datamaing_prog2

February 18, 2022

Name:

Mohith Krishna Behata

Email:

mbm2b@mst.edu

Course:

CS 5402

Assignment:

Programming assignment 2

Date:

2022-02-17

GitHublink:

https://git-classes.mst.edu/mbm2b/dataminingprog2

[1]:
```
#to remove unneccesary future warnings for seaborn
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
# Imported for data management (dataframes)
import pandas as pd
import numpy as np
import seaborn as sns
import sklearn as sk
import matplotlib.pyplot as plt
```

0.1 Concept Description:

Despite their ecological benefits, wooded area fires can threaten human lives and property. Understanding when they take place and what motives them is vital for managing them. The center of attention will be on visualizing it. There will be exploratory analyses on the facts to better recognize it and any relationships that may be existing in it. Problems with data collectionmean missing or inconsistent data, Usually due to a malfunctioning equipment or a lack of interest by users to provide the data, or simply problems associated with aggregating multiple data sources. It is better if we understand the statistics of each attribute and the relationship among the features which is extremely useful in several prediction and forecasting scenarios and in also deciding the appropriate visualizations and ML models. In this assignment, the goal is to understand these statistics, and relations within the dataset.

0.2 Data Collection:

The forest fire data set was provided by our Instructor Mr.Peery Koob However each attribute value present in the forest fire dataset might be collected with help of various types of machines and sensors and stored in forest fire csv file.

0.3 Example Description:

coord_X x-axis spatial coordinate within a topographical map of the area of interest: 1 to 9.This is a Intreval Data type. coord_Y y-axis spatial coordinate within a topographical map of the area of interest: 2 to 9.This is a Intreval Data type. month month of the year: 'jan' to 'dec'in which the forest fire happened.This is a Nominal data type. day day of the week: 'mon' to 'sun' in which the forest fire happened.This is a Nominal data type.. FFMC Fine Fuel Moisture Code from the Fire Weather Index (FWI) System: 18.7 to 96.20.This is a Intrevel Data type. DMC Duff Moisture Code from the Fire Weather Index (FWI) System: 1.1 to 291.3.This is a Intrevel Data type. DC Draught Code from the Fire Weather Index (FWI) System : 7.9 to 860.6.This is a Intrevel Data type. ISI Initial Spread Index from the Fire Weather Index (FWI) System : 0.0 to 56.10.This is a Intrevel Data type. temp Temperature in Celsius degrees: 2.2 to 33.30.This is a Intreval Data type. RH Relative humidity in %: 15.0 to 100.This is a Ratio Data type. wind Wind speed in km/h: 0.40 to 9.40.This is a intreval Data type. rain Outside rain in mm/m2 : 0.0 to 6.4.This is a Ratio Data type. area The burned area of the forest in hectares: 0.00 to 1090.84.This is a Ratio Data type.

Attributes level of measurement - coord_X - interval datatype. - coord_Y - interval datatype. - month - nominal datatype. - day - nominal datatype. - FFMC - interval datatype. - DMC - interval datatype. - DC - interval datatype. - ISI - interval datatype. - temp - interval datatype. - RH - ratio datatype. - wind - interval datatype. - rain - ratio datatype. - area - ratio datatype.

0.4 Data Import and Wrangling:

[2]:
```
# load dataset
dataframe = pd.read_csv("forestfires.csv")
```

[3]:
```
#Understand the data head()returns the First 5 rows of the dataset
dataframe.head()
```

[3]:
```
   coord_X  coord_Y month  day  FFMC   DMC     DC  ISI  temp  RH  wind  rain  \
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  18.0  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  9.0   8.3  97   4.0   0.2
4        8        6   mar  sun  89.3  51.3  102.2  9.6  11.4  99   1.8   0.0

   area
0   0.0
1   0.0
```

```
2    0.0
3    0.0
4    0.0
```

[4]:
```python
#Understand the data tail() returns the last 5 rows of the dataset
dataframe.tail()
```

[4]:
```
     coord_X  coord_Y month  day  FFMC    DMC     DC   ISI  temp  RH  wind  \
512        4        3   aug  sun  81.6   56.7  665.6   1.9  27.8  32   2.7
513        2        4   aug  sun  81.6   56.7  665.6   1.9  21.9  71   5.8
514        7        4   aug  sun  81.6   56.7  665.6   1.9  21.2  70   6.7
515        1        4   aug  sat  94.4  146.0  614.7  11.3  25.6  42   4.0
516        6        3   nov  tue  79.5    3.0  106.7   1.1  11.8  31   4.5

     rain   area
512   0.0   6.44
513   0.0  54.29
514   0.0  11.16
515   0.0   0.00
516   0.0   0.00
```

[5]:
```python
#for understanding the shape of dataset
print("Shape:", dataframe.shape)
```

```
Shape: (517, 13)
```

By this we understood there are 13 columns in this dataset and 517 rows in the dataset.

[6]:
```python
dataframe.corr()
```

[6]:
```
          coord_X   coord_Y      FFMC       DMC        DC       ISI      temp  \
coord_X  1.000000  0.539548 -0.050014 -0.048384 -0.085916  0.006210 -0.052299
coord_Y  0.539548  1.000000 -0.093677  0.007782 -0.101178 -0.024488 -0.023613
FFMC    -0.050014 -0.093677  1.000000  0.056102  0.065324  0.086470  0.062837
DMC     -0.048384  0.007782  0.056102  1.000000  0.682192  0.305128  0.469095
DC      -0.085916 -0.101178  0.065324  0.682192  1.000000  0.229154  0.496167
ISI      0.006210 -0.024488  0.086470  0.305128  0.229154  1.000000  0.393683
temp    -0.052299 -0.023613  0.062837  0.469095  0.496167  0.393683  1.000000
RH       0.085223  0.062221 -0.048512  0.073795 -0.039192 -0.132517 -0.529388
wind     0.018798 -0.020341 -0.012160 -0.105342 -0.203466  0.106826 -0.226362
rain     0.065387  0.033234  0.005575  0.074790  0.035861  0.067668  0.069429
area     0.063385  0.044873 -0.000806  0.072994  0.049383  0.008258  0.098703

               RH      wind      rain      area
coord_X  0.085223  0.018798  0.065387  0.063385
coord_Y  0.062221 -0.020341  0.033234  0.044873
FFMC    -0.048512 -0.012160  0.005575 -0.000806
DMC      0.073795 -0.105342  0.074790  0.072994
```

```
DC      -0.039192 -0.203466  0.035861  0.049383
ISI     -0.132517  0.106826  0.067668  0.008258
temp    -0.529388 -0.226362  0.069429  0.098703
RH       1.000000  0.069410  0.099751 -0.075519
wind     0.069410  1.000000  0.061119  0.012317
rain     0.099751  0.061119  1.000000 -0.007366
area    -0.075519  0.012317 -0.007366  1.000000
```

Partitioning dataset and Selecting attributes A random sample is preferred for obtaining a good mixture of data and ensuring a good distribution of the dataset. So I use sklearn's random sampling and splitting function. The data was divided 60:40, with 60% of the data serving as a training set and 40% as testing data. In addition, a 'y' value is required for the train test split. According to the correlation matrix ,the attribute 'rain' has a positive correlation value with almost all of the attributes except area ; thus, I will consider this to be the Y value, and all of the other attributes to be the X values. The three attributes that I will be selecting are: 1. Day - Nominal data type 2. DC - Interval data type 3. temp - Attribute with missing value

[7]:
```
from sklearn.model_selection import train_test_split
X = dataframe[['day','DC','temp']]
Y = dataframe[['rain']]
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.40,␣
 ↪random_state=42)
```

Values for each attribute Determining the possible values or range of the values for each attribute

For all the numeric data (interval and ratio) the minimum and maximum values would be provided as those values fall within the range.

For ordinal values I provide a list of unique values with respect to the dataset, and extrapolate to give all the actual possible values in a situation where the data set is more extensive.

[8]:
```
numeric=['coord_X','coord_Y','FFMC','DMC','DC','ISI','RH','temp','rain','wind','area']
nominal=['month','day']

for a in numeric:
    print(a, 'has values that fall between',dataframe[a].min(),'and',␣
 ↪dataframe[a].max())

for a in nominal:
    print('The possible values for ',a,' are: ',dataframe[a].unique())
```

```
coord_X has values that fall between 1 and 9
coord_Y has values that fall between 2 and 9
FFMC has values that fall between 9.9 and 921.0
DMC has values that fall between 1.1 and 291.3
DC has values that fall between 7.9 and 860.6
ISI has values that fall between 0.0 and 56.1
RH has values that fall between 15 and 100
temp has values that fall between 2.2 and 33.3
```

```
rain has values that fall between 0.0 and 6.4
wind has values that fall between 0.4 and 9.4
area has values that fall between 0.0 and 1090.84
The possible values for  month  are:  ['mar' 'oct' 'aug' 'sep' 'apr' 'jun' 'jul'
'feb' 'jan' 'dec' 'may' 'nov']
The possible values for  day  are:  ['fri' 'tue' 'sat' 'sun' 'mon' 'wed' 'thu']
```

month values in order are :[jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec] day values in order are :[mon, tue, wed, thu, fri, sat, sun]

[9]:
```
#Now we will check for unique values for each attribute present in the forest␣
↪fire dataset
dataframe.nunique()
```

[9]:
```
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
```

Determining attribute with concerns

[21]:
```
column = dataframe.columns
for a in column:
    if dataframe[a].isnull().any().sum():
        print(a,'has missing values')
    else:
        print(a,'has no missing values')
```

```
coord_X has no missing values
coord_Y has no missing values
month has no missing values
day has no missing values
FFMC has no missing values
DMC has no missing values
DC has no missing values
ISI has no missing values
temp has missing values
RH has no missing values
wind has no missing values
```

```
rain has no missing values
area has no missing values
```

"temp" attribute is the only attribute with missing value.

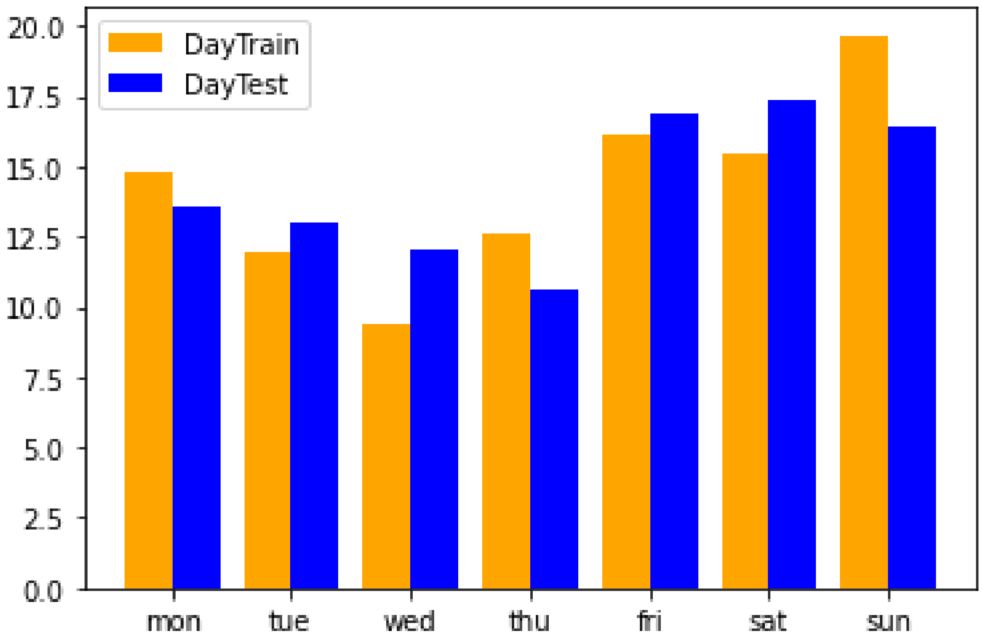
These missing values are of a concern in future analysis, one can either disregard/delete these rows or estimate the missing values with different methods available.

In this assignment, we need to use an attribute with missing values so we are not dealing with resolving the missing value.

Attribute statistics and visualization for both training and testing data For attribute day i would be showing the visualization the same attribute in training and testing data using bar chart. For attributes DC and Temp i would be showing the stastistics and visualize the same attribute in training and testing data using box plots. If the plots have a lot of variation this indicates that sampling has some issues

Day attribute Comparing train and test of Nominal attribute.

[10]:
```
daytrain = X_train.iloc[:,0]
daytest = X_test.iloc[:,0]
days = ['mon','tue','wed','thu','fri','sat','sun']
dtrainf = daytrain.value_counts()
dtestf = daytest.value_counts()
vals = []
for d in days:
    x = []
    x.append(dtrainf[d]/len(X_train)*100)
    x.append(dtestf[d]/len(X_test)*100)
    vals.append(x)
vals = np.array(vals)
x_axis = np.arange(len(vals))
plt.bar(x_axis-0.2, vals[:,0],width=0.4,color='orange',label='DayTrain')
plt.bar(x_axis+0.2, vals[:,1],width=0.4,color='blue',label="DayTest")
plt.xticks(x_axis, days)
plt.legend()
plt.show()
```

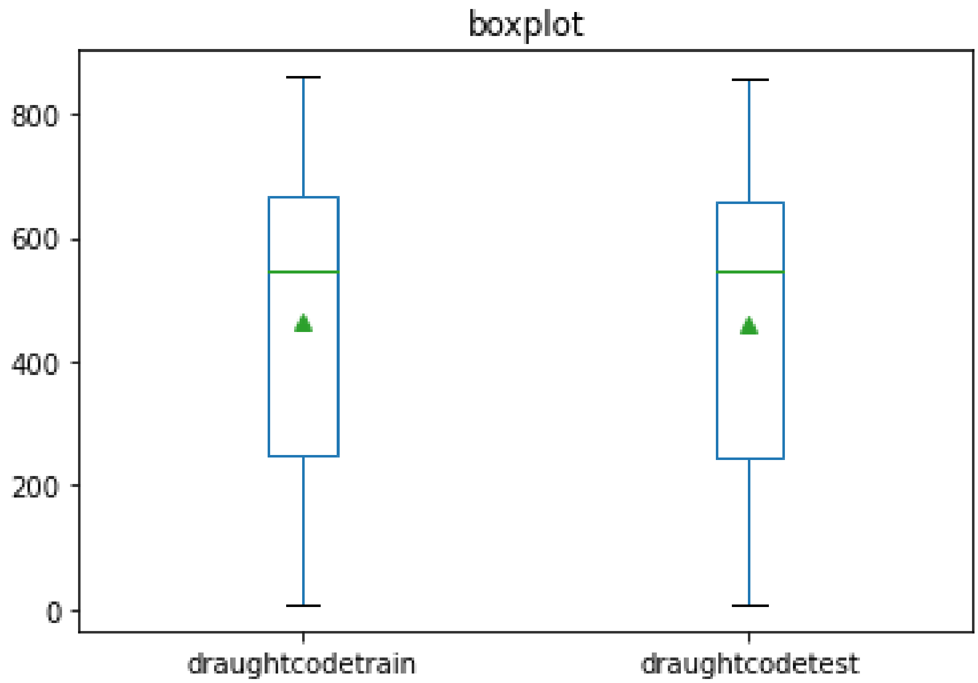


I plotted the percentage of occurrences of all days in the graph because the frequency alone would not have provided adequate statistics. The plots show that the distribution of each day in train data and test data is very close to each other.

DC attribute Compare train and test of intreval attribute.

[11]:
```
draughtcodetrain = X_train.iloc[:,1]
draughtcodetest= X_test.iloc[:,1]
dctrain = [np.min(draughtcodetrain),np.max(draughtcodetrain),np.
 ↪std(draughtcodetrain),np.mean(draughtcodetrain),np.median(draughtcodetrain)]
dctest =[np.min(draughtcodetest),np.max(draughtcodetest),np.
 ↪std(draughtcodetest),np.mean(draughtcodetest),np.median(draughtcodetest)]
print("minimum, maximum, standard deviation, mean, median")
print(dctrain,'for test')
print(dctest,'for train')
DF = pd.DataFrame({'draughtcodetrain':dctrain, 'draughtcodetest':dctest})
ax = DF[['draughtcodetrain', 'draughtcodetest']].plot(kind='box',␣
 ↪title='boxplot', showmeans=True)
plt.show()
```

```
minimum, maximum, standard deviation, mean, median
[7.9, 860.6, 251.08020426649668, 548.6564516129037, 665.45] for test
[9.3, 855.3, 242.86749470372428, 546.8671497584547, 658.2] for train
```

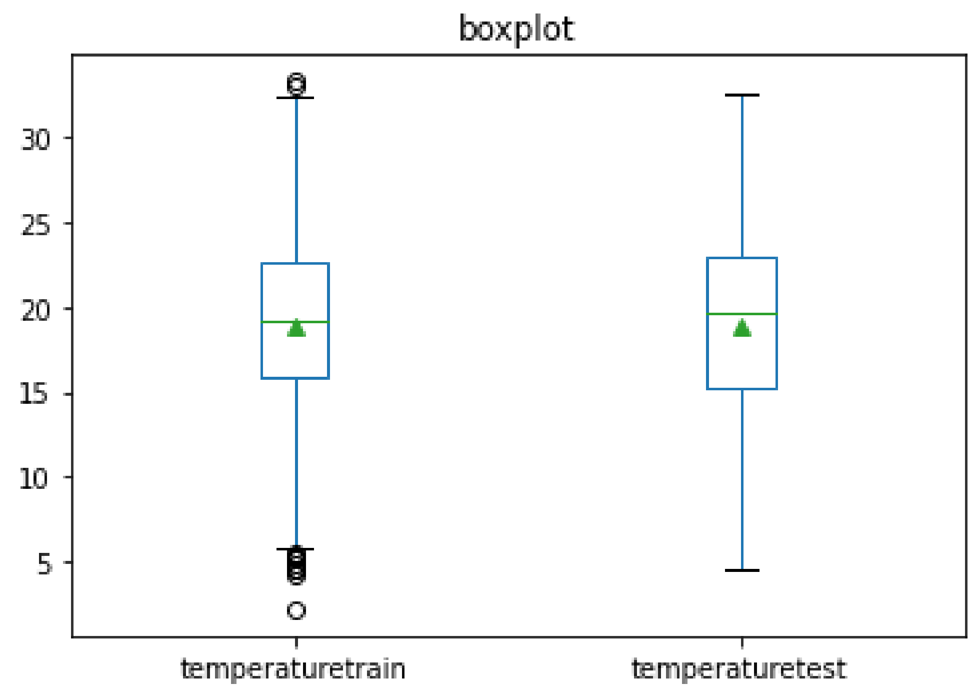


Considering the boxplot and statistics, we can see that the DC attribute in the train and the test data has a very similar distribution, primarily due to the means and the skewness of the data.

Temperature Compare train and test of attribute issues.

[12]:
```
temperaturetrain = X_train.iloc[:,2]
temperaturetest= X_test.iloc[:,2]
temptrain = [np.min(temperaturetrain),np.max(temperaturetrain),np.
 ↪std(temperaturetrain),np.mean(temperaturetrain),np.median(temperaturetrain)]
temptest =[np.min(temperaturetest),np.max(temperaturetest),np.
 ↪std(temperaturetest),np.mean(temperaturetest),np.median(temperaturetest)]
print("minimum, maximum, standard deviation, mean, median")
print(temptrain,'for train')
print(temptest,'for test')
DF = pd.DataFrame({'temperaturetrain':temperaturetrain, 'temperaturetest':
 ↪temperaturetest})
ax = DF[['temperaturetrain', 'temperaturetest']].plot(kind='box',␣
 ↪title='boxplot', showmeans=True)
plt.show()
```

```
minimum, maximum, standard deviation, mean, median
[2.2, 33.3, 5.6611589342130095, 18.89093851132685, nan] for train
[4.6, 32.6, 6.02717455474738, 18.903398058252435, nan] for test
```



As we can see even for the temperature attribute, the distributions are very similar in the training and the testing dataset. Check the boxplot and note the close values for min, max, and mean for the training and testing data. However, due to missing values, there is no median.

0.5 Exploratory Data Analysis:

0.5.1 Finding relationships

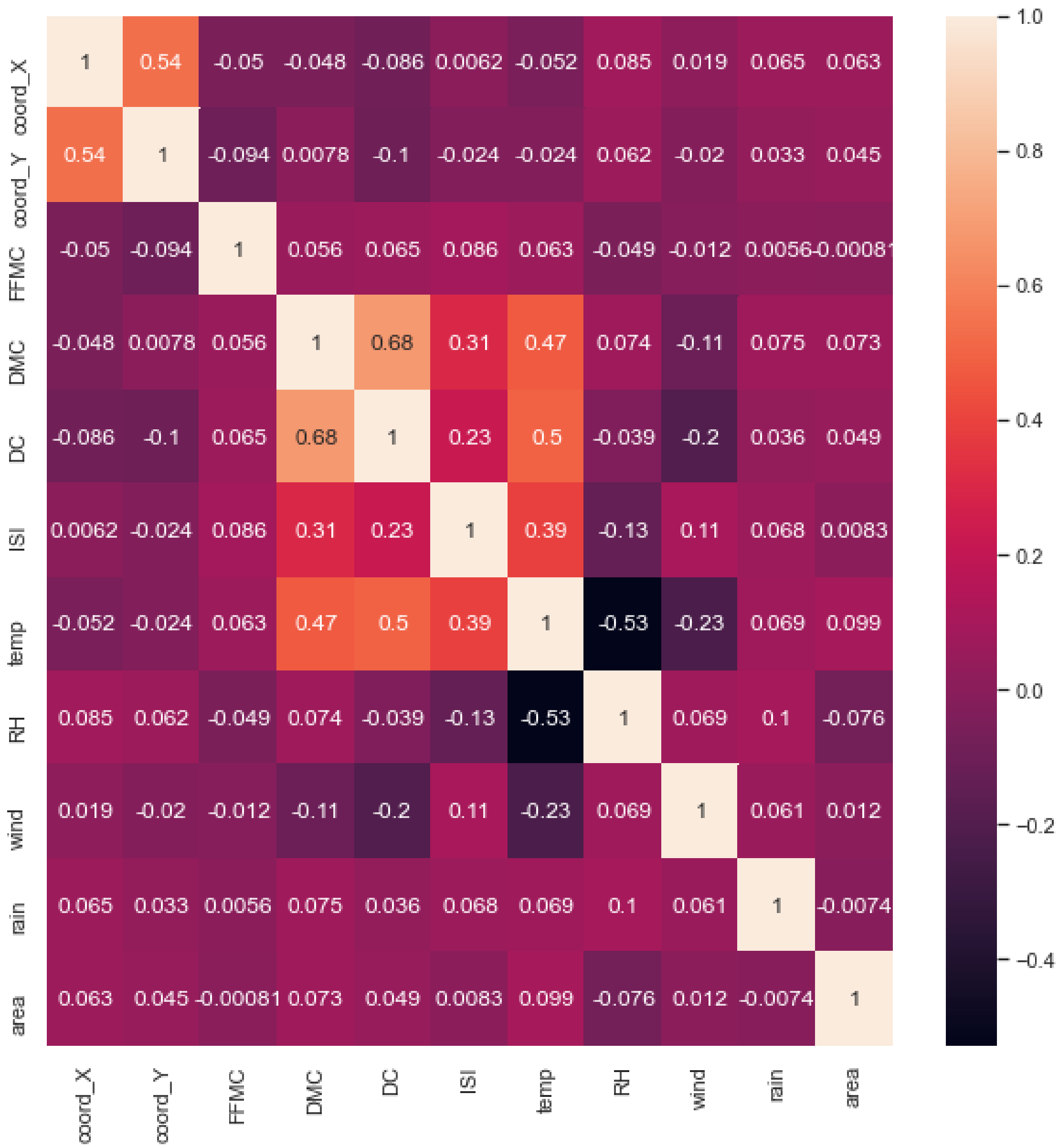
To identify the other possible relations we will use Pearson coefficient.To identify relations among different features within a dataset.

[13]:
```
##find Pearson correlation coefficient, which represents linear relationships␣
↪between two variables
corelation=dataframe.corr(method='pearson')
print(corelation)
```

```
          coord_X   coord_Y      FFMC       DMC        DC       ISI      temp  \
coord_X  1.000000  0.539548 -0.050014 -0.048384 -0.085916  0.006210 -0.052299
coord_Y  0.539548  1.000000 -0.093677  0.007782 -0.101178 -0.024488 -0.023613
FFMC    -0.050014 -0.093677  1.000000  0.056102  0.065324  0.086470  0.062837
DMC     -0.048384  0.007782  0.056102  1.000000  0.682192  0.305128  0.469095
DC      -0.085916 -0.101178  0.065324  0.682192  1.000000  0.229154  0.496167
ISI      0.006210 -0.024488  0.086470  0.305128  0.229154  1.000000  0.393683
temp    -0.052299 -0.023613  0.062837  0.469095  0.496167  0.393683  1.000000
RH       0.085223  0.062221 -0.048512  0.073795 -0.039192 -0.132517 -0.529388
wind     0.018798 -0.020341 -0.012160 -0.105342 -0.203466  0.106826 -0.226362
```

```
rain     0.065387  0.033234  0.005575  0.074790  0.035861  0.067668  0.069429
area     0.063385  0.044873 -0.000806  0.072994  0.049383  0.008258  0.098703

               RH      wind      rain      area
coord_X  0.085223  0.018798  0.065387  0.063385
coord_Y  0.062221 -0.020341  0.033234  0.044873
FFMC    -0.048512 -0.012160  0.005575 -0.000806
DMC      0.073795 -0.105342  0.074790  0.072994
DC      -0.039192 -0.203466  0.035861  0.049383
ISI     -0.132517  0.106826  0.067668  0.008258
temp    -0.529388 -0.226362  0.069429  0.098703
RH       1.000000  0.069410  0.099751 -0.075519
wind     0.069410  1.000000  0.061119  0.012317
rain     0.099751  0.061119  1.000000 -0.007366
area    -0.075519  0.012317 -0.007366  1.000000
```

[14]:
```
# Correlation Heatmap of the features in the dataset
plt.figure(figsize=(10,10))
sns.set(font_scale = 1)
sns.heatmap(dataframe.corr(),annot = True);
```

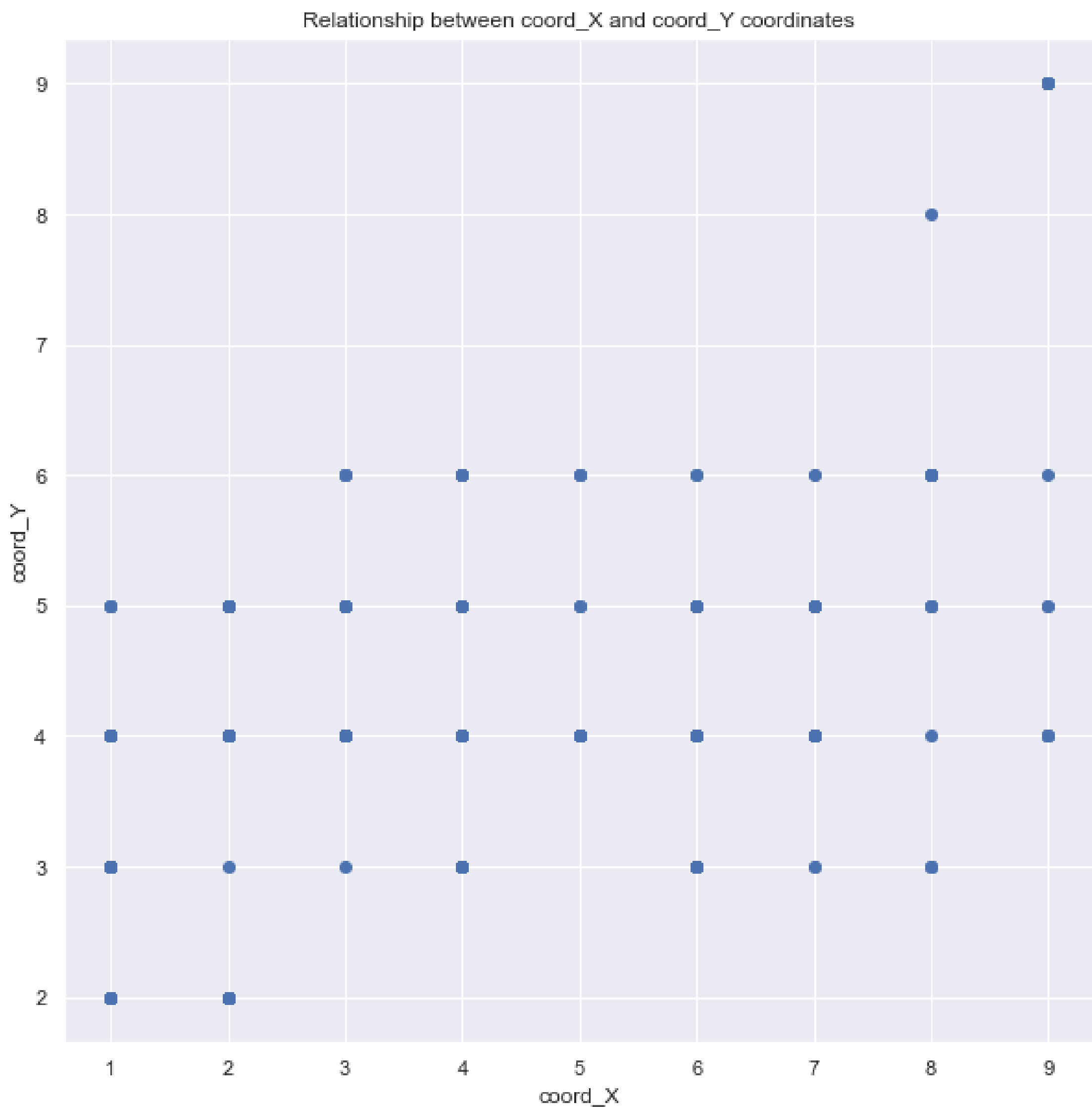


Visualizing other relationships From the above heat map we can identify relationship between attributes. The corelation values range between -1 and 1. I am considering only positive correlation values, values greater than 0.1 as a basis to identify and plot the relations. Following are the relationships based on the above assumptions. 1. Coord_X with Coord_Y. 2. DMC with DC, ISI, Temp. 3. DC with ISI and Temp. 4. ISI with Temp.

Relation between Coord_X with Coord_Y.

[15]:
```
plt.figure(figsize=(10,10))
x = np.array(dataframe['coord_X'])
y = np.array(dataframe['coord_Y'])
```

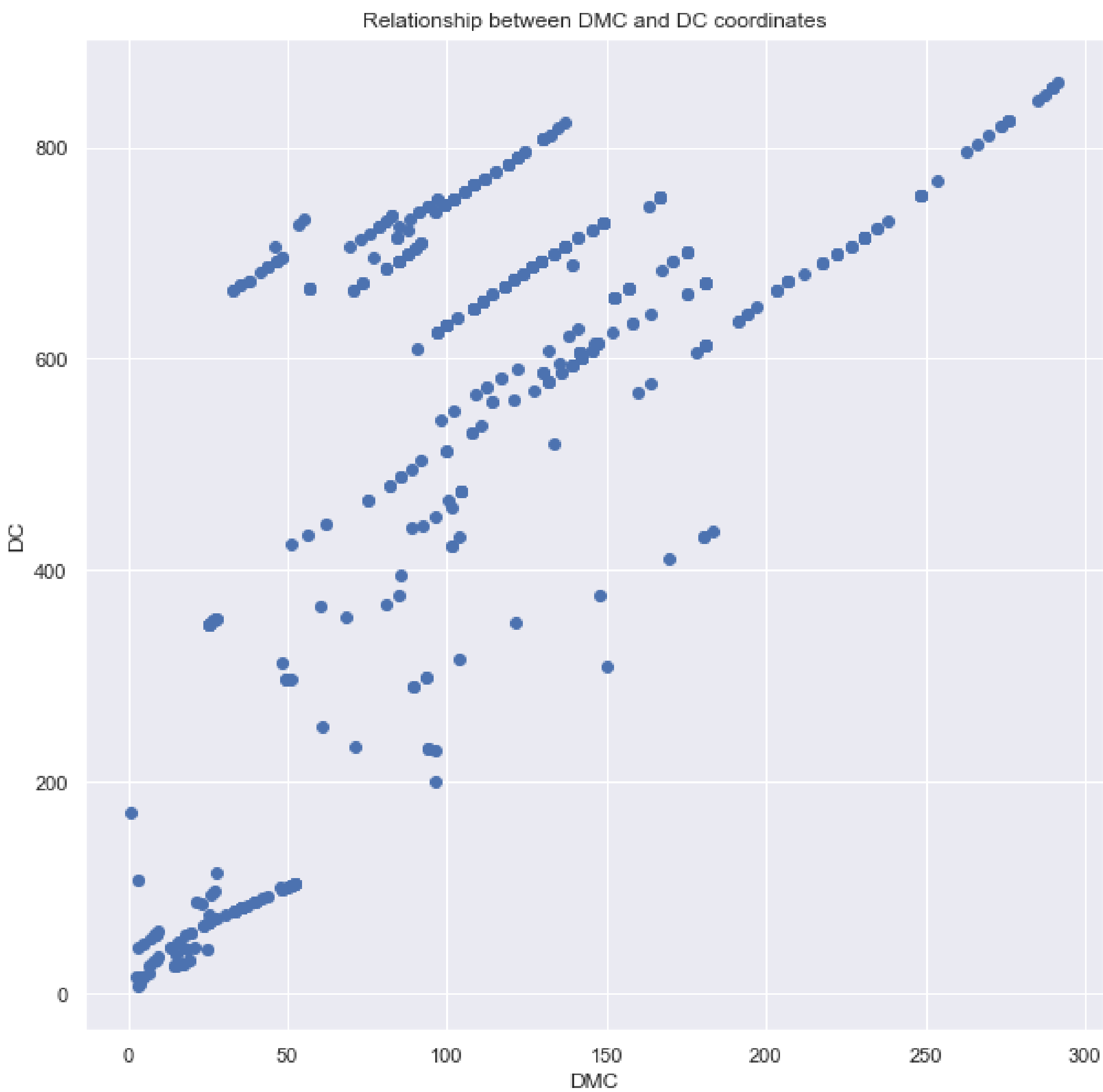
```
plt.title('Relationship between coord_X and coord_Y coordinates')
plt.xlabel('coord_X')
plt.ylabel('coord_Y')
plt.scatter(x, y)
plt.show()
```

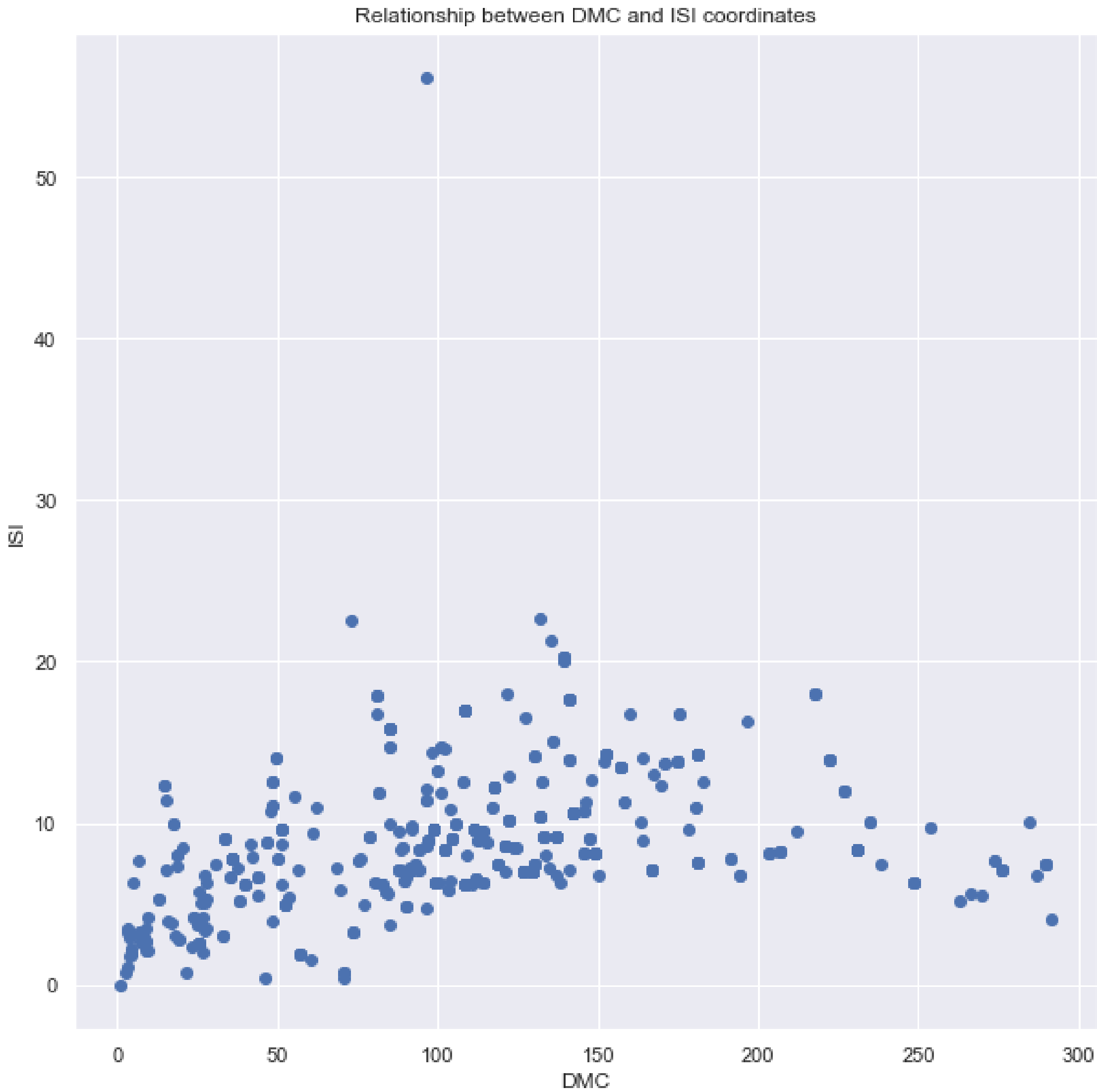


Relation between DMC with DC, ISI, Temp.

[16]:
```
plt.figure(figsize=(10,10))
x = np.array(dataframe['DMC'])
y = np.array(dataframe['DC'])
plt.title('Relationship between DMC and DC coordinates')
plt.xlabel('DMC')
```

```
plt.ylabel('DC')
plt.scatter(x, y)
plt.show()

plt.figure(figsize=(10,10))
x = np.array(dataframe['DMC'])
y = np.array(dataframe['ISI'])
plt.title('Relationship between DMC and ISI coordinates')
plt.xlabel('DMC')
plt.ylabel('ISI')
plt.scatter(x, y)
plt.show()

plt.figure(figsize=(10,10))
x = np.array(dataframe['DMC'])
y = np.array(dataframe['temp'])
plt.title('Relationship between DMC and temp coordinates')
plt.xlabel('DMC')
plt.ylabel('temp')
plt.scatter(x, y)
plt.show()
```


Relationship between DMC and DC coordinates
DC
800
600
400
200
0
DMC
0
50
100
150
200
250
300

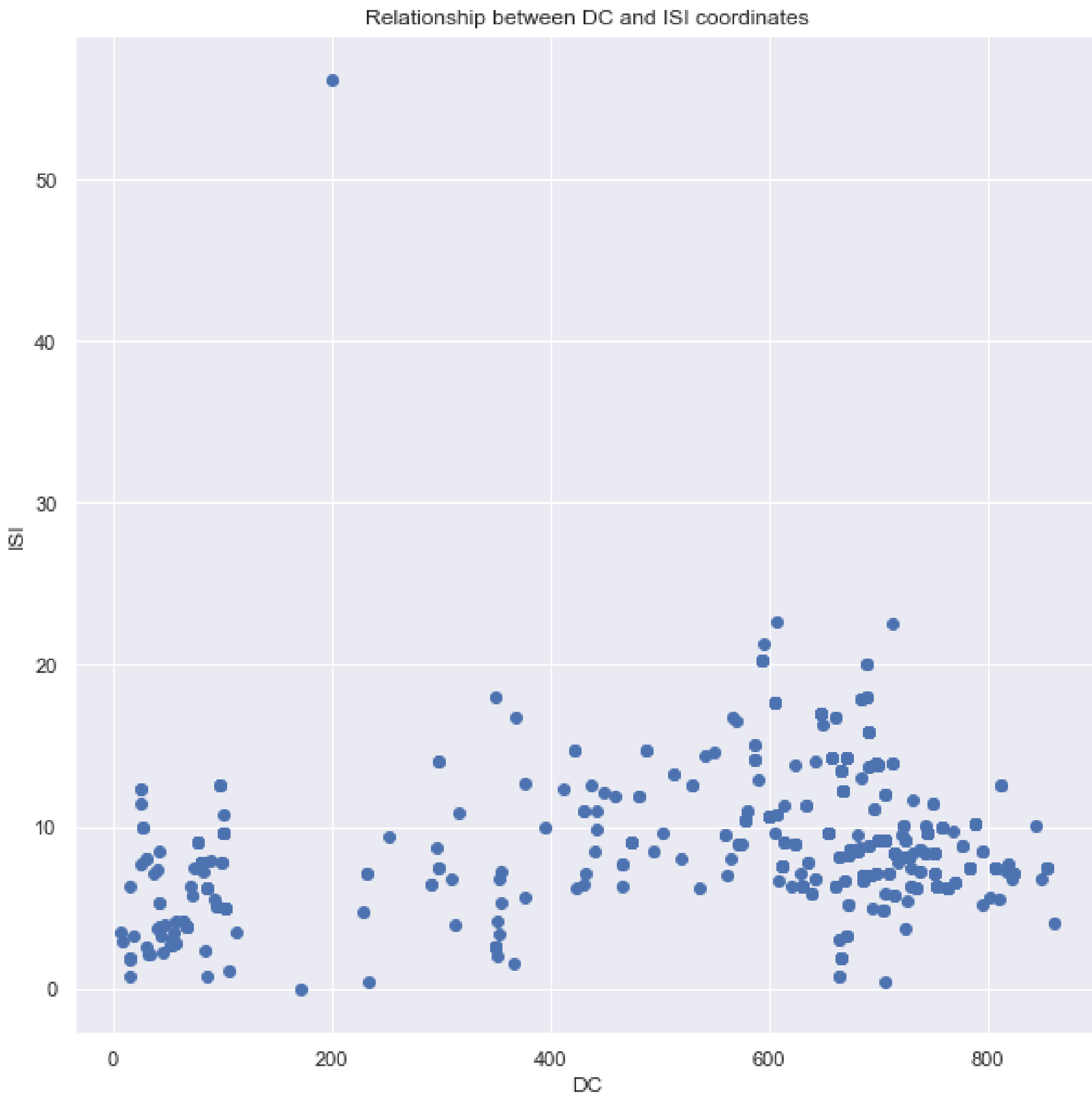


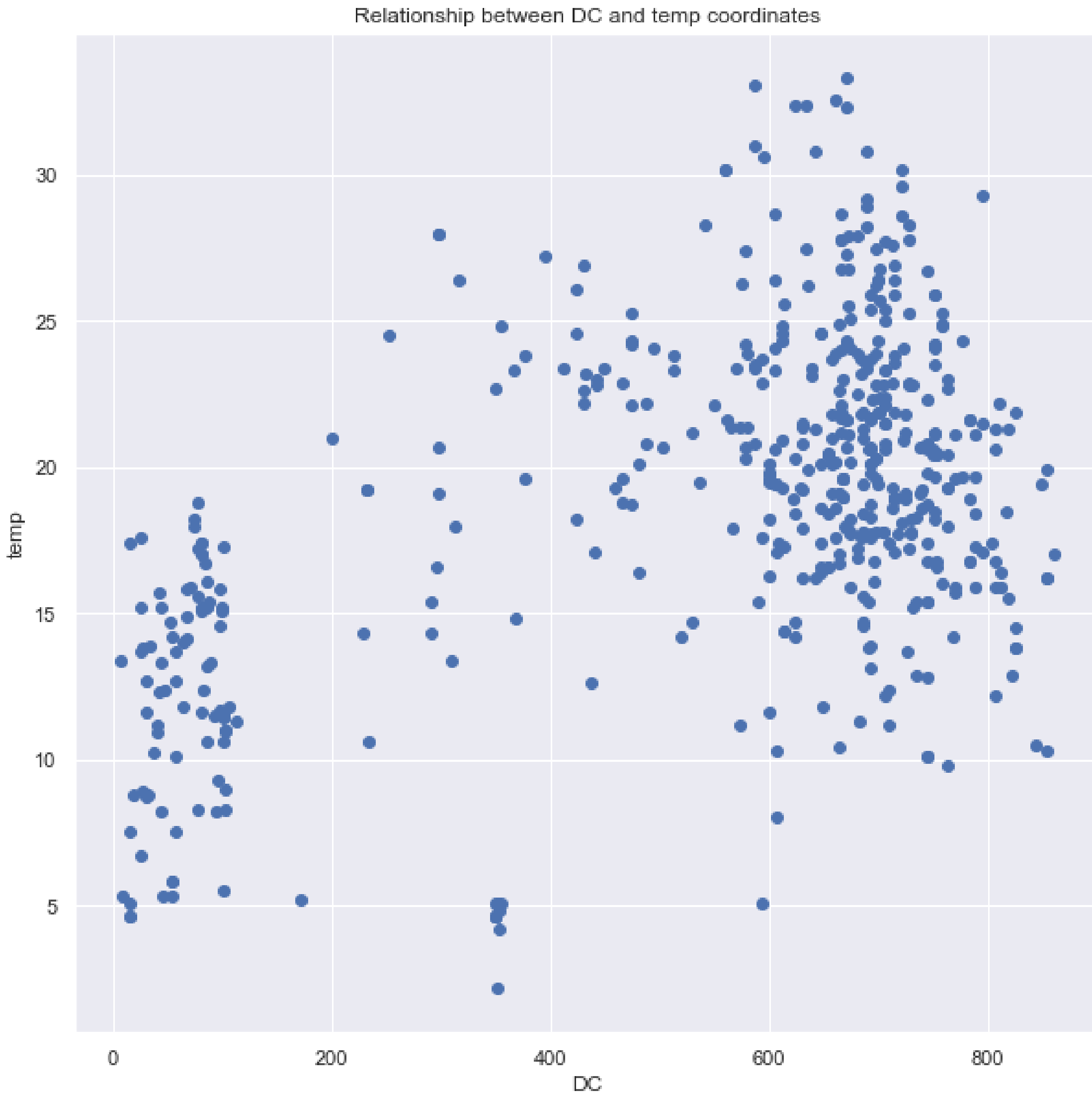
Relationship between DMC and ISI coordinates
ISI
DMC
0
10
20
30
40
50
0
50
100
150
200
250
300

Relation between DC with ISI, Temp.

[17]:
```
plt.figure(figsize=(10,10))
x = np.array(dataframe['DC'])
y = np.array(dataframe['ISI'])
plt.title('Relationship between DC and ISI coordinates')
plt.xlabel('DC')
plt.ylabel('ISI')
plt.scatter(x, y)
plt.show()

plt.figure(figsize=(10,10))
x = np.array(dataframe['DC'])
```

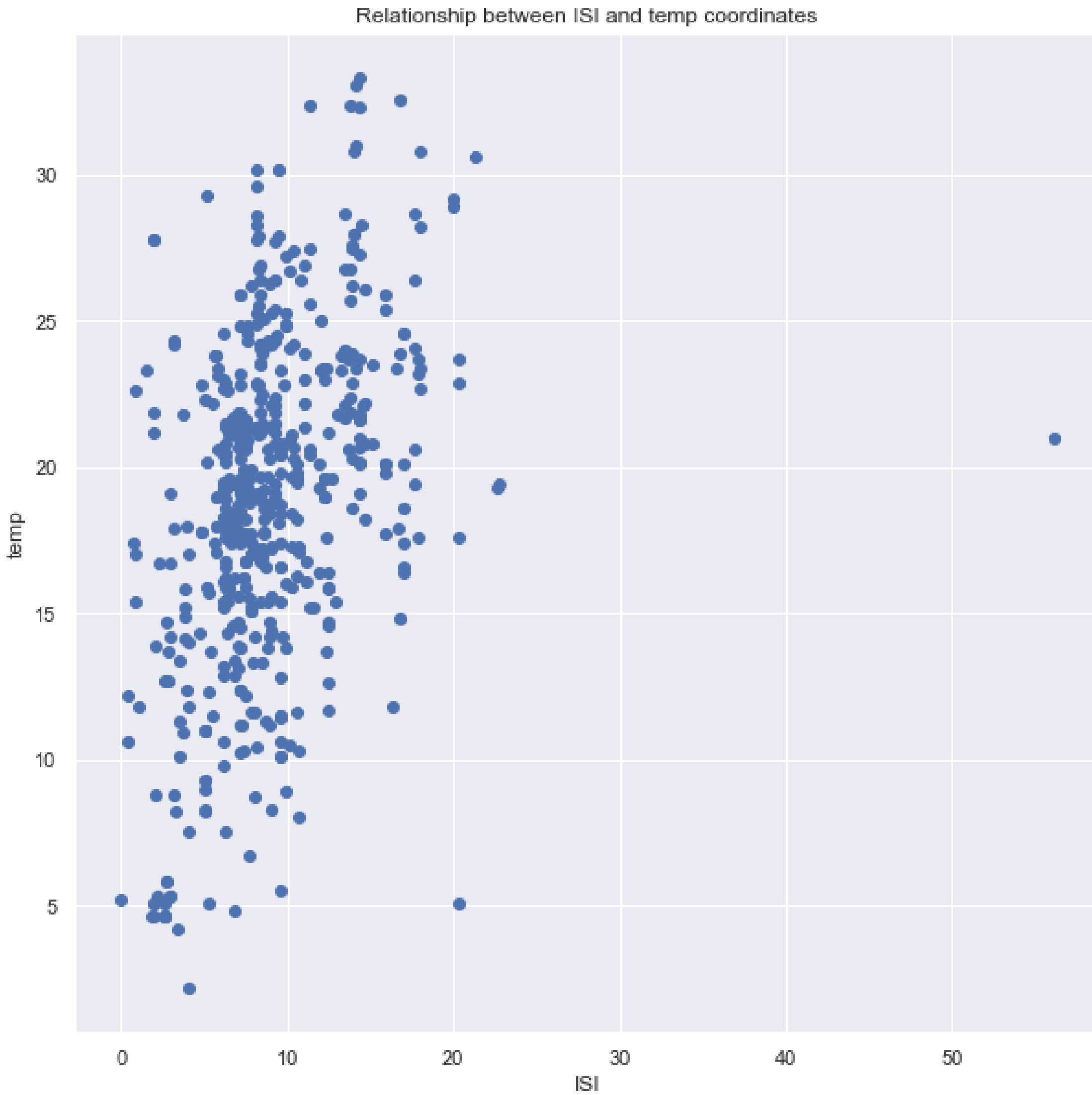
```
y = np.array(dataframe['temp'])
plt.title('Relationship between DC and temp coordinates')
plt.xlabel('DC')
plt.ylabel('temp')
plt.scatter(x, y)
plt.show()
```





Relation between ISI with Temp.

[18]:
```
plt.figure(figsize=(10,10))
x = np.array(dataframe['ISI'])
y = np.array(dataframe['temp'])
plt.title('Relationship between ISI and temp coordinates')
plt.xlabel('ISI')
plt.ylabel('temp')
plt.scatter(x, y)
plt.show()
```



[19]:
```
#To Know more about the data types
dataframe.dtypes
```

[19]:
```
coord_X       int64
coord_Y       int64
month        object
day          object
FFMC        float64
DMC         float64
DC          float64
ISI         float64
temp        float64
RH            int64
```

```
wind       float64
rain       float64
area       float64
dtype: object
```

for statistics of dataset - Describe method includes: - count: number of entries - mean: average of entries - std: standart deviation - min: minimum entry - 25%: first quantile - 50%: median or second quantile - 75%: third quantile - max: maximum entry - The median is the number that is in middle of the sequence.

[20]:
```
dataframe.describe()
```

[20]:
```
          coord_X     coord_Y        FFMC         DMC          DC         ISI  \
count  517.000000  517.000000  517.000000  517.000000  517.000000  517.000000
mean     4.669246    4.299807   92.091296  110.872340  547.940039    9.021663
std      2.313778    1.229900   37.111003   64.046482  248.066192    4.559477
min      1.000000    2.000000    9.900000    1.100000    7.900000    0.000000
25%      3.000000    4.000000   90.200000   68.600000  437.700000    6.500000
50%      4.000000    4.000000   91.600000  108.300000  664.200000    8.400000
75%      7.000000    5.000000   92.900000  142.400000  713.900000   10.800000
max      9.000000    9.000000  921.000000  291.300000  860.600000   56.100000

             temp          RH        wind        rain         area
count  515.000000  517.000000  517.000000  517.000000   517.000000
mean    18.895922   44.288201    4.017602    0.021663    12.847292
std      5.815985   16.317469    1.791653    0.295959    63.655818
min      2.200000   15.000000    0.400000    0.000000     0.000000
25%     15.550000   33.000000    2.700000    0.000000     0.000000
50%     19.300000   42.000000    4.000000    0.000000     0.520000
75%     22.800000   53.000000    4.900000    0.000000     6.570000
max     33.300000  100.000000    9.400000    6.400000  1090.840000
```

0.5.2 Visualizing every attribute of dataset

Now we are visualizing the data for each attribute in the forest fire dataset.

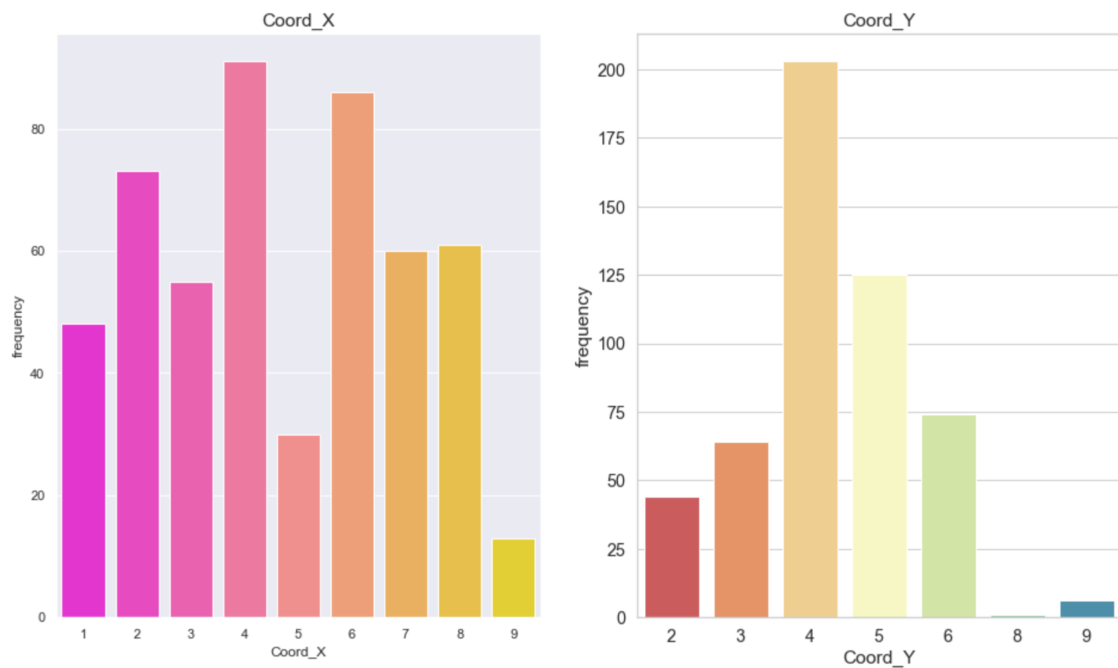
I have choosen to visualize coord_x, coord_y, month, day attributes using seaborn.countplot() method to visualize counts of observations.

For all other attributes we are using plot() function to draw point markers in the diagram. As visualizing data in histogram is would not give the understandable visualization.

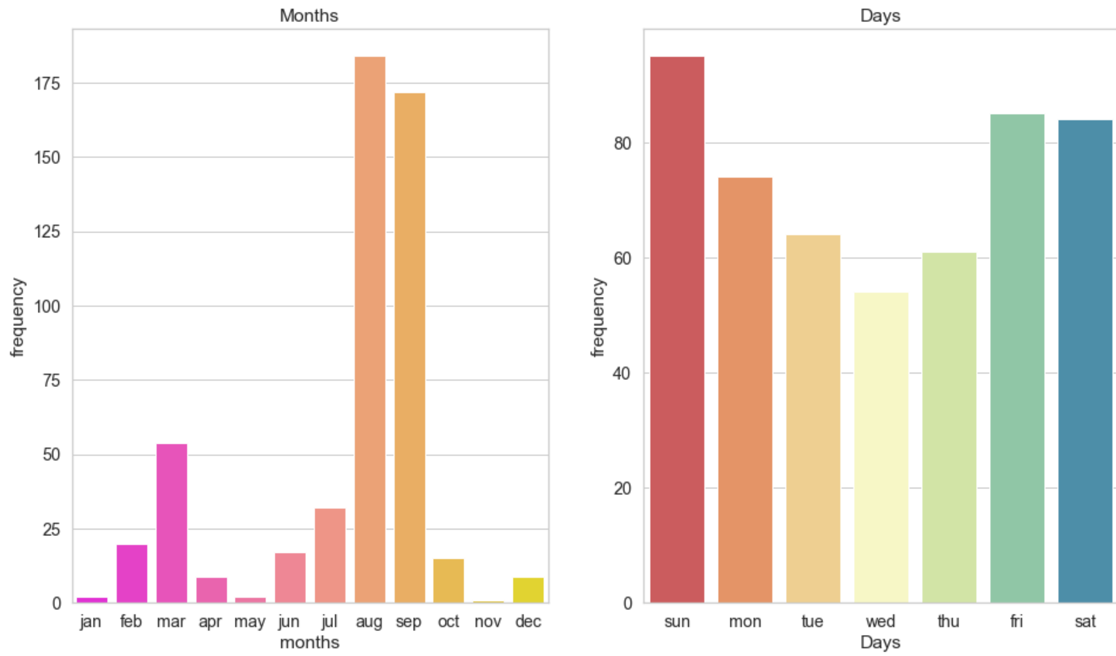
[22]:
```
plt.figure(figsize=(16,9))
plt.subplot(1, 2, 1)
sns.set(style = 'whitegrid', font_scale = 1.3)
day = sns.countplot(dataframe['coord_X'], palette = 'spring')
day.set(title = 'Coord_X', xlabel = 'Coord_X', ylabel = 'frequency')
```

```
plt.subplot(1, 2, 2)
sns.set(style = 'whitegrid', font_scale = 1.3)
day = sns.countplot(dataframe['coord_Y'], palette = 'Spectral')
day.set(title = 'Coord_Y', xlabel = 'Coord_Y', ylabel = 'frequency')
```

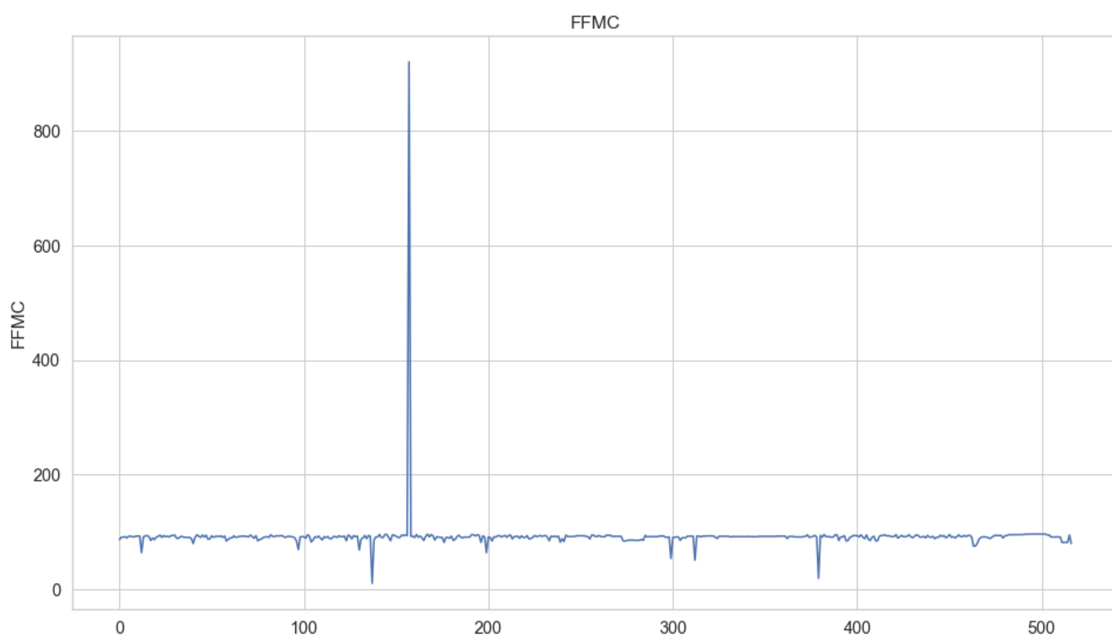
[22]: [Text(0.5, 1.0, 'Coord_Y'), Text(0.5, 0, 'Coord_Y'), Text(0, 0.5, 'frequency')]



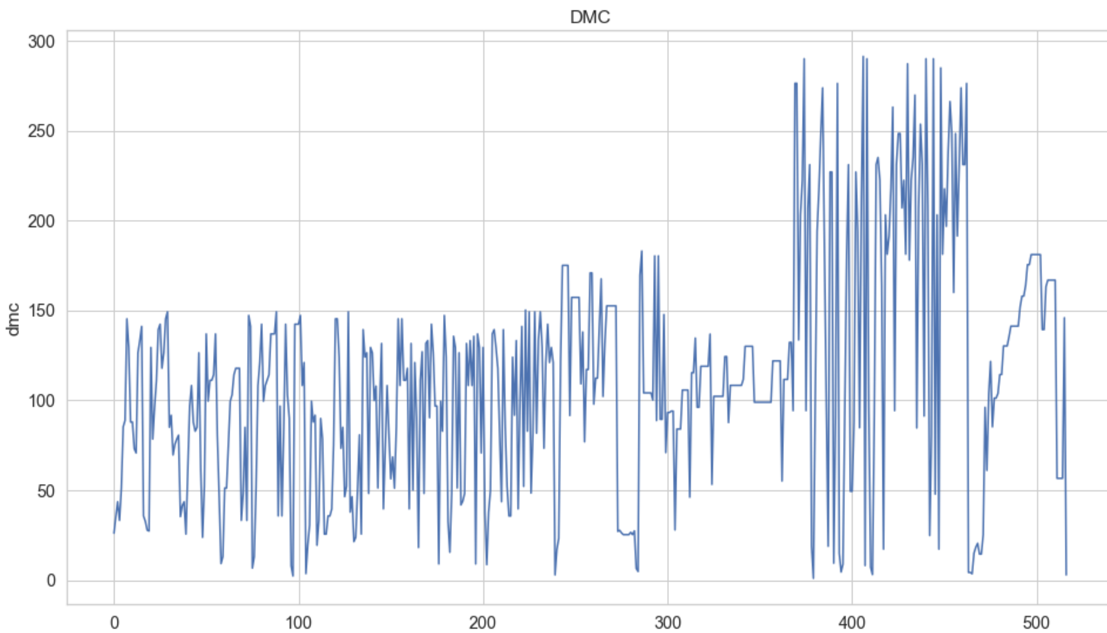
[23]:
```
plt.figure(figsize=(16,9))
plt.subplot(1, 2, 1)
sns.set(style = 'whitegrid', font_scale = 1.3)
day = sns.countplot(dataframe['month'], order = ['jan' ,'feb', 'mar', 'apr',␣
↪'may', 'jun', 'jul', 'aug', 'sep', 'oct', 'nov', 'dec' ], palette = 'spring')
day.set(title = 'Months', xlabel = 'months', ylabel = 'frequency');


plt.subplot(1, 2, 2)
sns.set(style = 'whitegrid', font_scale = 1.3)
day = sns.countplot(dataframe['day'], order = ['sun' ,'mon', 'tue', 'wed',␣
↪'thu', 'fri', 'sat'], palette = 'Spectral')
day.set(title = 'Days', xlabel = 'Days', ylabel = 'frequency');
```



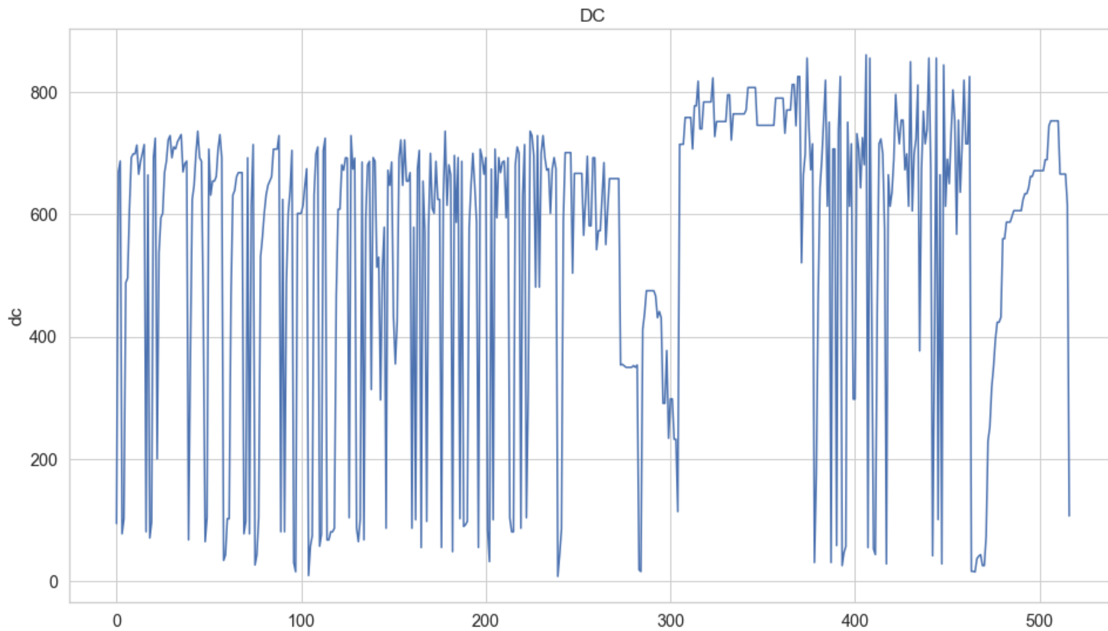
[24]:
```
plt.figure(figsize=(16,9))
xaxis=np.arange(0,len(dataframe))
yaxis=np.array(dataframe['FFMC'])
plt.plot(xaxis,yaxis)
plt.title('FFMC')
plt.ylabel('FFMC')
plt.show()
```



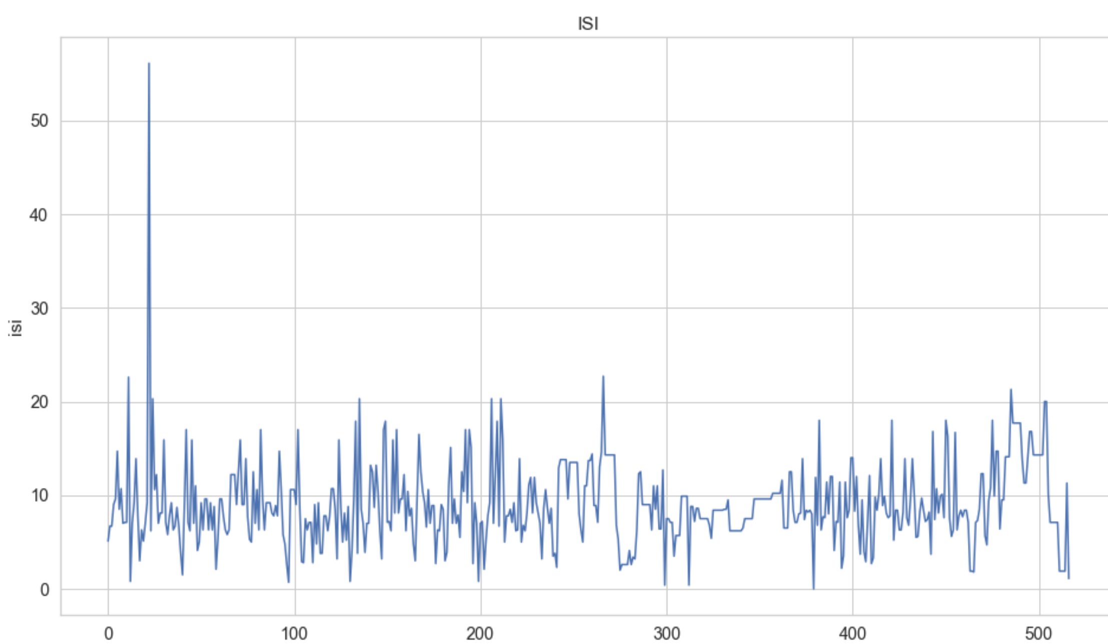
[25]:
```
plt.figure(figsize=(16,9))
xaxis=np.arange(0,len(dataframe))
yaxis=np.array(dataframe['DMC'])
plt.plot(xaxis,yaxis)
plt.title('DMC')
plt.ylabel('dmc')
plt.show()
```



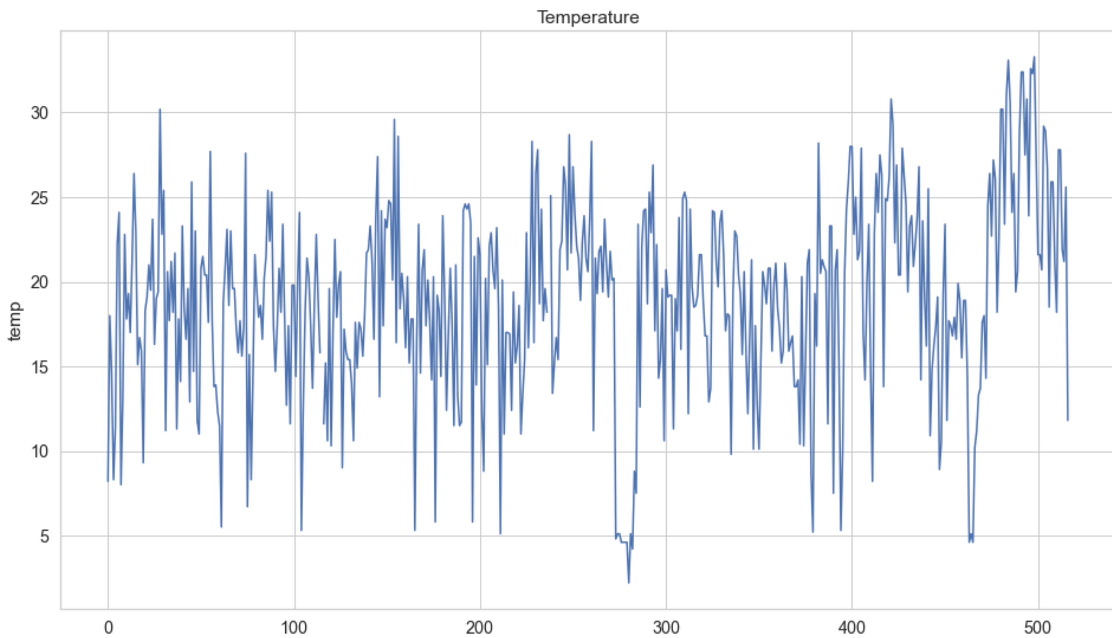
[26]:
```
plt.figure(figsize=(16,9))
xaxis=np.arange(0,len(dataframe))
yaxis=np.array(dataframe['DC'])
plt.plot(xaxis,yaxis)
plt.title('DC')
plt.ylabel('dc')
plt.show()
```



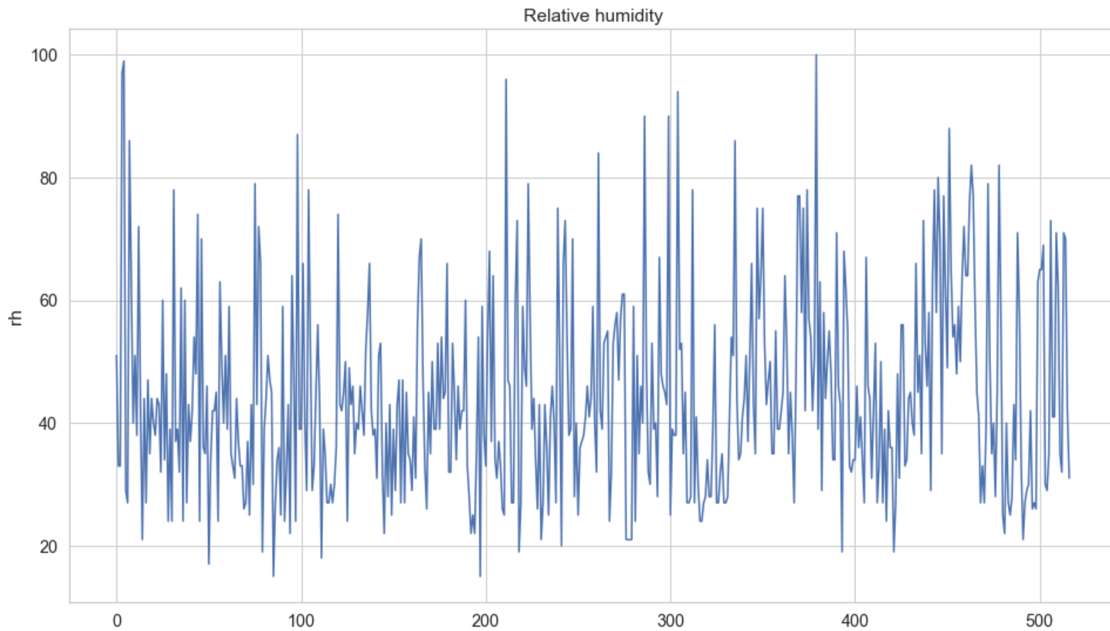
[27]:
```
plt.figure(figsize=(16,9))
xaxis=np.arange(0,len(dataframe))
yaxis=np.array(dataframe['ISI'])
plt.plot(xaxis,yaxis)
plt.title('ISI')
plt.ylabel('isi')
plt.show()
```



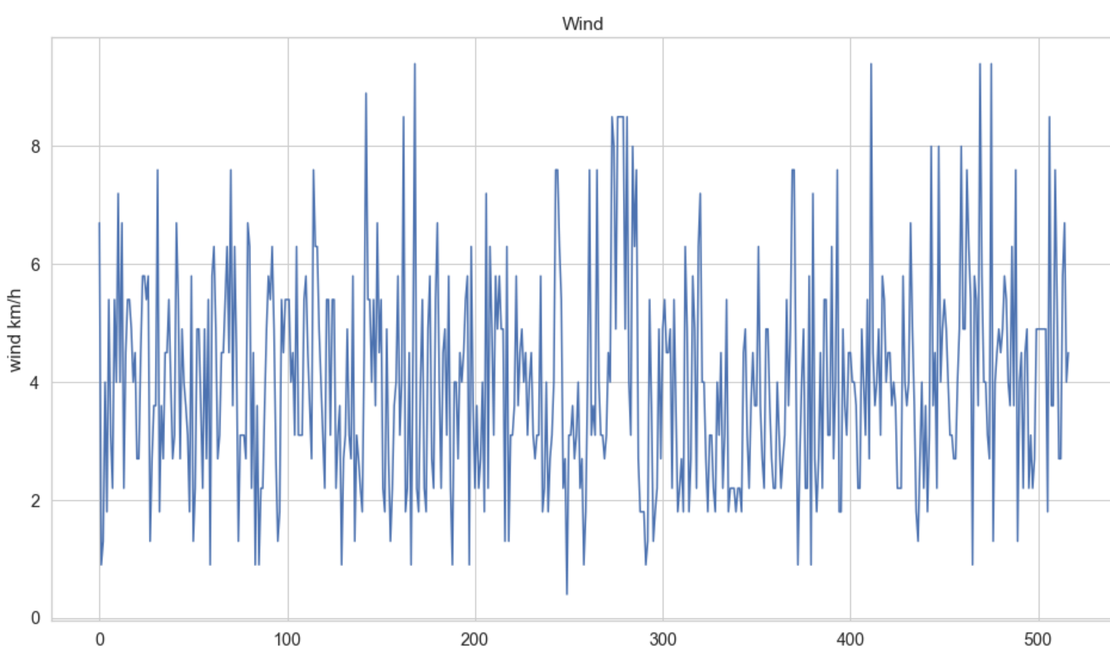
[28]:
```
plt.figure(figsize=(16,9))
xaxis=np.arange(0,len(dataframe))
yaxis=np.array(dataframe['temp'])
plt.plot(xaxis,yaxis)
plt.title('Temperature')
plt.ylabel('temp')
plt.show()
```



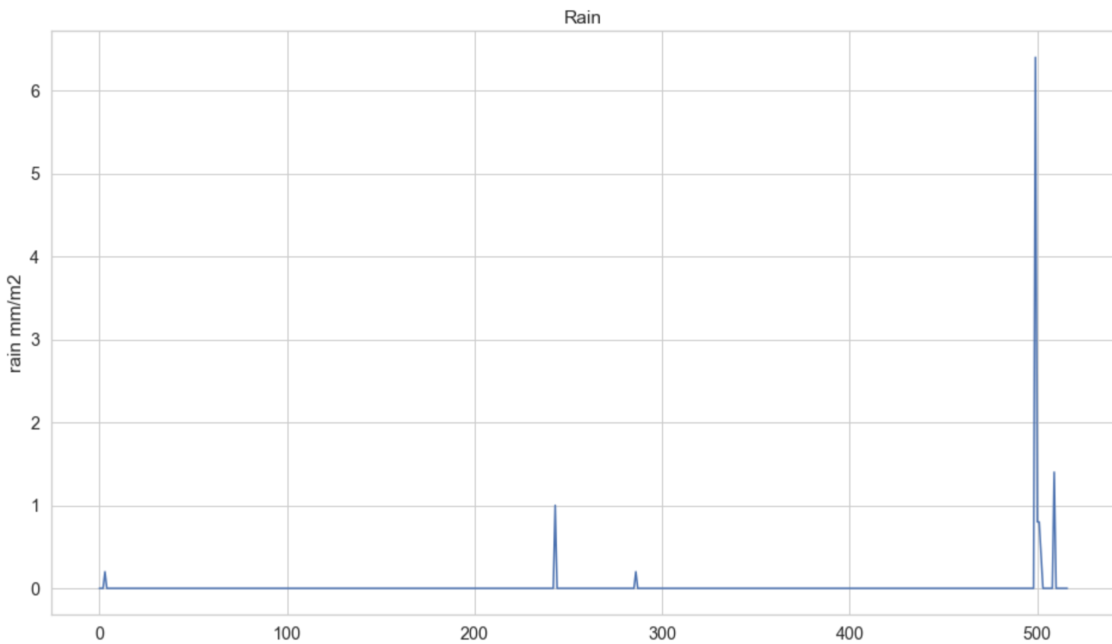
[29]:
```
plt.figure(figsize=(16,9))
xaxis=np.arange(0,len(dataframe))
yaxis=np.array(dataframe['RH'])
plt.plot(xaxis,yaxis)
plt.title('Relative humidity')
plt.ylabel('rh')
plt.show()
```



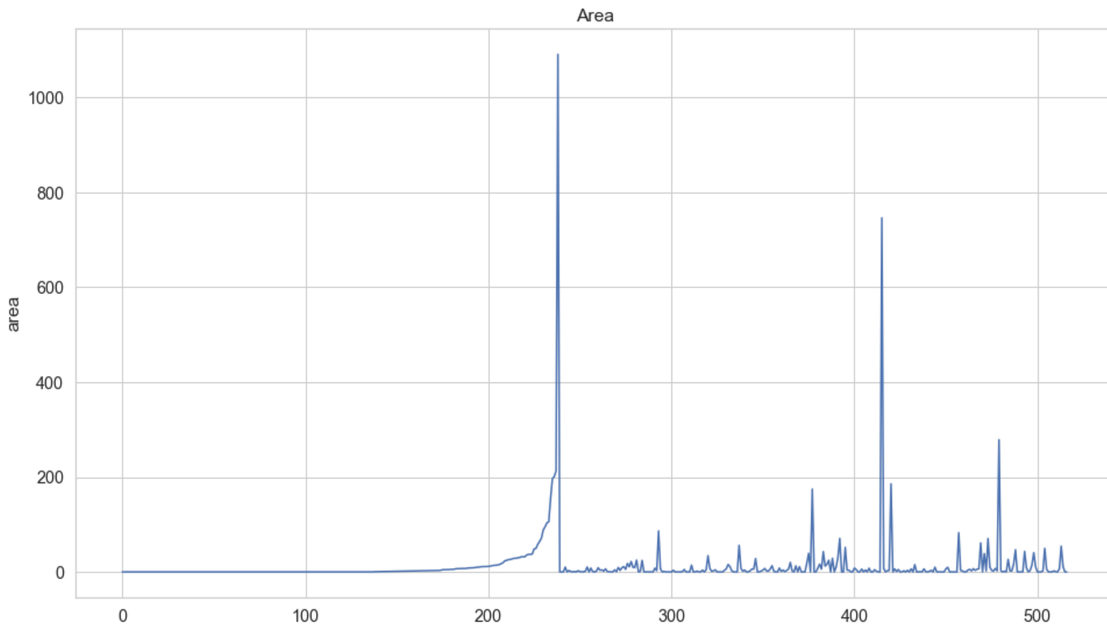
[30]:
```
plt.figure(figsize=(16,9))
xaxis=np.arange(0,len(dataframe))
yaxis=np.array(dataframe['wind'])
plt.plot(xaxis,yaxis)
plt.title('Wind')
plt.ylabel('wind km/h')
plt.show()
```



[31]:
```
plt.figure(figsize=(16,9))
xaxis=np.arange(0,len(dataframe))
yaxis=np.array(dataframe['rain'])
plt.plot(xaxis,yaxis)
plt.title('Rain')
plt.ylabel('rain mm/m2')
plt.show()
```



[32]:
```
plt.figure(figsize=(16,9))
xaxis=np.arange(0,len(dataframe))
yaxis=np.array(dataframe['area'])
plt.plot(xaxis,yaxis)
plt.title('Area')
plt.ylabel('area')
plt.show()
```



0.6 Mining Analytics

Further we can build a model that can show when the forest fire can happen by prediction .

0.7 Evaluation

Not applicable

0.8 Results:

1. According to the above observation made attributes the client might consider using are month or temp or area or rain attributes.
2. Attributes the client might consider adding or creating are forest_type this attribiute gives weather forest is dry or green as there is a high chance of forest fires happening when the forest is dry.
3. The dependent variable the client might consider using when building a model is area.
4. The independent variable the client might consider using when building a model is rest of attributes in the forest fire dataset.
5. Missing or inconsistent values have an impact on data analysis, as evidenced by the temperature attribute and our inability to compute the median for this attribute due to missing values.
6. There are always some relationships between the attributes of a dataset. This can be either positive or negative. This is information that is required for feature selection when designing ML models.
7. When the data is carefully sampled, the distribution and statistics for all attributes in train and test data are very similar. This is especially important when training and evaluating ML models. If these dtributions are incorrect, the evaluations become inconclusive.

0.9 References:

Anaconda Cloud. Wordcloud. Retrieved (2022, January 27) from https://anaconda.org/conda-forge/wordcloud

Python pool. (2021). Retrieved (2022, February 17) from https://www.pythonpool.com/matplotlib-figsize

Justify the text in jupyter.Retrieved (2022, February 17) from https://stackoverflow.com/questions/35077507/how-to-right-align-and-justify-align-in-markdown

NumPy.Retrieved (2022, February 17) from https://numpy.org/doc/stable/index.html

creating subplots.Retrieved (2022, February 17) from https://matplotlib.org/stable/gallery/subplots_axes_and_fi

Remove unnecssary future warnings while using seaborn.Retrieved (2022, February 17) from https://stackoverflow.com/questions/64130332/seaborn-futurewarning-pass-the-following-variables-as-keyword-args-x-y

countplot.Retrieved (2022, February 17) from https://machinelearningknowledge.ai/seaborn-countplot-tutorial-for-beginners/

Pandas Package. Retrieved (2022, February 17) from https://pandas.pydata.org/

Matplotlib.Retrieved (2022, February 17) from https://matplotlib.org/

Seaborn.Retrieved (2022, February 17) from https://seaborn.pydata.org/

Sk learn. Retrieved (2022, February 17) from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

iloc.Retrieved (2022, February 17) from https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.iloc.htm