Name: Nishit Patel napn9w@umsystem.edu

Course: CS 5402

Assignment: Programming Assignment 02

Date: 2022-06-28

```
# 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 twoparts: 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 identival to the same attribute in the test set. when choosing three attributes in the training aet are identical to the samw attribute in the test set. when choosing three characteristics one 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 Decription:

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

coord_Y - y-axis spatial corrdinate 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 form the fire weather index (FWI) system

DMC - Duff Moistrue code from the Fire Weather Index (FXI) System

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

ISI - Inital Spread Index from the Fire Weather Index (FWI) System

temp - tempersature in Celsius degrees

RH - relative humidity in %

wind - wind speed in km/h

rain - outisde 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 spearated 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("/content/drive/MyDrive/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 dataypes of each colum
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 dataype 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['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

```
# groundby 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

```
# Groundby multiple categories to find count
df_forest.groupby(['day','coord_X','coord_Y','FFMC','DMC','DC','ISI','RH','month','temp','
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f797b667bd0>
```

Example:

return the burnt area in hectares

```
from pandas.core.frame import DataFrame
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
day          7
FFMC       108
DMC        215
DC         219
```

```
ISI        119
temp       192
RH          75
wind        21
rain         7
area       251
dtype: int64
```

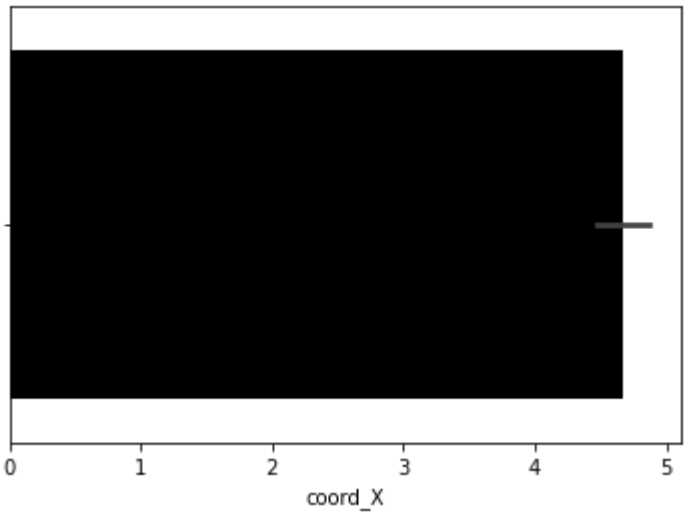
1. Determine whether the data is Nominal, Ordinal, Interval, or Ratio:

According to my understanding, I've integrated the qualities Nominal, Ordinal, Interval and Ration data from the given dataset.

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

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f797ad66d10>
```

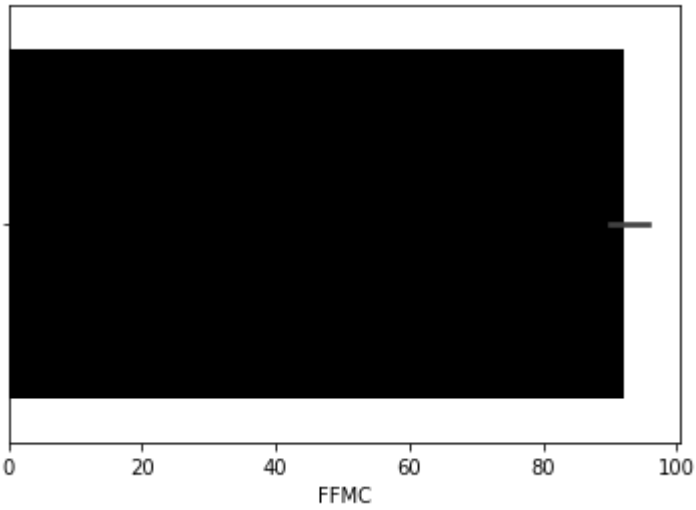



```
#coord_Y
sns.barplot(x='coord_Y', data=df_forest, color='black')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f797ab21e10>
```

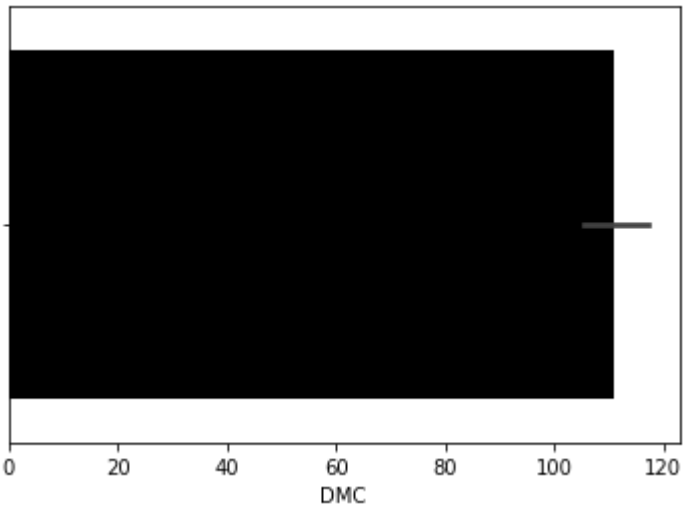
```
#FFMC
sns.barplot(x='FFMC', data=df_forest, color='black')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f797a597290>
```




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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f797a597990>
```

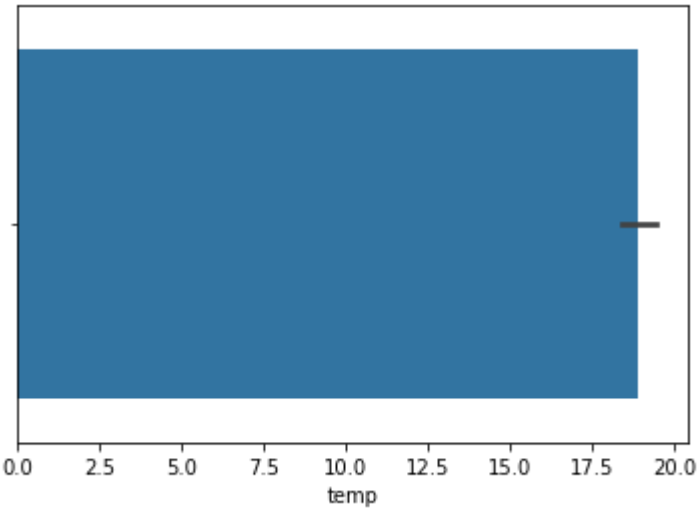



```
#DC
sns.barplot(x='DC', data=df_forest, color='black')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f797a4dc110>
```

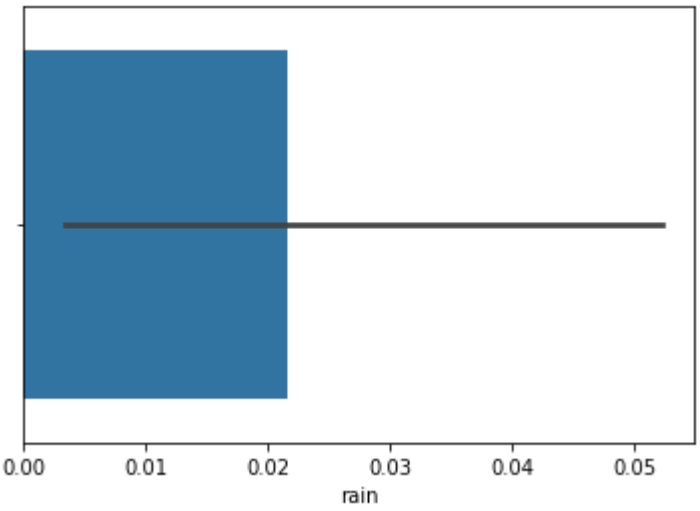
```
#TEMP
sns.barplot(x=df_forest["temp"],)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f797a433250>
```



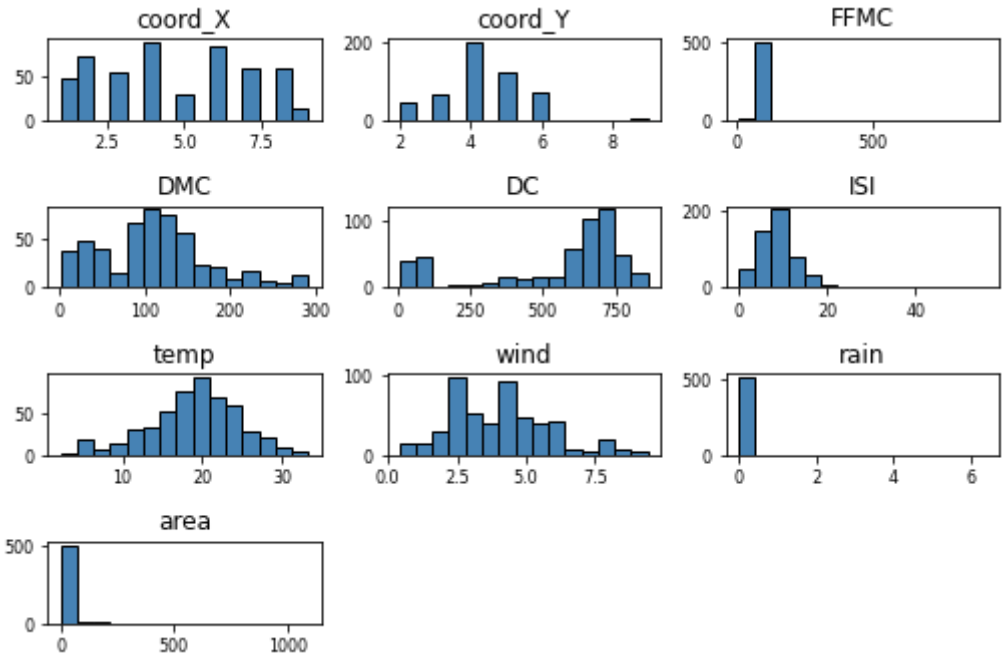

```
#rain
sns.barplot(x=df_forest["rain"],)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f797a410150>
```



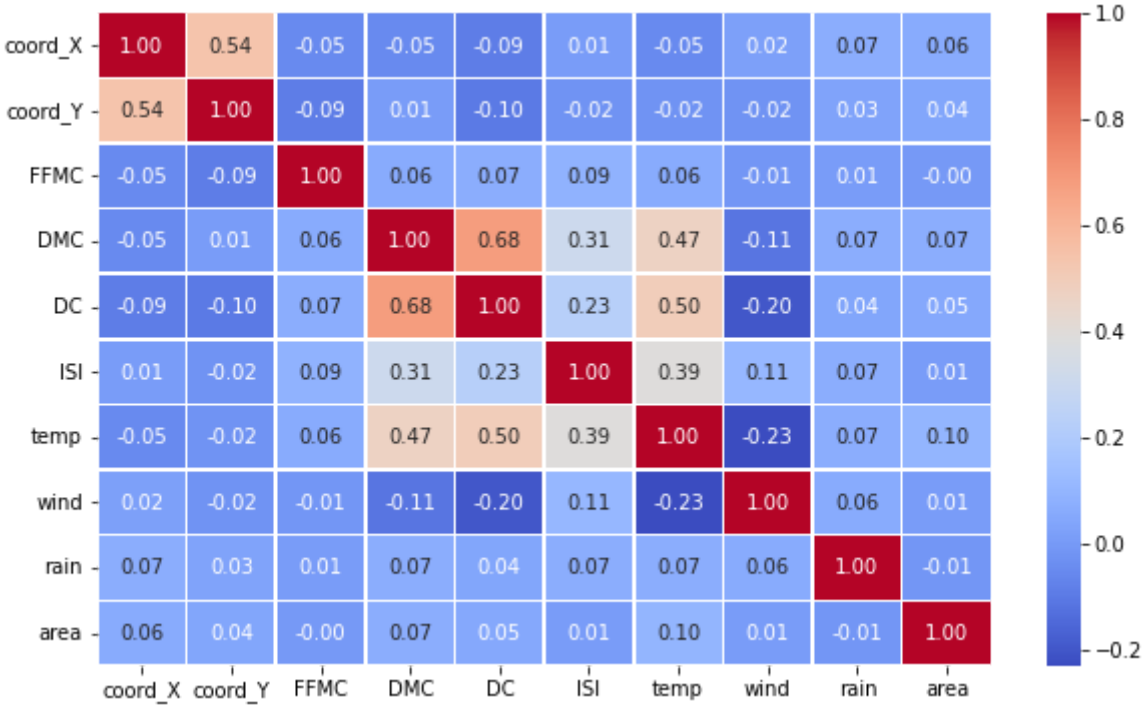

```
sns.catplot(x="temp", y="area",hue="wind", col="temp", data=df_forest, kind="bar", height=

#histogram plot for various attributes
```

```
df_forest.hist(bins=15, color='steelblue', edgecolor='black', linewidth=1.0, xlabelsize=8,
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",linewidth=0.5)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7971c27b10>
```



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

```
#To determine the value range fro each Attribute
df_forest.coord_X.unique()

#df_forest.nunique()
```

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

```
# To detemine 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. Detemine if there are any data values that we maybe concerned about and state why they are of concoern. if/when concerns are discovered, suggest what can be done to address those concers.

```
#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        True
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 dataset into training dataset and test dataset.

80% train data 20% test data

```
# define the values contained in the array corresponf to the numeric paramenters start, st
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

x_test
```

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

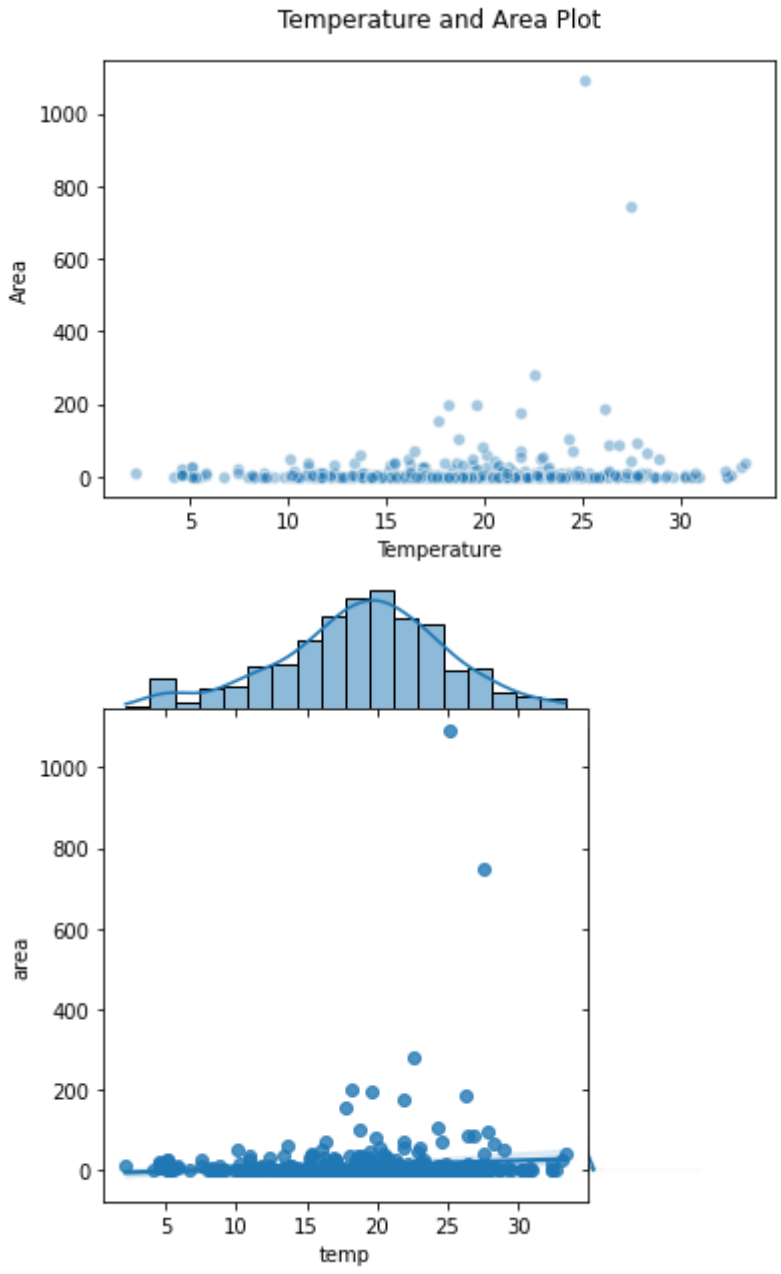
```
x_train
```

```
array([[8, 9],
       [4, 5],
       [0, 1],
       [6, 7]])
```

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=0.5)
plt.scatter(df_forest['temp'],df_forest['area'],alpha=0.4, edgecolor='w')
plt.xlabel('Temperature')
plt.ylabel('Area')
plt.title('Temperature and Area Plot', y=1.05)
```

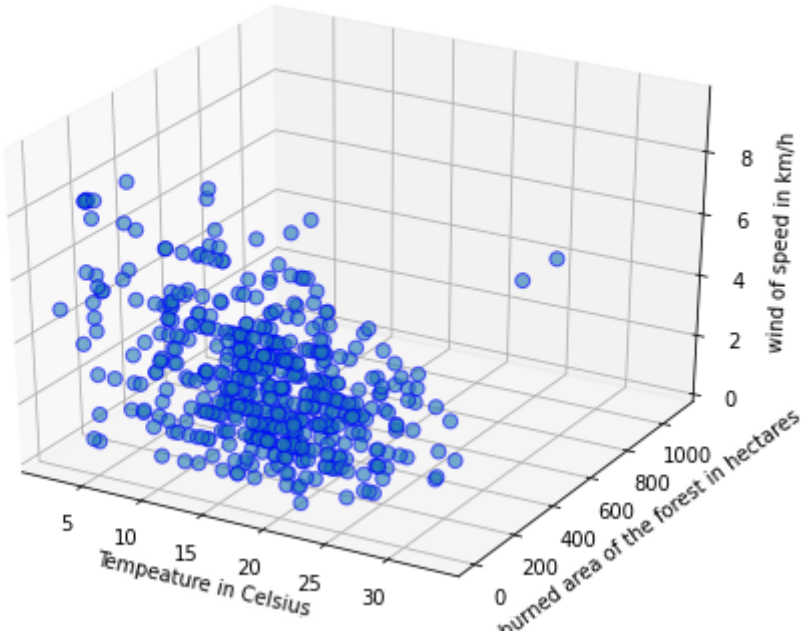
```
#joint Plot
jp = sns.jointplot(x='temp', y='area', data=df_forest, kind = 'reg', space = 0, height=5,
```



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

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

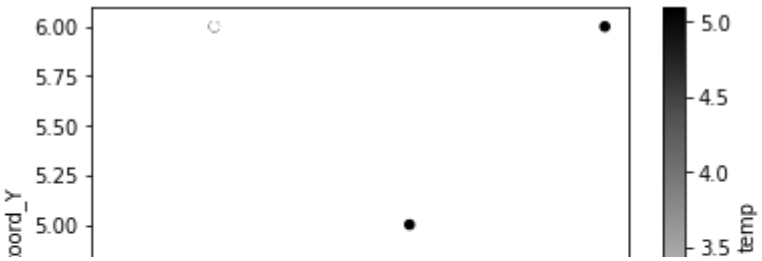


Exploratory Data Analysis:

```
# summerizing the data excluding the Nan values
df_forest.describe()
```

	coord_X	coord_Y	FFMC	DMC	DC	ISI	te
count	515.000000	515.000000	515.000000	515.000000	515.000000	515.000000	515.0000
mean	4.679612	4.302913	92.101165	111.001553	548.591845	9.035728	18.8959
std	2.311423	1.227716	37.182684	64.055503	247.560609	4.561636	5.8159
min	1.000000	2.000000	9.900000	1.100000	7.900000	0.000000	2.2000
25%	3.000000	4.000000	90.200000	69.150000	439.300000	6.500000	15.5500
50%	4.000000	4.000000	91.600000	108.300000	664.200000	8.400000	19.3000
75%	7.000000	5.000000	92.900000	142.400000	713.900000	10.900000	22.8000
max	9.000000	9.000000	921.000000	291.300000	860.600000	56.100000	33.3000

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



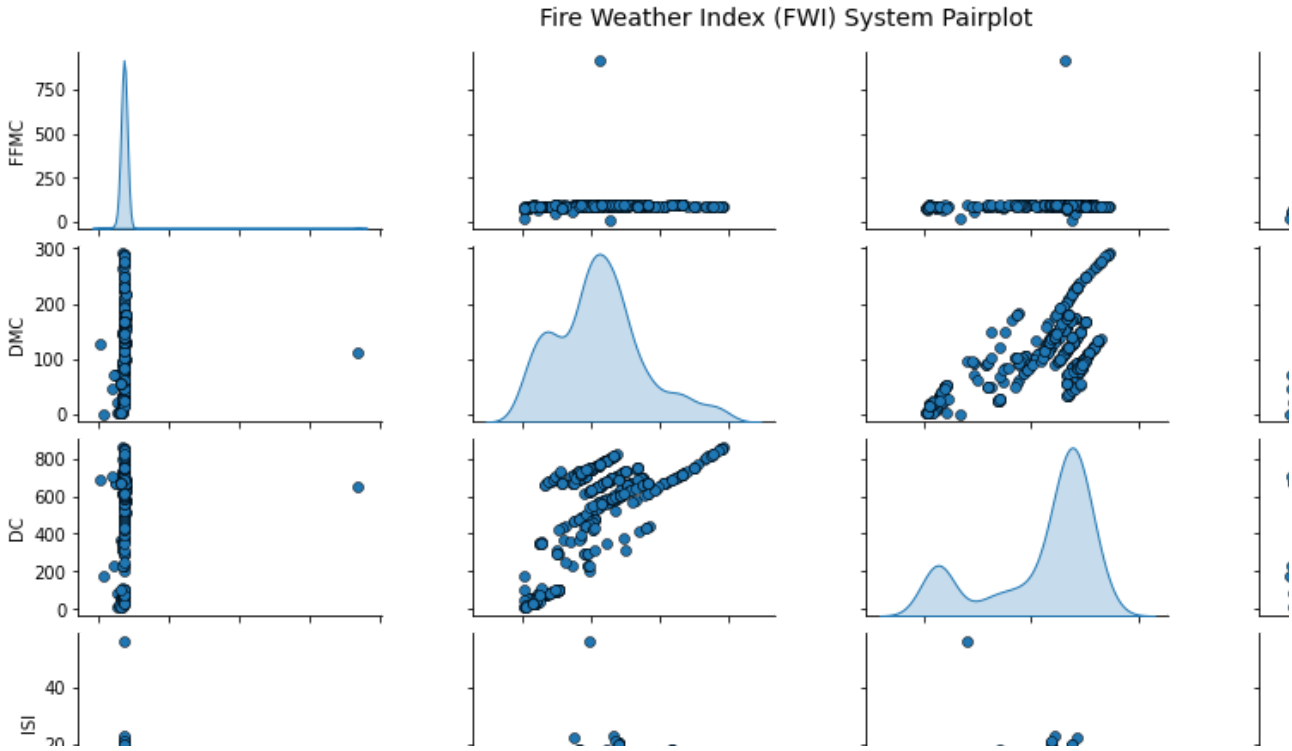

```
# Correlation Matrix between attributes using spearman menthod
corr_matrix = df_forest.corr(method='spearman')
corr_matrix
```

	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	0.0
coord_Y	0.491639	1.000000	-0.016863	0.008177	-0.101999	-0.013351	-0.039904	-0.0
FFMC	-0.064310	-0.016863	1.000000	0.506550	0.257638	0.781080	0.591856	-0.0
DMC	-0.080476	0.008177	0.506550	1.000000	0.556984	0.422985	0.501816	-0.1
DC	-0.073207	-0.101999	0.257638	0.556984	1.000000	0.100270	0.306887	-0.2
ISI	-0.014774	-0.013351	0.781080	0.422985	0.100270	1.000000	0.414878	0.1
temp	-0.052665	-0.039904	0.591856	0.501816	0.306887	0.414878	1.000000	-0.1
wind	0.025365	-0.014473	-0.032402	-0.106328	-0.202119	0.138894	-0.178615	1.0
rain	0.109331	0.079154	0.096754	0.120522	0.007885	0.117297	0.025753	0.1
area	0.065161	0.053631	0.027664	0.068030	0.057345	0.012390	0.077977	0.0

Mining and Analytics:

I attemted 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", l

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



Evaluation:

N/A

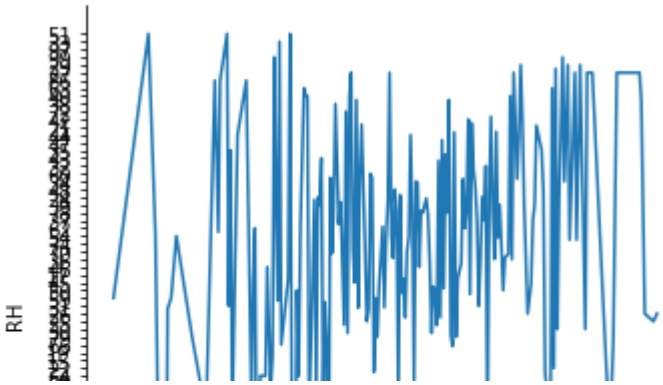
Results:

An independent variable is one whose change has no effect on the other varibales in the expriment. 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) ans predicted (depentent) variables in the dataset. Predictor(independent variable)-coord X, corrd Y, month, day, temperature, and area

predicted (dependent variables are depentent 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);
```



Refernces:

CS5402 Lecture Notes - Having Enough Data.pptx

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

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