Prediction of burned areas of forest fire using data mining Techniques

Name: Chandrashekar Akkenapally

Email:chandrashekar.akkenapally@mst.edu (mailto:chandrashekar.akkenapally@mst.edu)

Course: CS 5402

Assignment: Programming Assignment 2

Repository:https://git-classes.mst.edu/ca7kr/cs5402-prog02/ (https://git-classes.mst.edu/ca7kr/cs5402-prog02/)

Date: 06/28/2022

Concept Description

Exploring a data mining approach to predict the burned areas of the forest fire using different attributes such as month, temp, area etc. Segaration of attributes into four levels of measurements to understand the burned areas of the forest fire.

Data Collection

The client has provided the data set on forest fires which they had collected in the form of a comma separated file.

Example Description

Forest fire data set has 13 attributes which will help us to predict the burned area and explore different techniques using the various attributes.
Coord_X: x-axis spatial coordinate within a topographical map of the area of interest.
Example: 7
Coord_Y: y-axis spatial coordinate within a topographical map of the area of interest.
Example: 5
Month: The month in which the forest fire happened.
Example: mar
Day: The day of the week in which the forest fire happened.
Example: fri
FFMC: Fine Fuel Moisture Code from the Fire Weather Index (FWI) System.
Example: 86.2
DMC: Duff Moisture Code from the Fire Weather Index (FWI) System.
Example: 26.2
DC: Draught Code from the Fire Weather Index (FWI) System.
Example: 94.3
ISI: Initial Spread Index from the Fire Weather Index (FWI) System.
Example: 5.1
Temp: Temperature in Celsius degrees.
Example: 8.2
RH: Relative humidity in %.
Example: 51
Wind: Wind speed in km/h.
Example: 6.7
Rain: Outside rain in mm/m2.
Example: 0.0
Area: The burned area of the forest in hectares.
Example: 0.00

Data Import and Wrangling

1) Forest fire csv files are read using pandas.
2) Importing all the required functions for data mining and analysis.
3) Displaying the top 5 rows from the Forest fire csv files.

In [54]:

```
import numpy as np
import pandas as pd
import collections
import matplotlib.cm as cm
import matplotlib as mpl
from matplotlib import rcParams
import matplotlib.pyplot as plt
%matplotlib inline
from pandas.plotting import scatter_matrix
from wordcloud import WordCloud, STOPWORDS
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

forest_data = pd.read_csv(r"C:\Users\MYPC\Documents\Introduction to Data Mining\Assignment2
                  ,skip_blank_lines=True,na_filter=True,encoding='latin-1')
forest_data.head() # displays top 5 rows
```

Out[54]:

	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	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0

In [55]:

```
forest_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 517 entries, 0 to 516
Data columns (total 13 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   coord_X  517 non-null    int64  
 1   coord_Y  517 non-null    int64  
 2   month    517 non-null    object 
 3   day      517 non-null    object 
 4   FFMC     517 non-null    float64
 5   DMC      517 non-null    float64
 6   DC       517 non-null    float64
 7   ISI      517 non-null    float64
 8   temp     515 non-null    float64
 9   RH       517 non-null    int64  
 10  wind     517 non-null    float64
 11  rain     517 non-null    float64
 12  area     517 non-null    float64
dtypes: float64(8), int64(3), object(2)
memory usage: 52.6+ KB
```

- All the attributes have 517 entries except temp
- Temp attribute has only 515 entries which are non-null

Level Of Measurement

Nominal

- coord_X
- coord_Y

Ordinal

- month

Interval

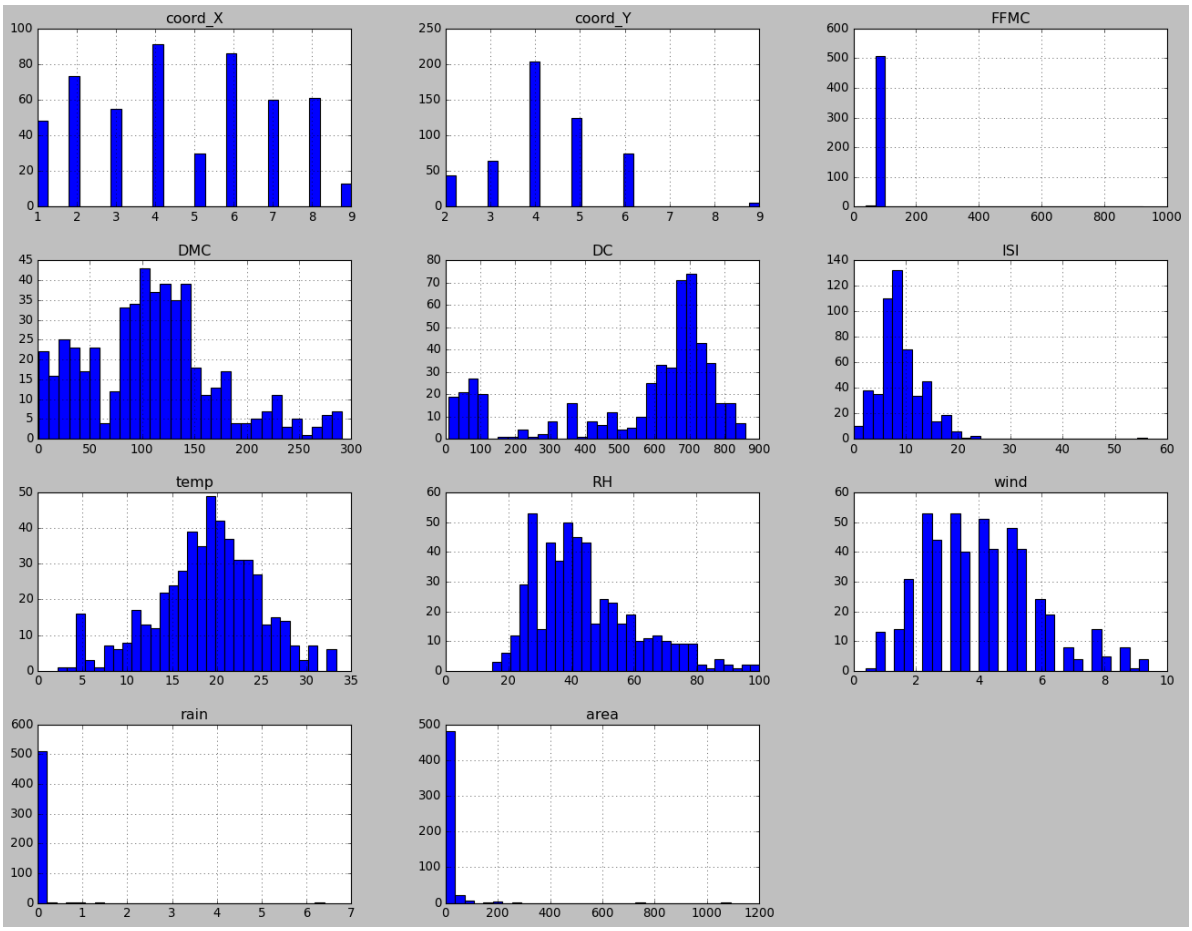
- Day
- FFMC
- DMC
- DC
- ISI
- Temp

Ratio

- RH
- Wind
- Rain
- Area

In [57]:

```
plt.style.use("classic")
forest_data.hist(bins=30, figsize=(20,15))
plt.show()
```



In [58]:

```
forest_data.describe()
```

Out[58]:

	coord_X	coord_Y	FFMC	DMC	DC	ISI	temp
count	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	515.000000
mean	4.669246	4.299807	92.091296	110.872340	547.940039	9.021663	18.895922
std	2.313778	1.229900	37.111003	64.046482	248.066192	4.559477	5.815985
min	1.000000	2.000000	9.900000	1.100000	7.900000	0.000000	2.200000
25%	3.000000	4.000000	90.200000	68.600000	437.700000	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	10.800000	22.800000
max	9.000000	9.000000	921.000000	291.300000	860.600000	56.100000	33.300000

In [59]:

```
forest_data.isna().sum()
```

Out[59]:

```
coord_X    0
coord_Y    0
month      0
day        0
FFMC       0
DMC        0
DC         0
ISI        0
temp       2
RH         0
wind       0
rain       0
area       0
dtype: int64
```

Temp attribute has 2 NaN- Not A Number values, which can be seen below

In [60]:

```
forest_data[forest_data['temp'].isna()]
```

Out[60]:

	coord_X	coord_Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
115	3	5	mar	tue	88.1	25.7	67.6	3.8	NaN	27	6.3	0.0	0.00
237	1	2	sep	tue	91.0	129.5	692.6	7.0	NaN	40	2.2	0.0	212.88

The missing data can be filled in two ways

- By replacing the mean value.

- By droping the rows from the dataset.

Here, we are replacing the NaN values with the mean of the temp.

In [61]:

```
forest_data["temp"].fillna(value=forest_data["temp"].mean(), inplace=True)
```

In [63]:

```
forest_data.isna().sum()
```

Out[63]:

```
coord_X    0
coord_Y    0
month      0
day        0
FFMC       0
DMC        0
DC         0
ISI        0
temp       0
RH         0
wind       0
rain       0
area       0
dtype: int64
```

Replacing the month values and day values with the numbers. It will help us to understand and perform the different data mining techniques.

In [64]:

```
# Coverting the days and months into the integers
forest_data.month.replace(('jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov
forest_data.day.replace(('mon','tue','wed','thu','fri','sat','sun'),(1,2,3,4,5,6,7), inplac
```

Exploratory Data Analysis

Spliting the forest data set into training data and test data.

In [65]:

```
# dividing the data into test and training sets
training_data, test_data = train_test_split(forest_data, test_size=0.2, random_state=42)
work_set = training_data.copy() # assigning a copy of train set to work_set
```

In [66]:

```
training_data.describe()
```

Out[66]:

	coord_X	coord_Y	month	day	FFMC	DMC	DC	
count	413.000000	413.000000	413.000000	413.000000	413.000000	413.000000	413.000000	41
mean	4.661017	4.322034	7.476998	4.225182	92.540436	112.178692	549.813075	
std	2.294222	1.208957	2.276426	2.107579	41.296066	65.814101	249.214533	
min	1.000000	2.000000	1.000000	1.000000	18.700000	1.100000	7.900000	
25%	3.000000	4.000000	7.000000	2.000000	90.200000	70.800000	437.700000	
50%	4.000000	4.000000	8.000000	5.000000	91.600000	108.400000	664.500000	
75%	7.000000	5.000000	9.000000	6.000000	92.900000	142.400000	715.100000	1
max	9.000000	9.000000	12.000000	7.000000	921.000000	291.300000	860.600000	2

In [67]:

```
test_data.describe()
```

Out[67]:

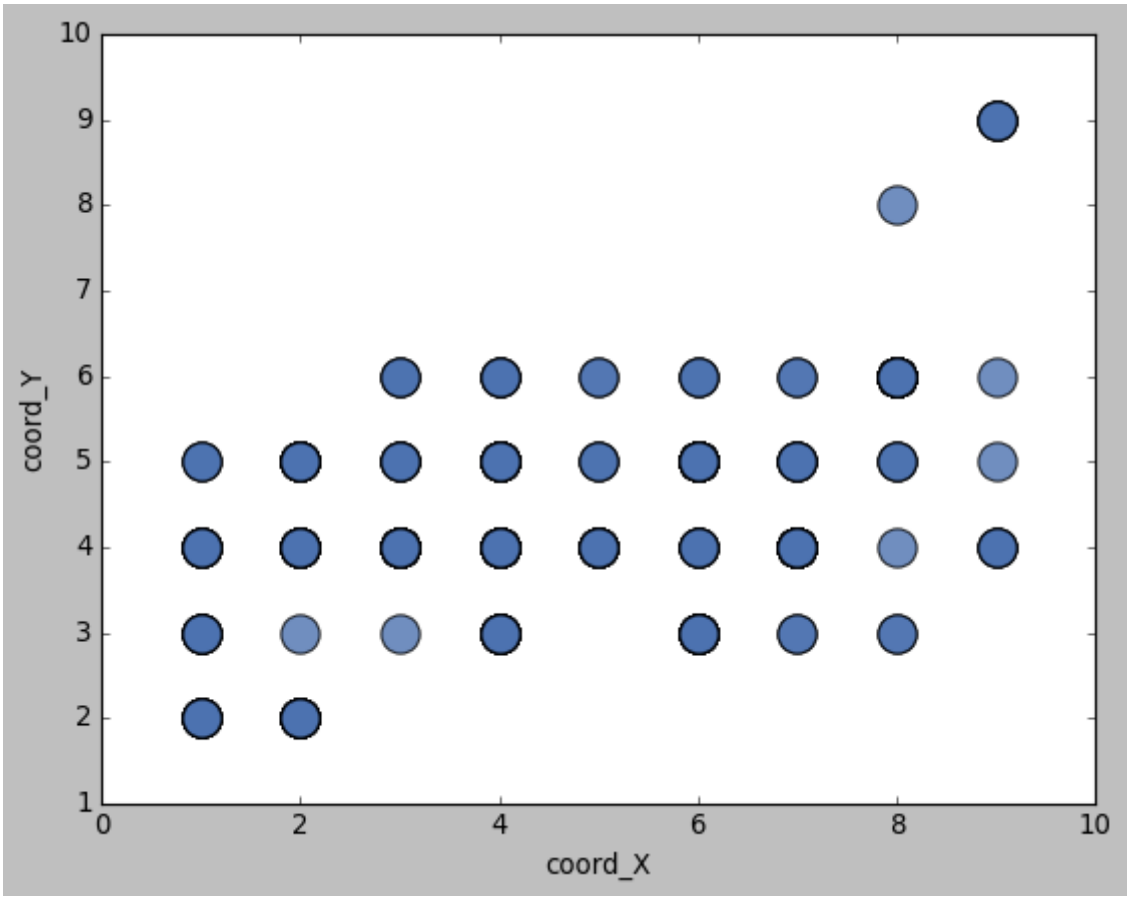
	coord_X	coord_Y	month	day	FFMC	DMC	DC	
count	104.000000	104.000000	104.000000	104.000000	104.000000	104.000000	104.000000	10
mean	4.701923	4.211538	7.471154	4.394231	90.307692	105.684615	540.501923	1
std	2.400970	1.312177	2.285270	1.932945	8.603653	56.476402	244.499813	
min	1.000000	2.000000	1.000000	1.000000	9.900000	3.600000	9.300000	
25%	2.000000	3.000000	8.000000	3.000000	90.200000	56.325000	462.025000	
50%	5.000000	4.000000	8.000000	5.000000	91.600000	108.300000	647.100000	
75%	7.000000	5.000000	9.000000	6.000000	93.100000	139.875000	694.100000	1
max	9.000000	9.000000	12.000000	7.000000	96.100000	248.400000	822.800000	5

In [69]:

```
plt.style.use("seaborn-deep")
work_set.plot(kind='scatter', x='coord_X', y='coord_Y', alpha=0.8, s=300)
```

Out[69]:

```
<AxesSubplot:xlabel='coord_X', ylabel='coord_Y'>
```

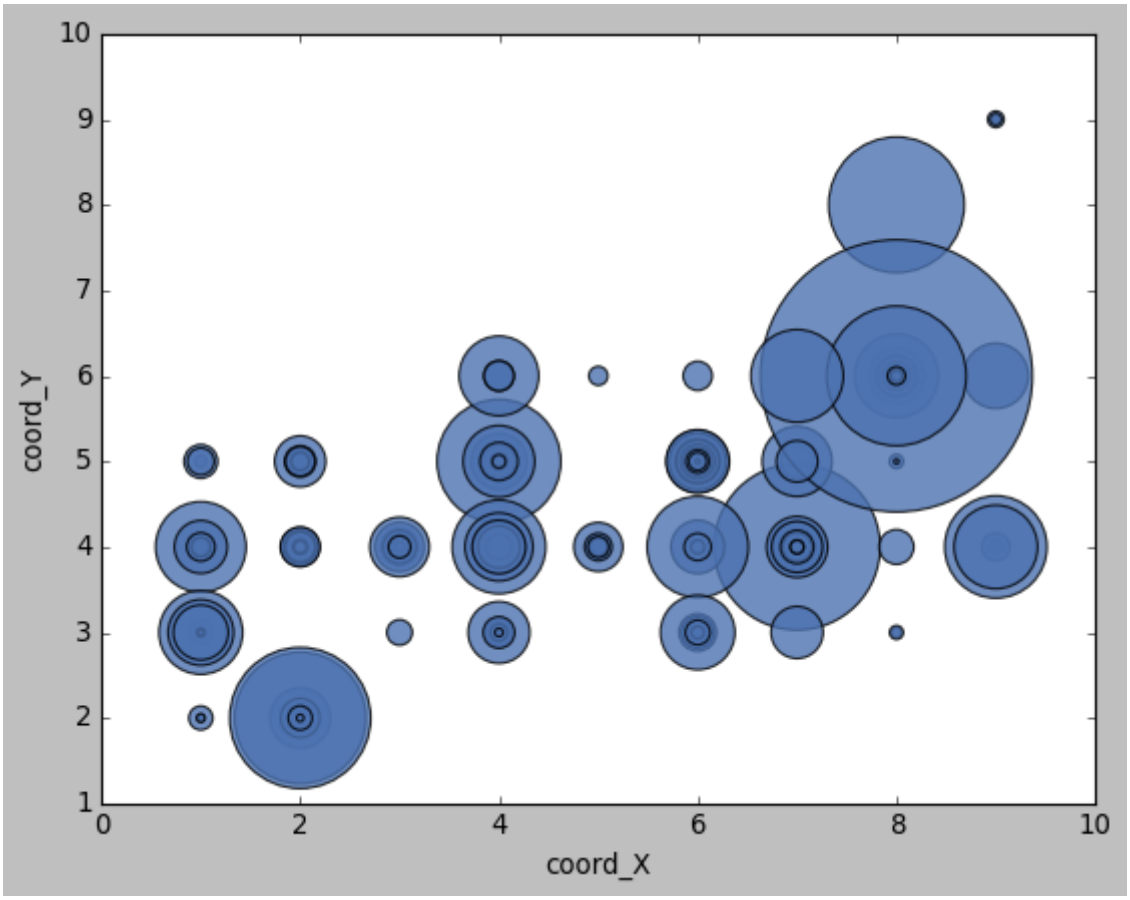


In [71]:

```
plt.style.use("seaborn-deep")
work_set.plot(kind='scatter', x='coord_X', y='coord_Y', alpha=0.8, s=20*work_set['area'])
# plotting the graphs by increasing the size to see the affect of area over the datapoints
```

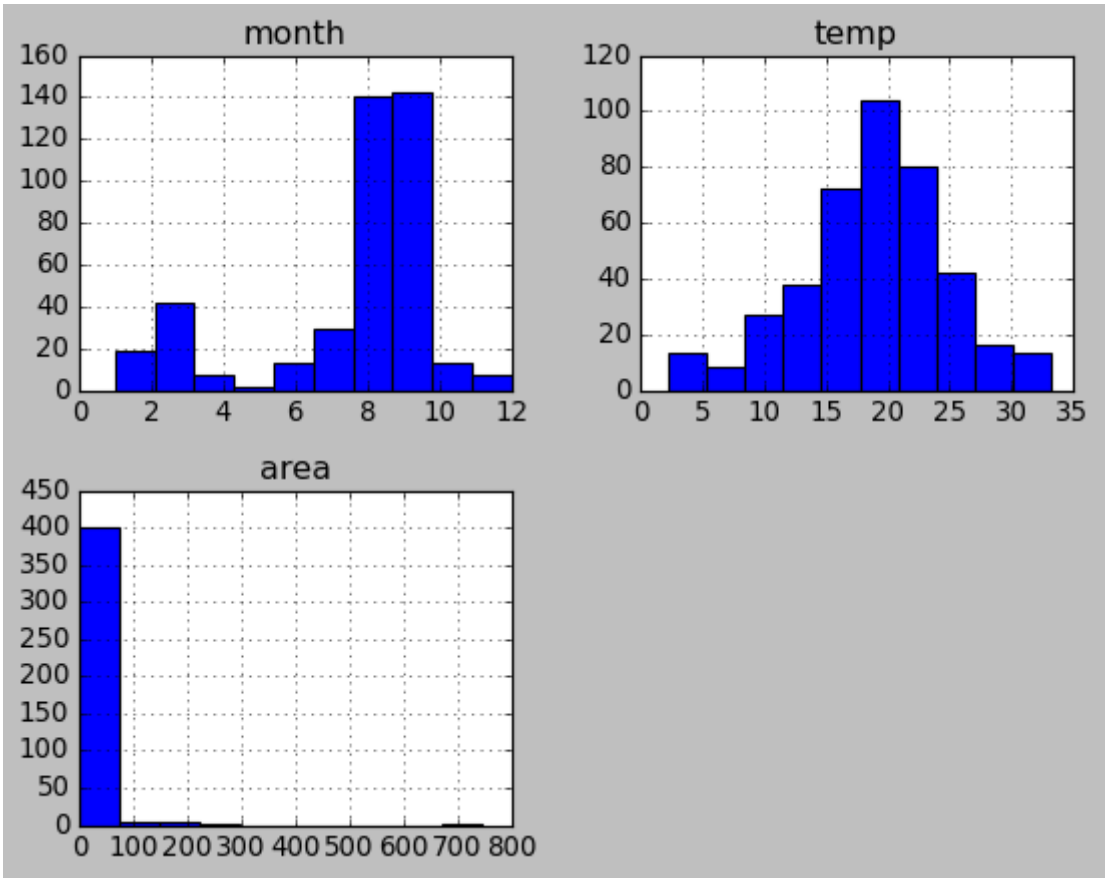
Out[71]:

```
<AxesSubplot:xlabel='coord_X', ylabel='coord_Y'>
```



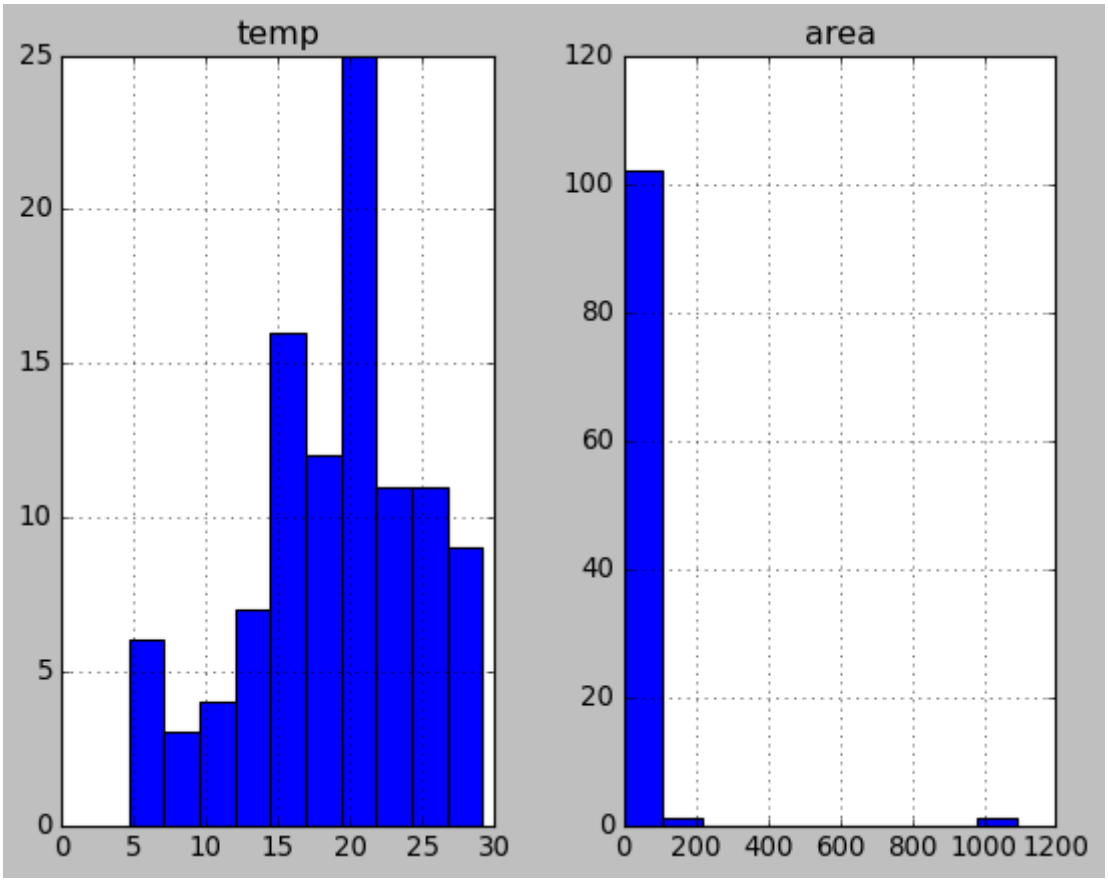
In [72]:

```
plt.style.use("classic")
work_set.hist(["month","temp","area"])
plt.show()
```



In [28]:

```
test_data.hist(["month","temp","area"])
plt.show()
```



Mining or Analytics

Let's find out the overall burned area for each month.
Here we consider three attributes.

- Month from Ordinal.
- Temp from Interval
- Area from Ratio.

In [73]:

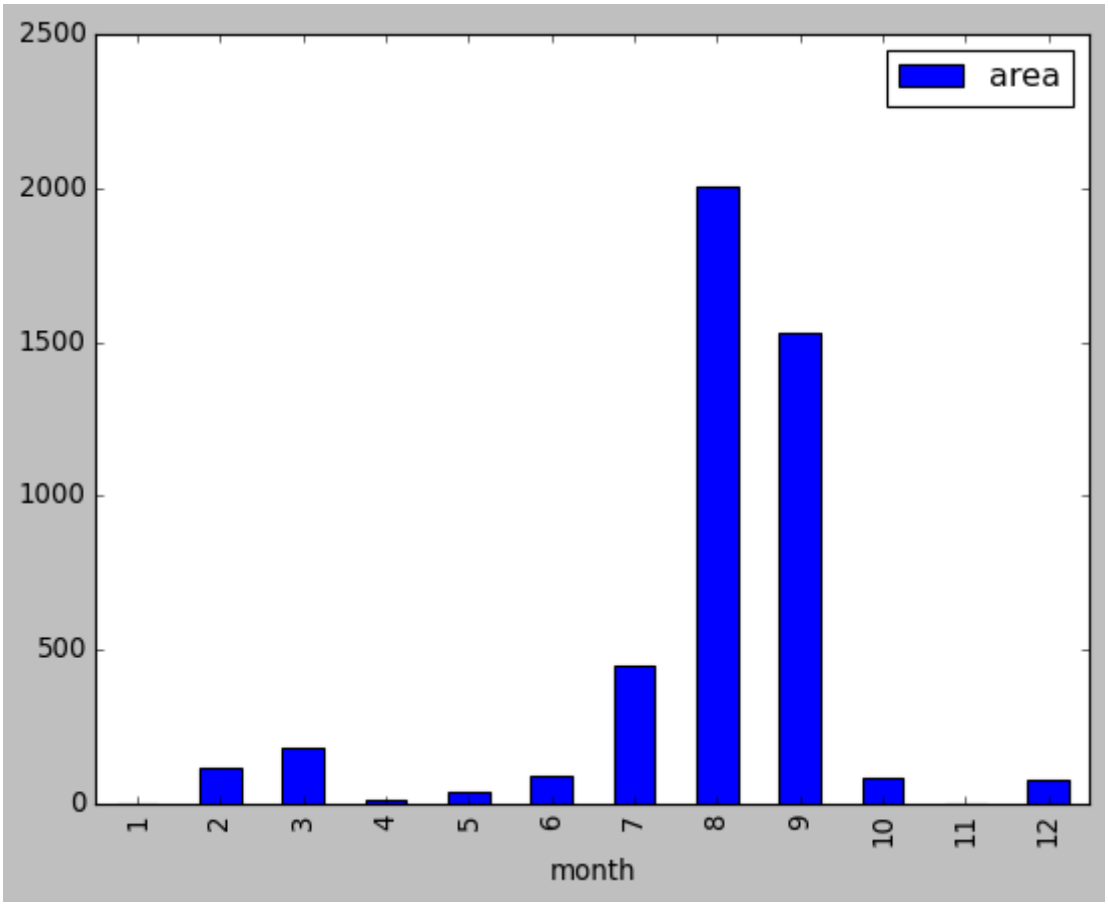
```
#table = pd.pivot_table(work_set, values='area', index=['month','day'], aggfunc=np.sum)
table = pd.pivot_table(work_set, values=['area'], index=['month'], aggfunc=np.sum)
table.reset_index(inplace = True)
table
```

Out[73]:

	month	area
0	1	0.00
1	2	118.66
2	3	184.42
3	4	14.28
4	5	38.48
5	6	90.24
6	7	452.47
7	8	2002.57
8	9	1531.49
9	10	85.87
10	11	0.00
11	12	79.09

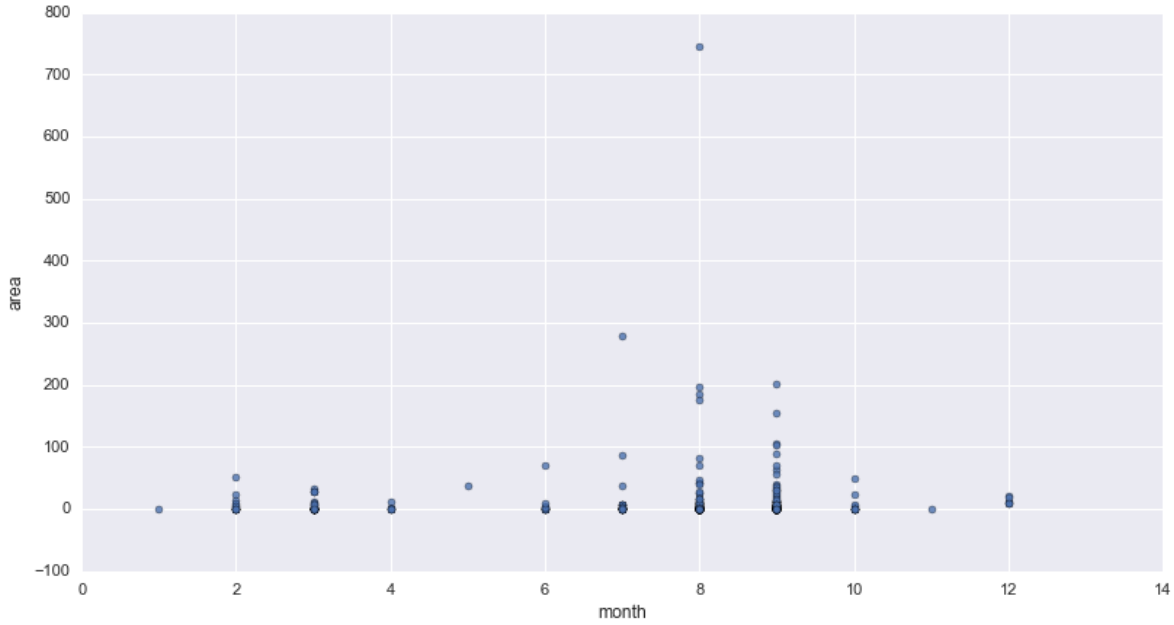
In [74]:

```
table.plot(x = 'month',y ='area',kind='bar')
plt.show()
```



In [75]:

```
plt.style.use("seaborn")
work_set.plot(kind="scatter", x="month", y="area",alpha=0.8, figsize=(12,6))
plt.show()
```

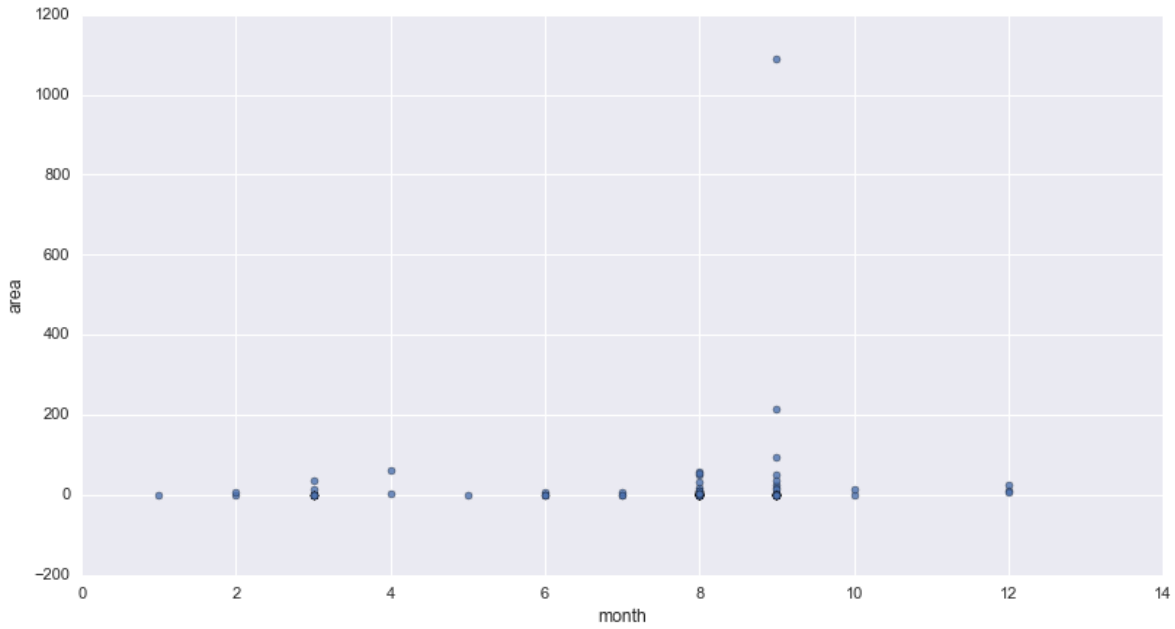


In [76]:

```
test_data.plot(kind="scatter", x="month", y="area", alpha =0.8, figsize=(12,6))
```

Out[76]:

```
<AxesSubplot:xlabel='month', ylabel='area'>
```



In [39]:

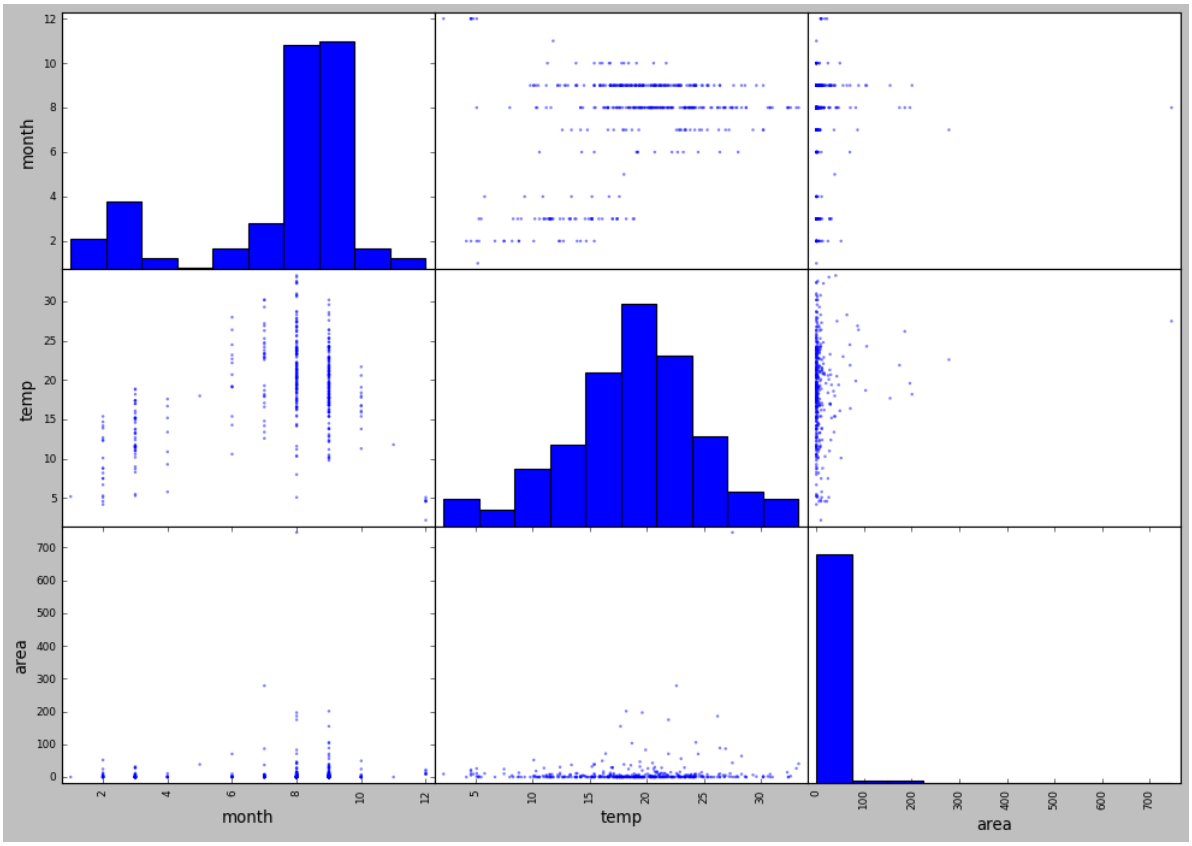
```
test_data.corr()
```

Out[39]:

	coord_X	coord_Y	FFMC	DMC	DC	ISI	temp	RH
coord_X	1.000000	0.577988	0.097119	-0.110089	-0.030782	0.108495	0.017175	-0.105721
coord_Y	0.577988	1.000000	-0.162767	-0.093418	-0.163139	-0.102345	-0.152332	0.125896
FFMC	0.097119	-0.162767	1.000000	0.173974	0.127366	0.268071	0.293929	-0.277033
DMC	-0.110089	-0.093418	0.173974	1.000000	0.683086	0.218095	0.600795	-0.055189
DC	-0.030782	-0.163139	0.127366	0.683086	1.000000	0.107640	0.579464	-0.130591
ISI	0.108495	-0.102345	0.268071	0.218095	0.107640	1.000000	0.356551	-0.139824
temp	0.017175	-0.152332	0.293929	0.600795	0.579464	0.356551	1.000000	-0.448408
RH	-0.105721	0.125896	-0.277033	-0.055189	-0.130591	-0.139824	-0.448408	1.000000
wind	-0.034303	-0.034269	0.031628	-0.226246	-0.245569	0.069715	-0.244874	-0.072239
rain	0.057933	0.015724	0.040157	0.159442	0.101256	-0.006070	0.054819	0.228106
area	0.021807	0.028698	0.030923	0.024292	0.065743	-0.033042	0.116508	-0.126331

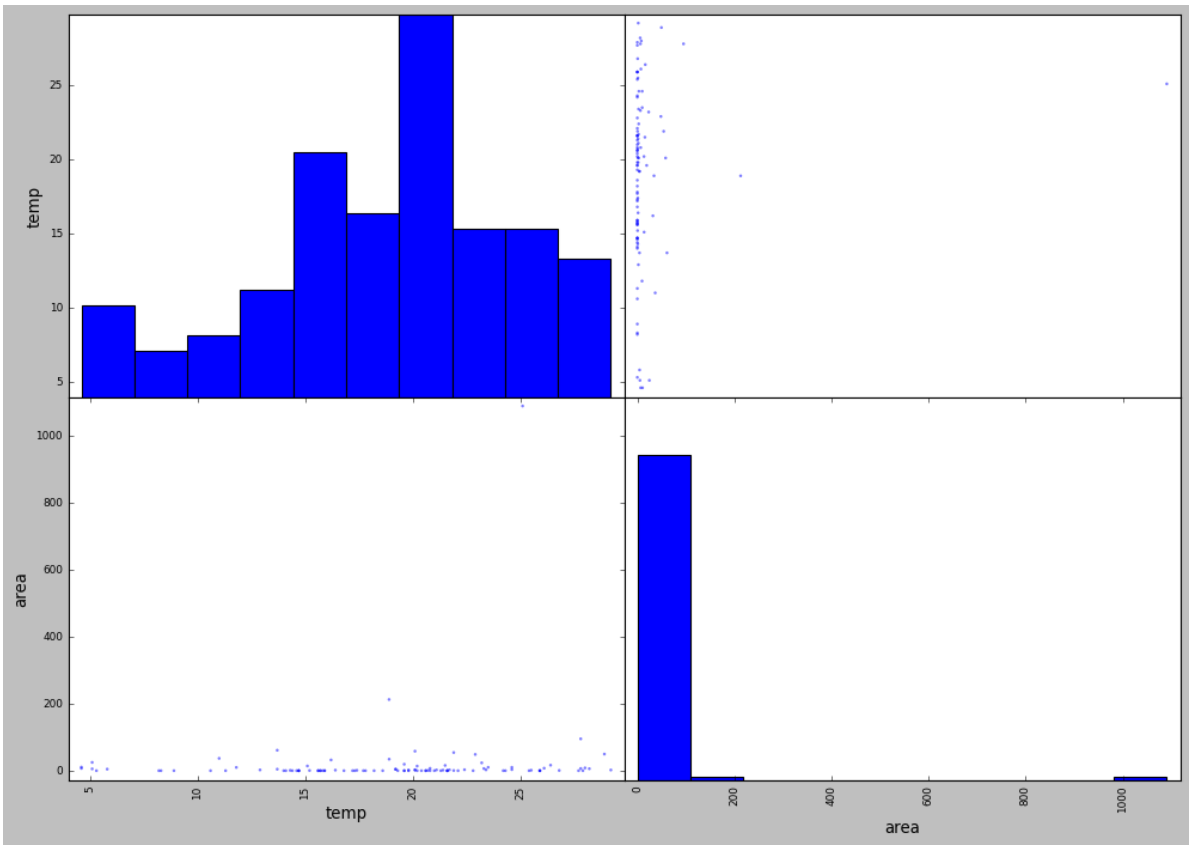
In [77]:

```
plt.style.use("classic")
attributes = ['month', 'temp','area']
scatter_matrix(work_set[attributes], figsize=(15, 10))
plt.show()
```



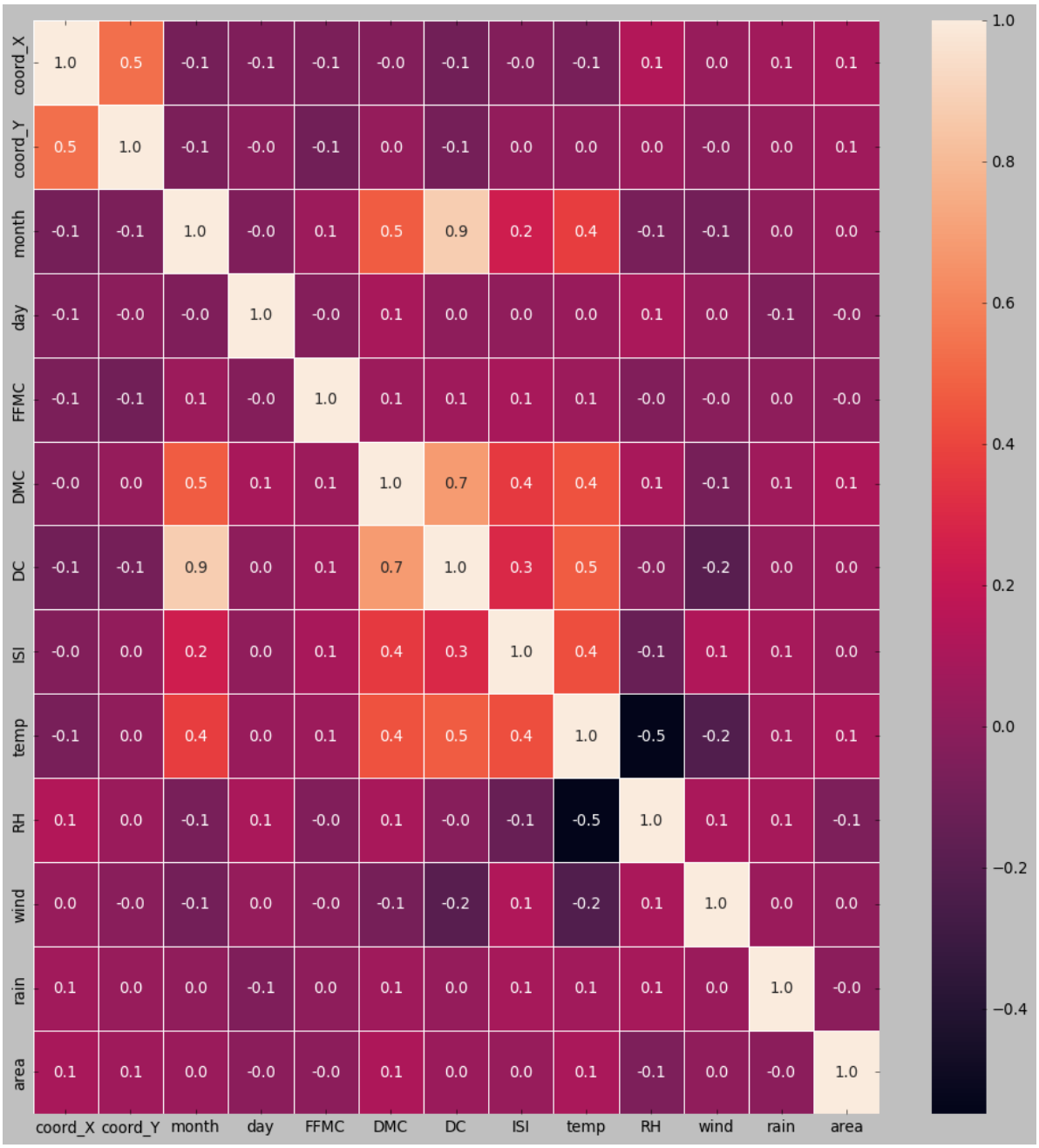
In [41]:

```
attributes = ['month', 'temp','area']
scatter_matrix(test_data[attributes], figsize=(15, 10))
plt.show()
```



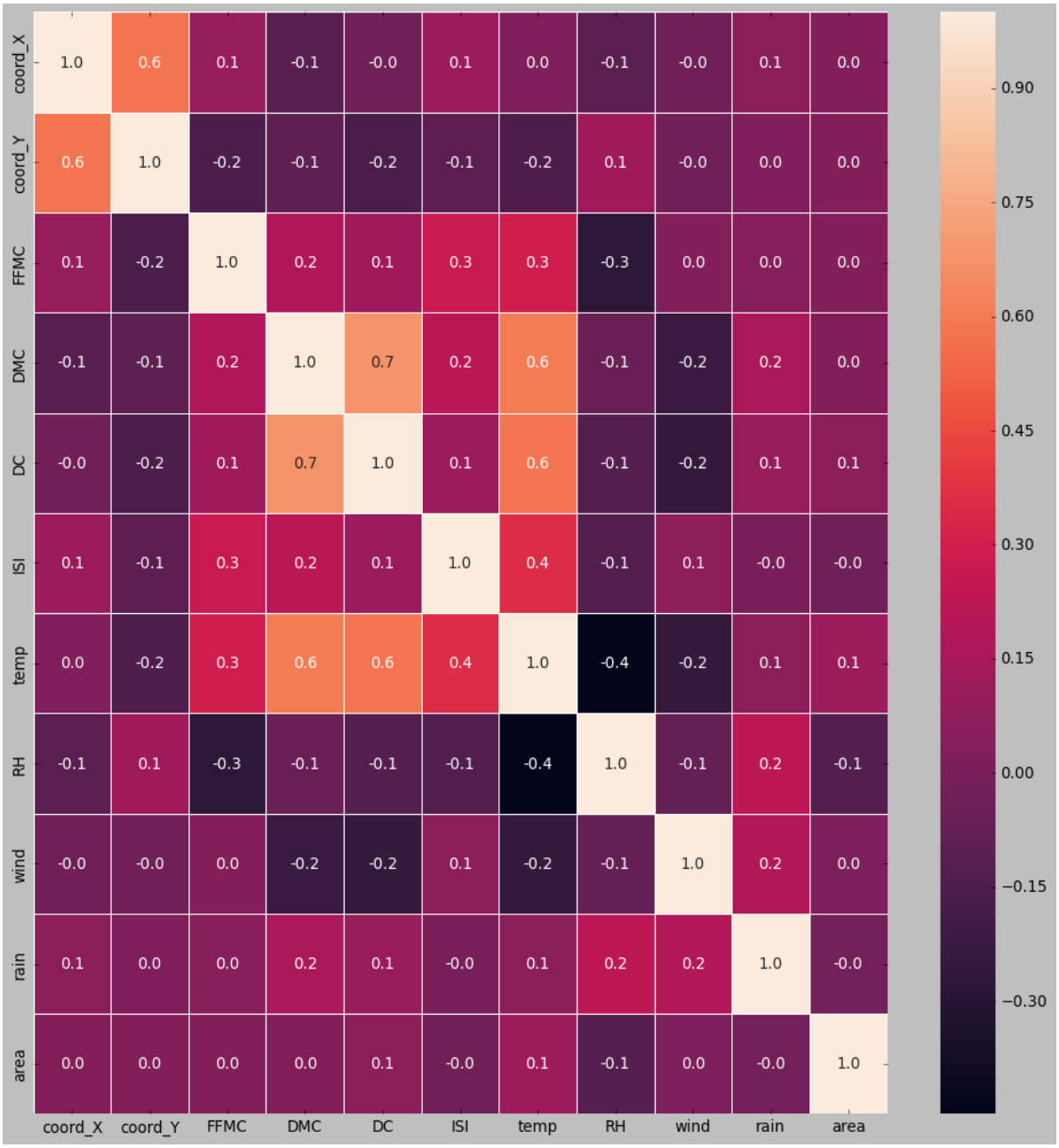
In [78]:

```
import seaborn as sns
#import classic as sns
#correlation map for training data
f,ax = plt.subplots(figsize=(15, 15))
sns.heatmap(work_set.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
plt.show()
```



In [49]:

```
#correlation map for test data
f,ax = plt.subplots(figsize=(15, 15))
sns.heatmap(test_data.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
plt.show()
```



Evaluation

From the training data and test data we can see the similar characteristics using the plots, scattar

matrix and heatmap.

Results

- Considering the month, temp and area variables will help us to predict burned areas.
- Adding the Oxygen field to the dataset might help the client to understand more about the forest fire.
- Month, day and RH can be consider as independent variables.
- Area and temp can be consider as dependent variables.

Reference

- https://pandas.pydata.org/docs (https://pandas.pydata.org/docs)
- https://realpython.com/train-test-split-python-data/ (https://realpython.com/train-test-split-python-data/)
- https://towardsdatascience.com/ (https://towardsdatascience.com/)
- https://pythongeeks.org/python-scatter-plot/ (https://pythongeeks.org/python-scatter-plot/)