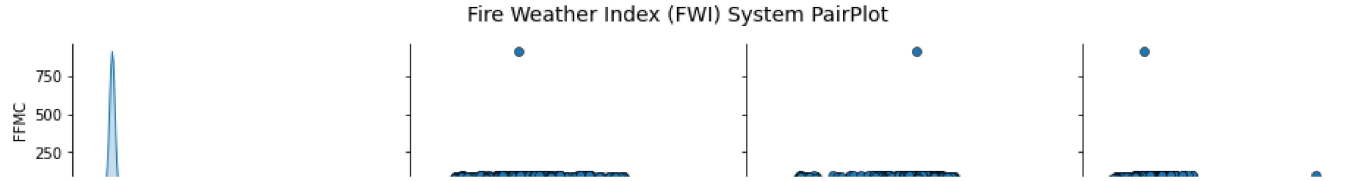
	coord_X	coord_Y	FFMC	DMC	DC	ISI	temp

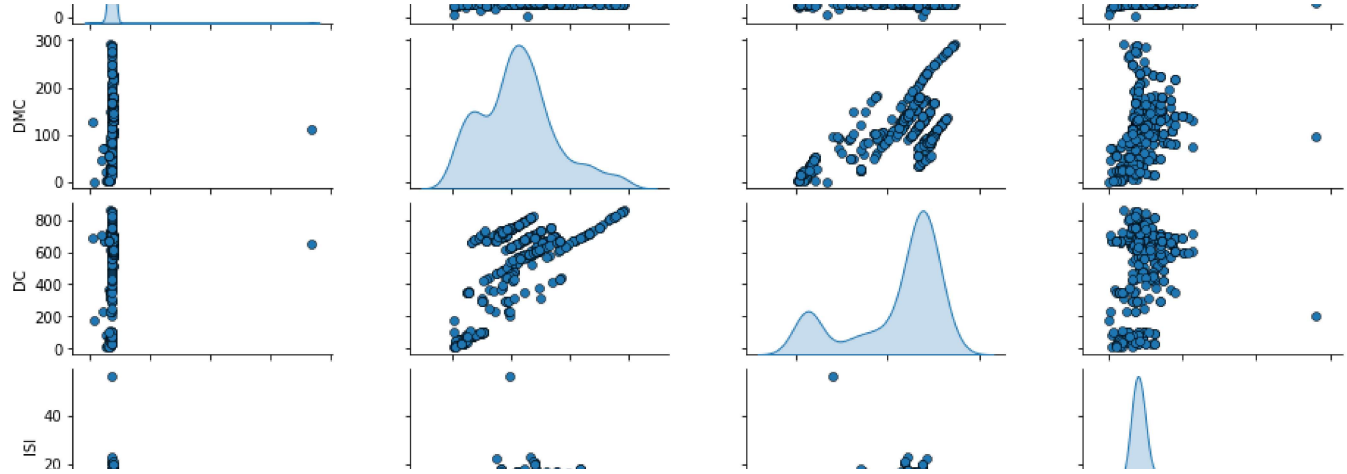
coord_X	1.000000	0.491639	-0.064310	-0.080476	-0.073207	-0.014774	-0.052665	0.0640
coord_Y	0.491639	1.000000	-0.016863	0.008177	-0.101999	-0.013351	-0.039904	0.0520
FFMC	-0.064310	-0.016863	1.000000	0.506550	0.257638	0.781080	0.591856	-0.3239
DMC	-0.080476	0.008177	0.506550	1.000000	0.556984	0.422985	0.501816	0.0314
DC	-0.073207	-0.101999	0.257638	0.556984	1.000000	0.100270	0.306887	0.0225
ISI	-0.014774	-0.013351	0.781080	0.422985	0.100270	1.000000	0.414878	-0.1817
temp	-0.052665	-0.039904	0.591856	0.501816	0.306887	0.414878	1.000000	-0.5215
RH	0.064074	0.052045	-0.323937	0.031465	0.022519	-0.181799	-0.521528	1.0000
wind	0.025365	-0.014473	-0.032402	-0.106328	-0.202119	0.138894	-0.178615	0.0408
rain	0.109331	0.079154	0.096754	0.120522	0.007885	0.117297	0.025753	0.1812
area	0.065161	0.053631	0.027664	0.068030	0.057345	0.012390	0.077977	-0.0263

Mining and Analytics:

I attempted to illustrate the parameters generated from the Fire Weather Index (FWI) System using pairplot.

```
cols = ['FFMC', 'DMC', 'DC', 'ISI']
pp = sns.pairplot(df_forest[cols], height=1.8, aspect=1.8,
                  plot_kws=dict(edgecolor="k", linewidth=0.5),
                  diag_kind="kde", diag_kws=dict(shade=True))

fig = pp.fig
fig.subplots_adjust(top=0.93, wspace=0.3)
t = fig.suptitle('Fire Weather Index (FWI) System PairPlot', fontsize=14)
```





Evaluation:

Not applicable.

Results:

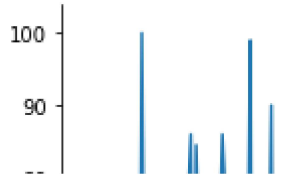
An independent variable is one whose change has no effect on the other variables in the experiment. The dependent variable, on the other hand, is what is being examined or measured in the experiment. As a result, the dependent variable is reliant on the independent variable.

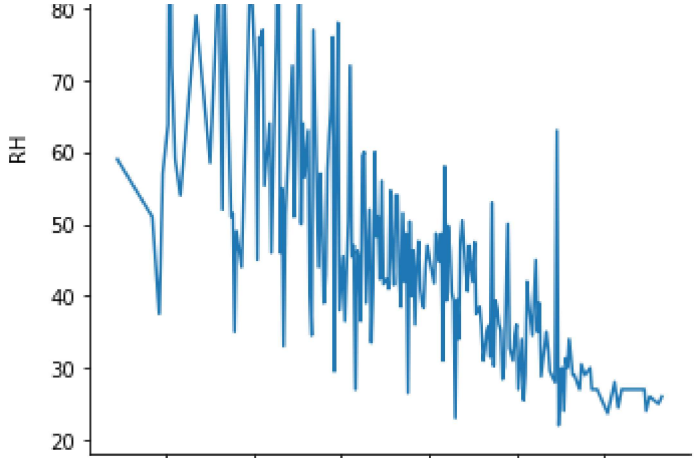
The customer may wish to examine the following variables as predictor (independent) and predicted (dependent) variables in this dataset.

Predictor (independent variables) - coord X, coord Y, month, day, temperature, and area

Predicted (dependent variables are dependent on "Temp") - RH, Wind, Rain, FFMC, DMC, DC, ISI, RH, DMC, DMC, DMC, DMC, DMC, DMC, DMC, DMC

```
# Example of how Relative Humidity is increasing with Temperature

sns.relplot(x="temp", y="RH", ci=None, kind="line", data=df_forest);
```





References:

CS5402 Lecture Notes - Having Enough Data.pptx

https://www.geeksforgeeks.org/use-pandas-to-calculate-statistics-in-python/

https://www.geeksforgeeks.org/python-pandas-dataframe-describe-method/

https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/nominal-ordinal-interval-ratio/

https://towardsdatascience.com/the-art-of-effective-visualization-of-multi-dimensional-data-6c7202990c57

GIT LINK:
https://git-classes.mst.edu/vmf8b/cs5402-programming-assignment-02