Praca_domowa_1

March 9, 2021

1 Praca domowa nr 1 - WUM2021L

1.0.1 Autor: Bartosz Sawicki

[1]:
```
import pandas as pd
import numpy as np
import requests
import seaborn as sns

from matplotlib import pyplot as plt
from scipy import stats
from pandas_profiling import ProfileReport
```

1.1 Pobranie danych

[2]:
```
url = 'https://api.apispreadsheets.com/api/dataset/forest-fires/'
r = requests.get(url)
data = r.json()

df = pd.DataFrame.from_dict(data['data'], orient='columns')
```

1.2 Ogólne informacje o zbiorze

[3]:
```
df.head()
```

[3]:

	X	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	51.0	6.7	0.0	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33.0	0.9	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33.0	1.3	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97.0	4.0	0.2	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99.0	1.8	0.0	0.0

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 517 entries, 0 to 516
Data columns (total 13 columns):
```

```
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   X       517 non-null    int64
 1   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    517 non-null    float64
 9   RH      517 non-null    float64
 10  wind    517 non-null    float64
 11  rain    517 non-null    float64
 12  area    517 non-null    float64
dtypes: float64(9), int64(2), object(2)
memory usage: 52.6+ KB
```

[5]:
```
df.describe()
```

[5]:

	X	Y	FFMC	DMC	DC	ISI
count	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000
mean	4.669246	4.299807	90.644681	110.872340	547.940039	9.021663
std	2.313778	1.229900	5.520111	64.046482	248.066192	4.559477
min	1.000000	2.000000	18.700000	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	96.200000	291.300000	860.600000	56.100000

	temp	RH	wind	rain	area
count	517.000000	517.000000	517.000000	517.000000	517.000000
mean	18.889168	44.288201	4.017602	0.021663	12.847292
std	5.806625	16.317469	1.791653	0.295959	63.655818
min	2.200000	15.000000	0.400000	0.000000	0.000000
25%	15.500000	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

Co oznaczają skróty `FFMC`, `DMC`, `DC`, `ISI`?

W skrócie: wskaźniki systemu **FWI** (Fire Weather Index)

- `FFMC` - The **Fine Fuel Moisture Code** represents fuel moisture of forest litter fuels under the shade of a forest canopy. It is intended to represent moisture conditions for shaded litter fuels, the equivalent of 16-hour timelag. It ranges from 0-101. Subtracting the FFMC value from 100 can provide an estimate for the equivalent (approximately 10h) fuel moisture content, most accurate when FFMC values are roughly above 80.

- `DMC` - The **Duff Moisture Code** represents fuel moisture of decomposed organic material underneath the litter. System designers suggest that it is represents moisture conditions for the equivalent of 15-day (or 360 hr) timelag fuels. It is unitless and open ended. It may provide insight to live fuel moisture stress.
- `DC` - The **Drought Code**, much like the Keetch-Byrum Drought Index, represents drying deep into the soil. It approximates moisture conditions for the equivalent of 53-day (1272 hour) timelag fuels. It is unitless, with a maximum value of 1000. Extreme drought conditions have produced DC values near 800.
- `ISI` - The **Initial Spread Index** integrates fuel moisture for fine dead fuels and surface windspeed to estimate a spread potential. ISI is a key input for fire behavior predictions in the FBP system. It is unitless and open ended.

[6]:
```
df.columns
```

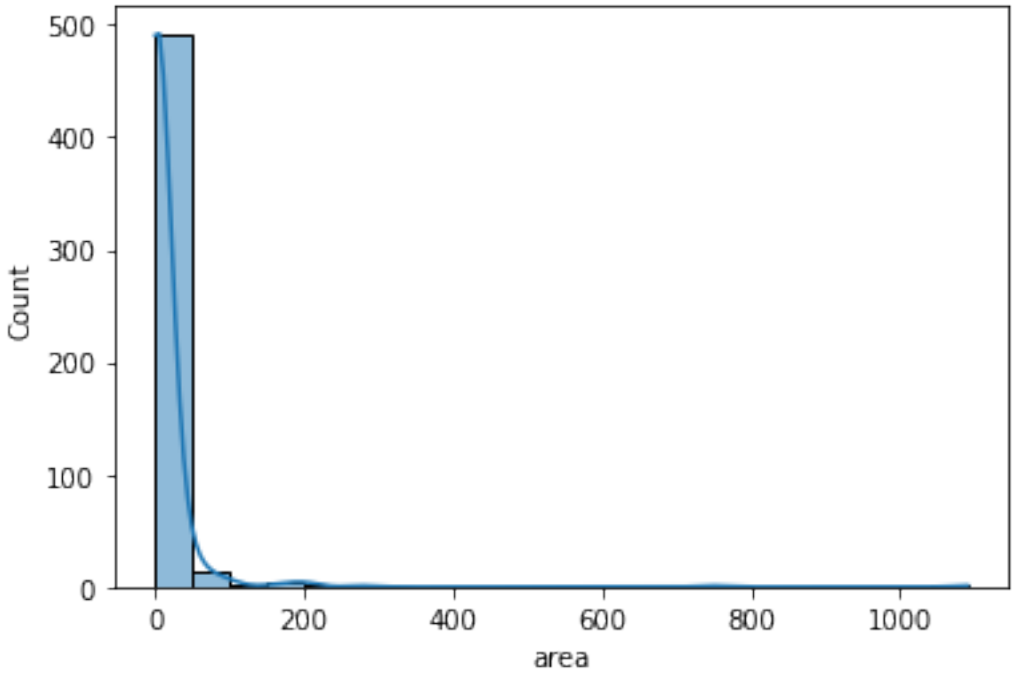
[6]:
```
Index(['X', 'Y', 'month', 'day', 'FFMC', 'DMC', 'DC', 'ISI', 'temp', 'RH',
       'wind', 'rain', 'area'],
      dtype='object')
```

1.3 Analiza zmiennej objaśnianej - area

[7]:
```
df['area'].describe()
```

[7]:
```
count     517.000000
mean       12.847292
std        63.655818
min         0.000000
25%         0.000000
50%         0.520000
75%         6.570000
max      1090.840000
Name: area, dtype: float64
```

[8]:
```
sns.histplot(df['area'], bins=round(1+3.322*np.log(df['area'].shape[0])),␣
↪kde=True)
plt.show()
```



Większość obserwacji jest bliska 0. Rozkład prawostronny.

[9]:
```
print("Skośność: %f" % df['area'].skew())
print("Kurtoza: %f" % df['area'].kurt())
```

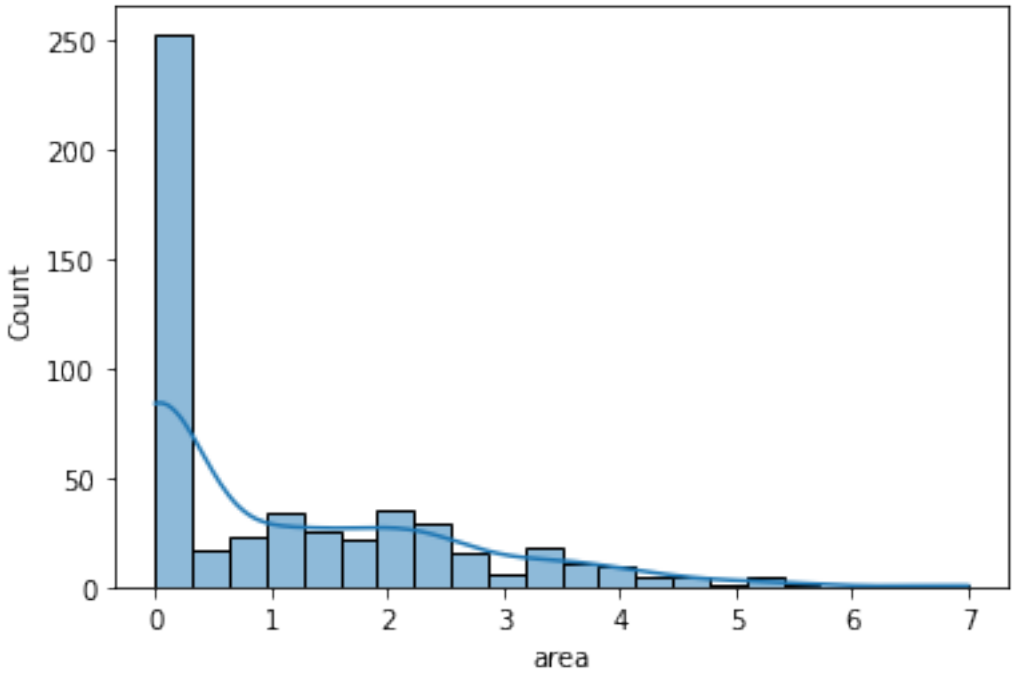
```
Skośność: 12.846934
Kurtoza: 194.140721
```

Potwierdza to obserwacje o prawostronnym i stromym rozkładzie.

1.3.1 Transformacja area

Na stronie zbioru danych zasugerowano aby przetransformować zmienną logarytmem. Sprawdźmy. Zastosujemy x -> log(x+1) aby uniknąć problemów z nieskończonością.

[10]:
```
sns.histplot(df['area'].apply(lambda x: np.log(x+1)), bins=round(1+3.322*np.
↪log(df['area'].shape[0])), kde=True)
plt.show()
```



Dodajmy tak przetransformowaną zmienną do zbioru danych. Sprawdzimy jak wygląda na wykresach zestawiona z innymi cechami.

[11]:
```
df['log_area'] = df['area'].apply(lambda x: np.log(x+1))
df.head()
```

[11]:

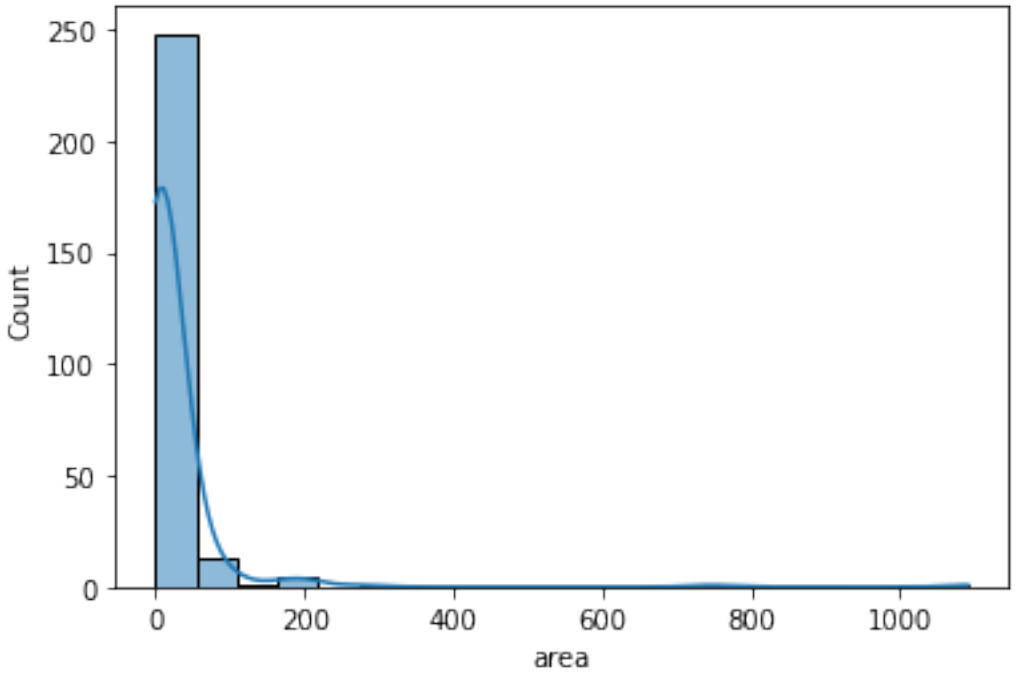
	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	log_area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51.0	6.7	0.0	0.0	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33.0	0.9	0.0	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33.0	1.3	0.0	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97.0	4.0	0.2	0.0	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99.0	1.8	0.0	0.0	0.0

Zobaczmy jak wyglądają dodatnie wartości `area`

[12]:
```
len(df[df['area']>0])
```

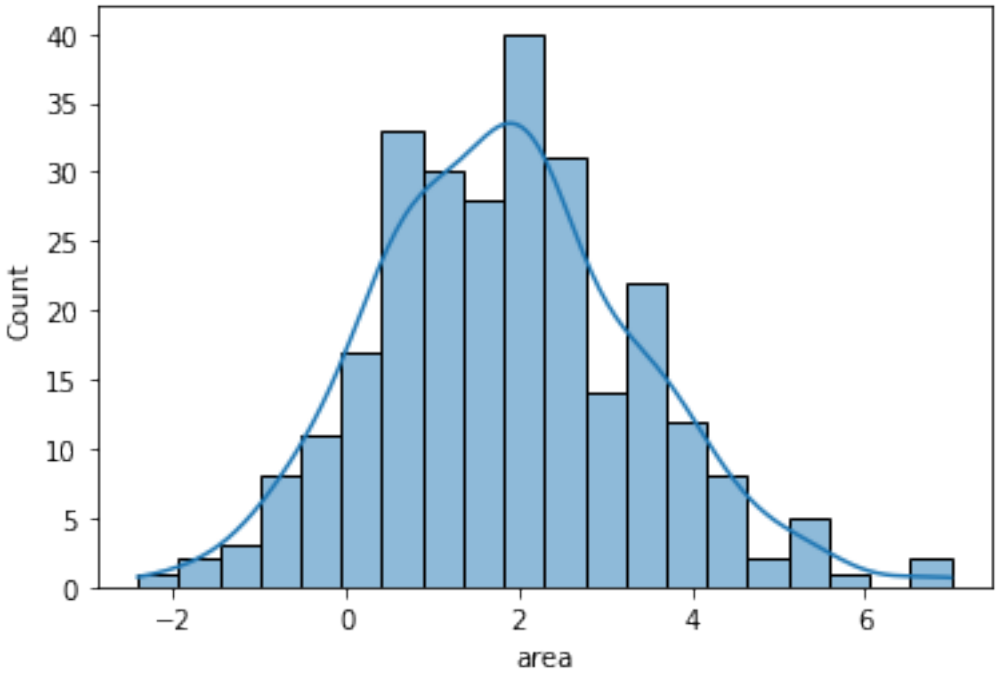
[12]: 270

[13]:
```
sns.histplot(df.loc[df['area']>0, 'area'], bins=round(1+3.322*np.
 ↪log(df[df['area']>0].shape[0])), kde=True)
plt.show()
```



Teraz możemy zastosować zwykły logarytm.

[14]:
```
sns.histplot(df.loc[df['area']>0, 'area'].apply(np.log), bins=round(1+3.322*np.
 ↪log(df[df['area']>0].shape[0])), kde=True)
plt.show()
```



Rozkład bardziej przypomina rozkład normalny. Można rozważyć podzielenie zadania na 2 części. Najpierw klasyfikujemy czy pożar wybuchnie (`area` > 0), a później przybliżamy log(`area`) jakimś modelem.

1.4 Utworzenie kolumn numerycznych kodujących dni tygodnia i miesiące

Dzięki temu będziemy mogli wyliczyć miary statystyczne i zobczyć histogramy.

- dni tygodnia: 1=poniedziałek, … , 7=niedziela
- miesiące: 1=styczeń, … , 12=grudzień

[15]:
```
df['month_num'] = pd.to_datetime(df.month, format='%b').dt.month
weekdays = {'mon':1, 'tue':2, 'wed':3, 'thu':4, 'fri':5, 'sat':6, 'sun':7}
df['day_num'] = df['day'].map(weekdays)
df.head()
```

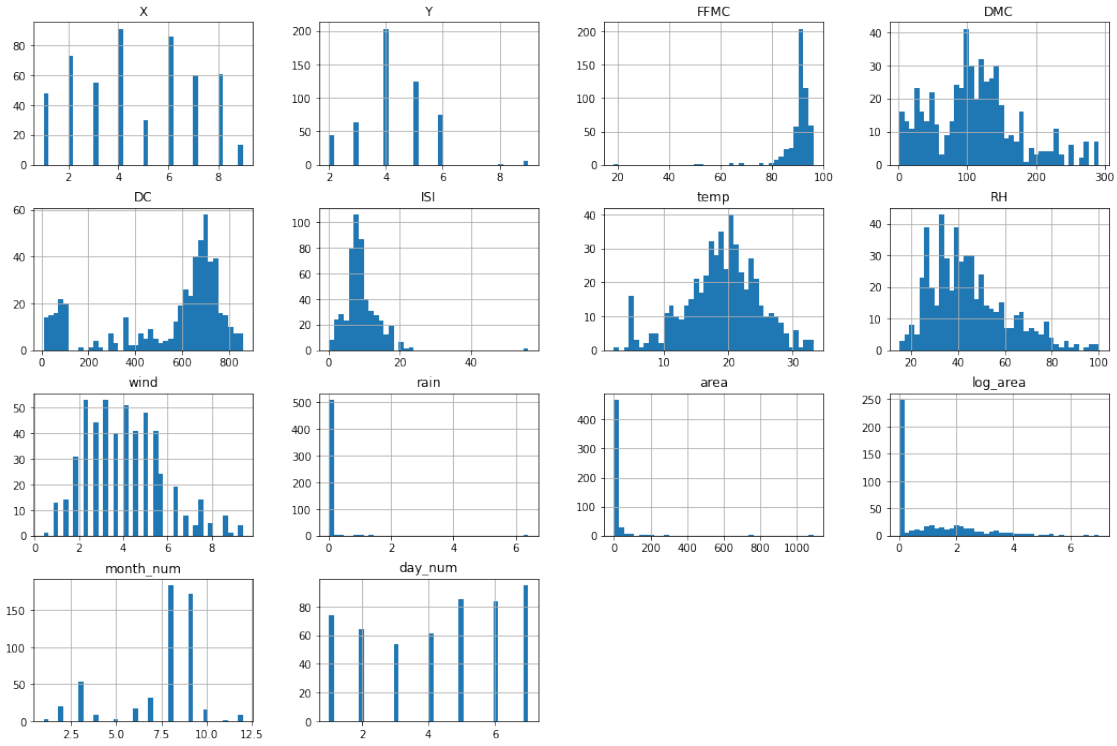
[15]:

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	log_area	month_num	day_num
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51.0	6.7	0.0	0.0	0.0	3	5
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33.0	0.9	0.0	0.0	0.0	10	2
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33.0	1.3	0.0	0.0	0.0	10	6
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97.0	4.0	0.2	0.0	0.0	3	5
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99.0	1.8	0.0	0.0	0.0	3	7

```
2       0.0          10         6
3       0.0           3         5
4       0.0           3         7
```

1.5 Histogramy zmiennych

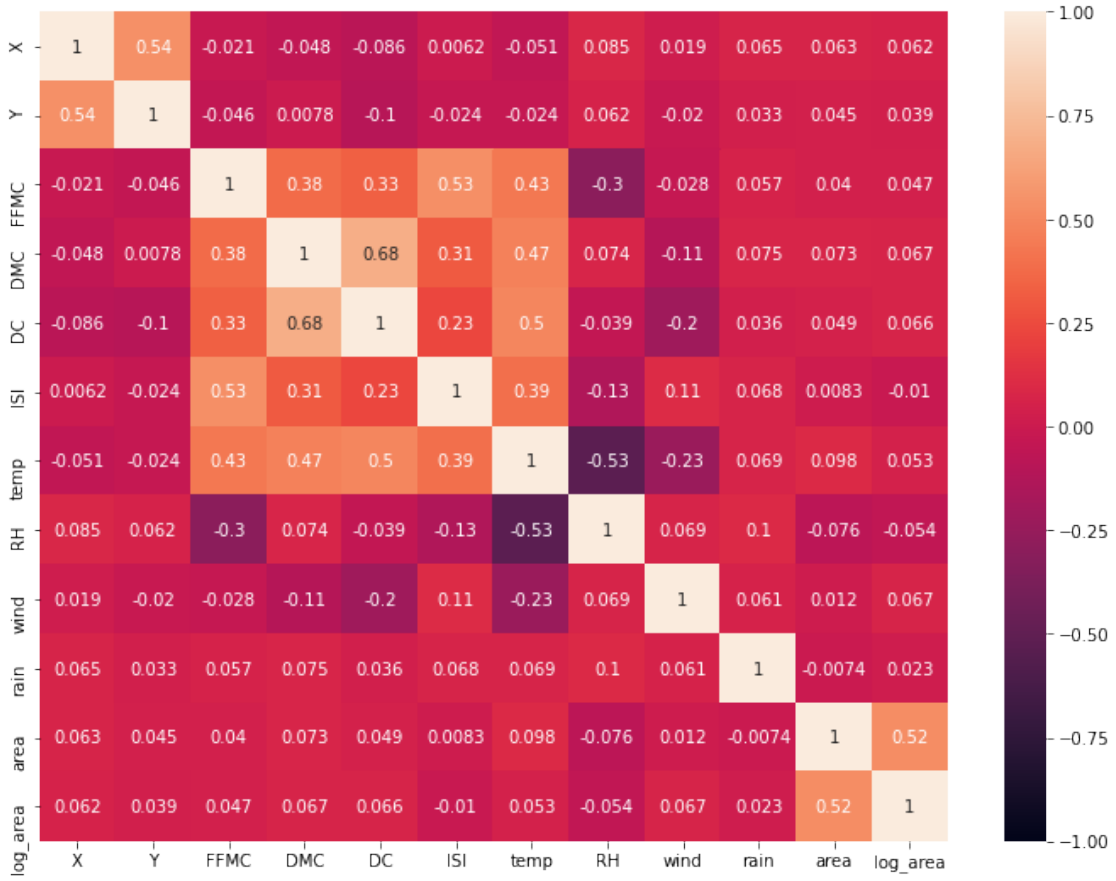
[16]:
```
df.hist(bins = 40, figsize = (18,12))
plt.show()
```



- `rain` bardzo dużo obserwacji ma wartość bliską 0. Może warto usunąć tę kolumnę.

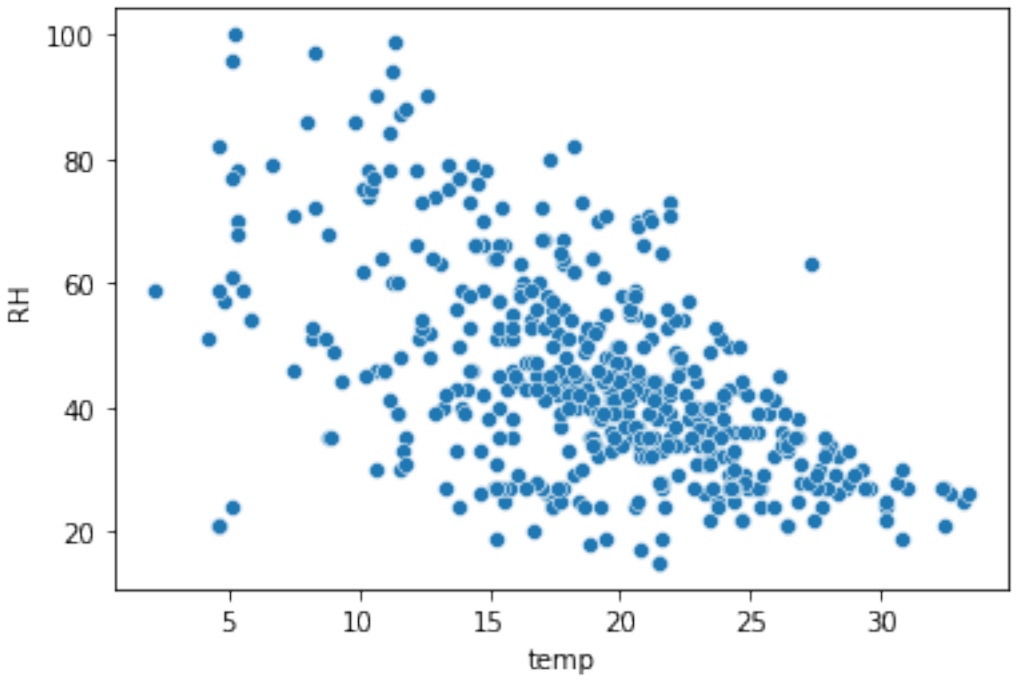
1.6 Korelacje cech

[17]:
```
corr = df.drop(['month_num', 'day_num'], axis = 1).corr()
_, __ = plt.subplots(figsize=(12,9))
sns.heatmap(corr, vmin=-1,annot=True)
plt.show()
```



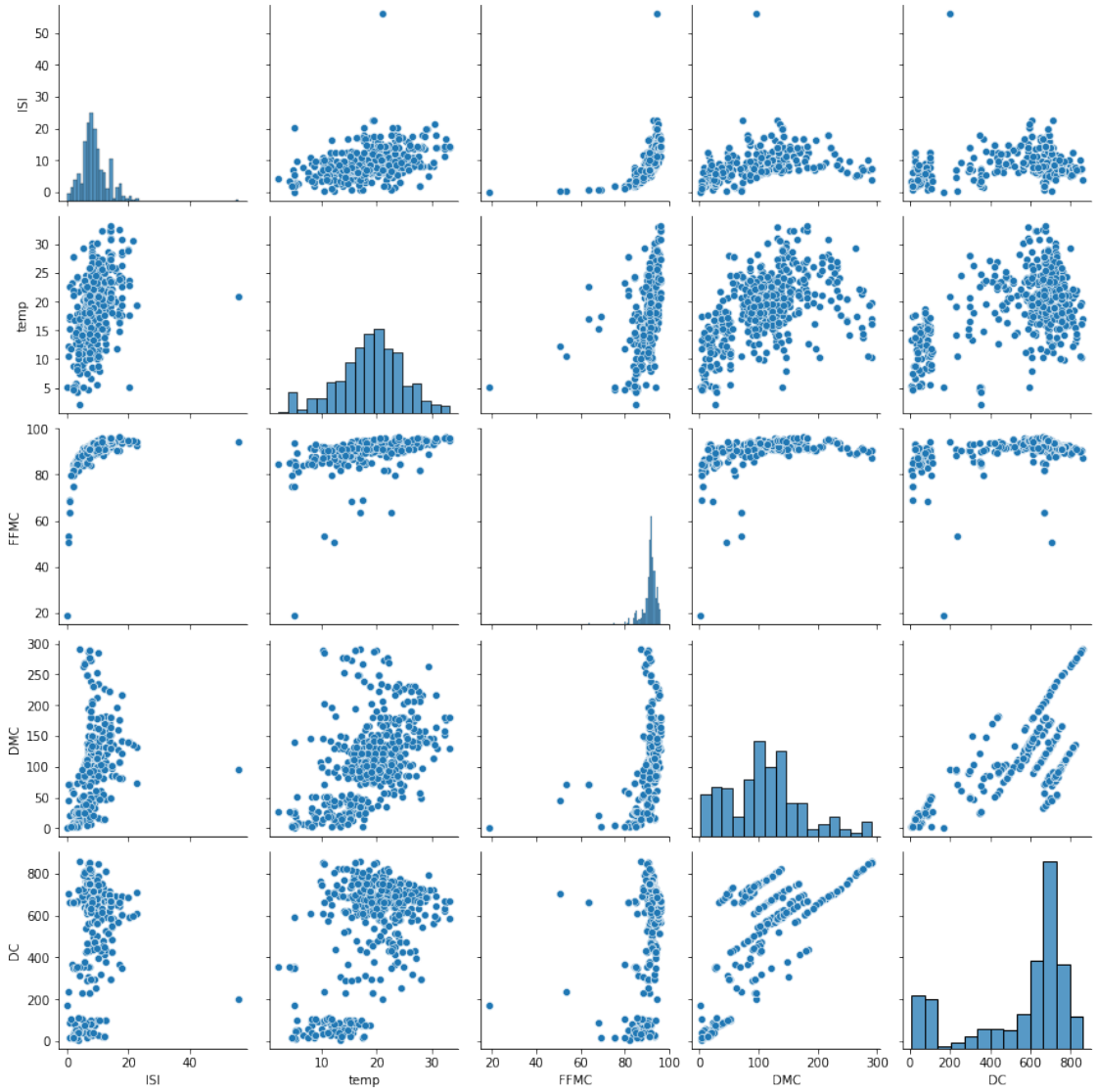
- `temp` odwrotnie skorelowany z `RH`. Im cieplej tym względna wilgotność niższa.

[18]:
```
sns.scatterplot(x = df['temp'], y = df['RH'])
plt.show()
```



- wzajemnie skorelowane `ISI`, `temp`, `FFMC`, `DMC`, `DC`.

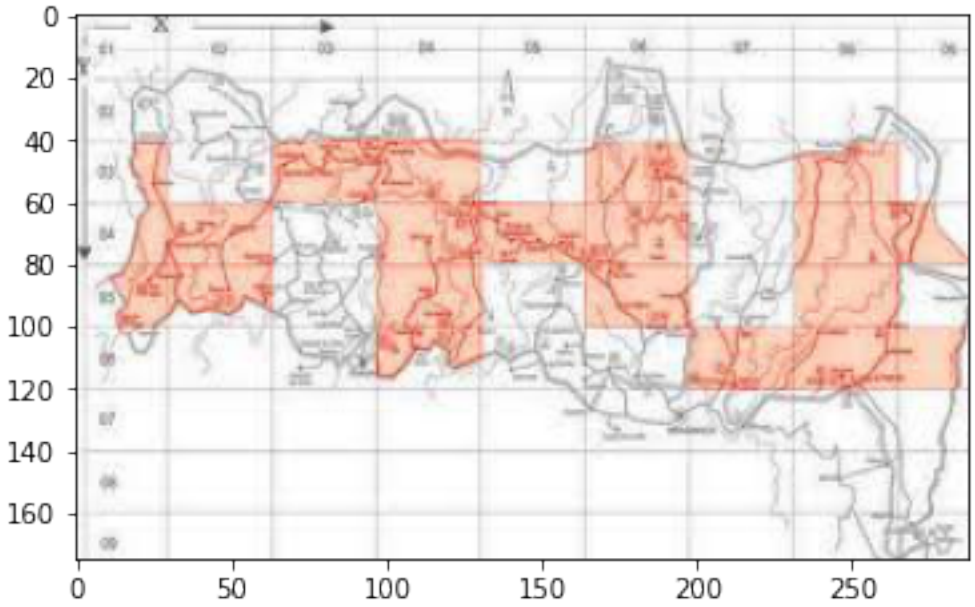
[19]:
```
cols = ['ISI', 'temp', 'FFMC', 'DMC', 'DC']
sns.pairplot(df[cols])
plt.show()
```



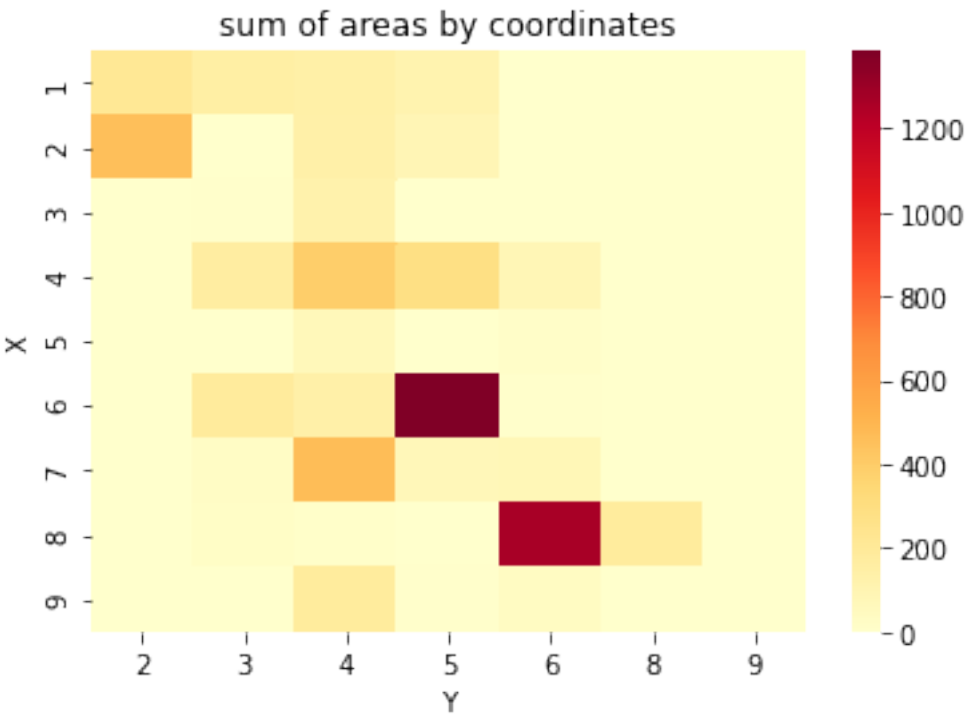
1.7 Współrzędne geograficzne

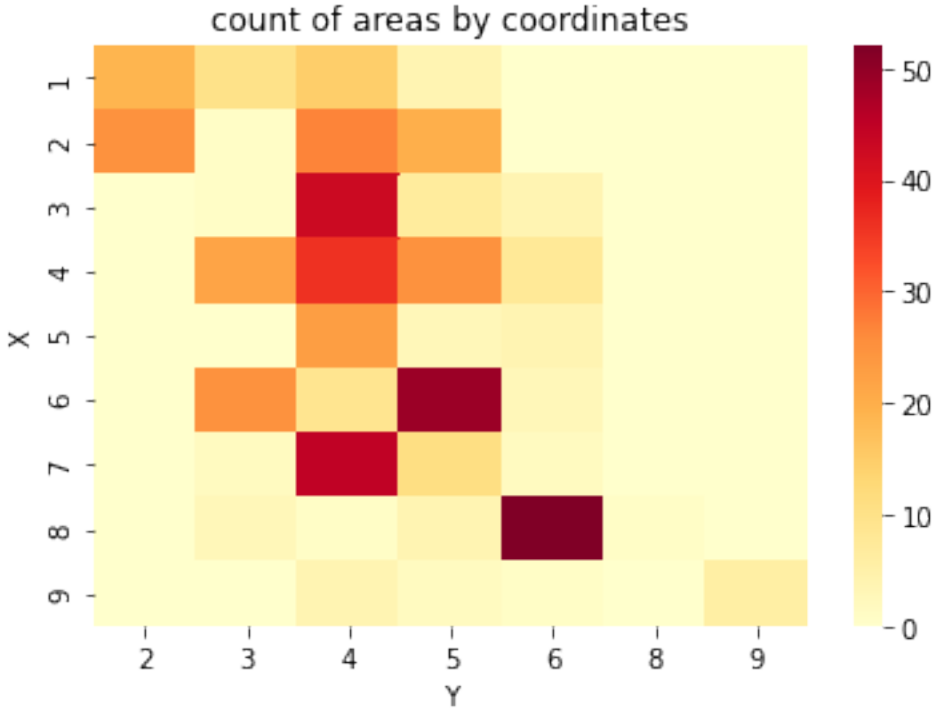
Mapa przedstawiająca podział parku na sektory

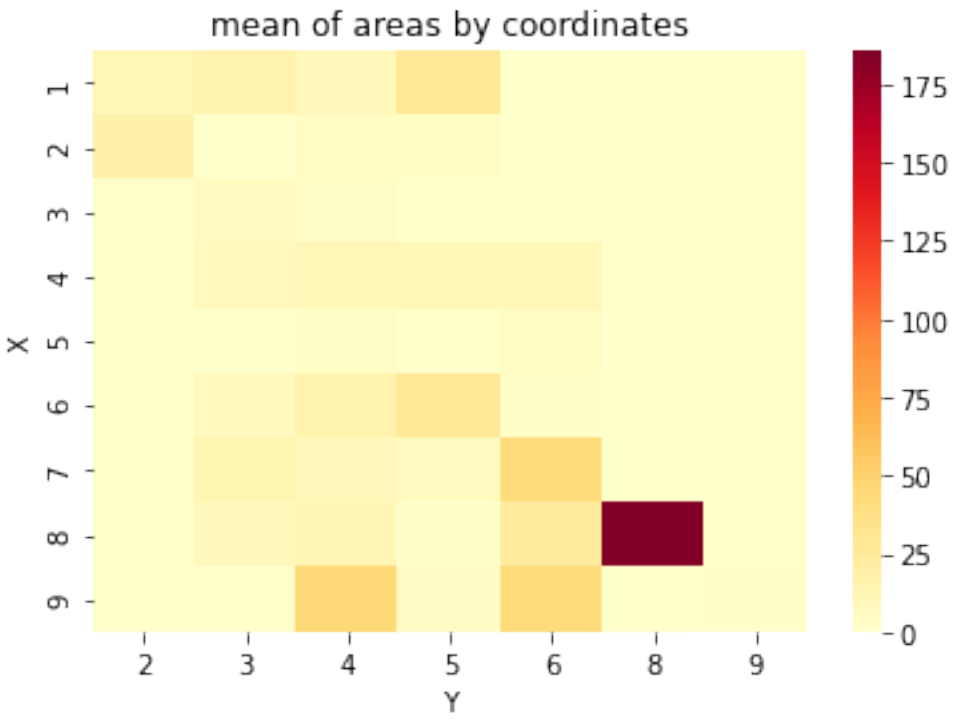
[28]:
```
import matplotlib.image as mpimg
image = mpimg.imread("images/download.jpeg")
plt.imshow(image)
plt.show()
```

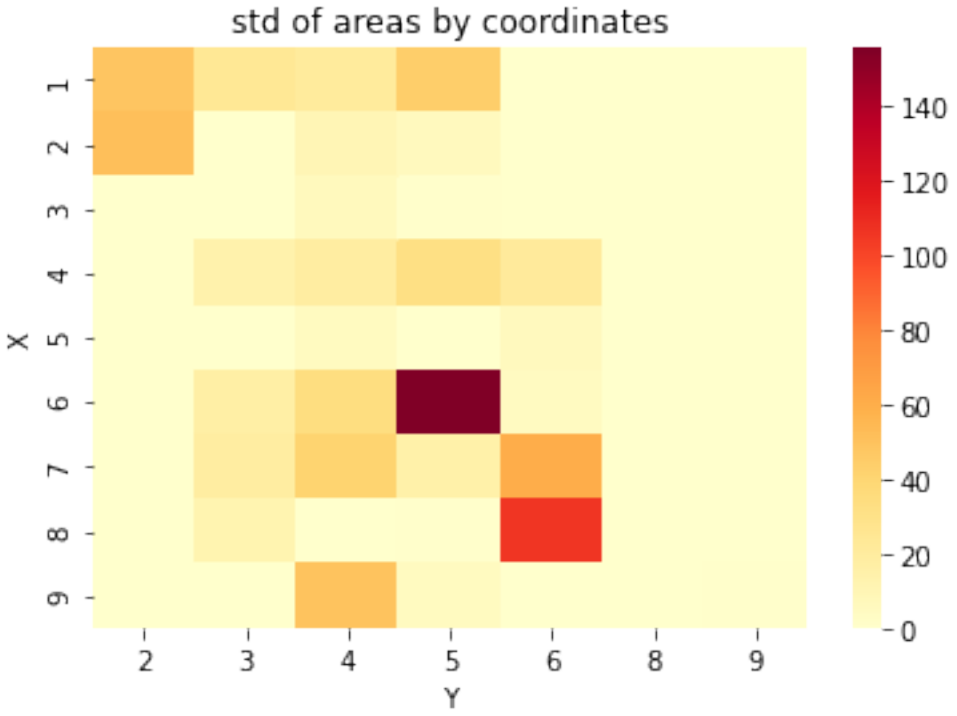


[20]:
```
def create_heatmap(agg_fun):
    geo_df = df.loc[:,['X','Y','area']].groupby(['X', 'Y'], as_index = False).
 ↪agg(agg_fun)
    geo_df_pivot = geo_df.pivot(index = 'X', columns = 'Y', values = 'area')
    geo_df_pivot[geo_df_pivot.isna()] = 0
    sns.heatmap(geo_df_pivot, cmap = 'YlOrRd').set_title(agg_fun + ' of areas␣
 ↪by coordinates')
    plt.show()

create_heatmap('sum')
create_heatmap('count')
create_heatmap('mean')
create_heatmap('std')
```


sum of areas by coordinates
X
1
2
3
4
5
6
7
8
9
Y
2
3
4
5
6
8
9
1200
1000
800
600
400
200
0


count of areas by coordinates
X
1
2
3
4
5
6
7
8
9
Y
2
3
4
5
6
8
9
50
40
30
20
10
0


mean of areas by coordinates
X
1
2
3
4
5
6
7
8
9
Y
2
3
4
5
6
8
9
175
150
125
100
75
50
25
0

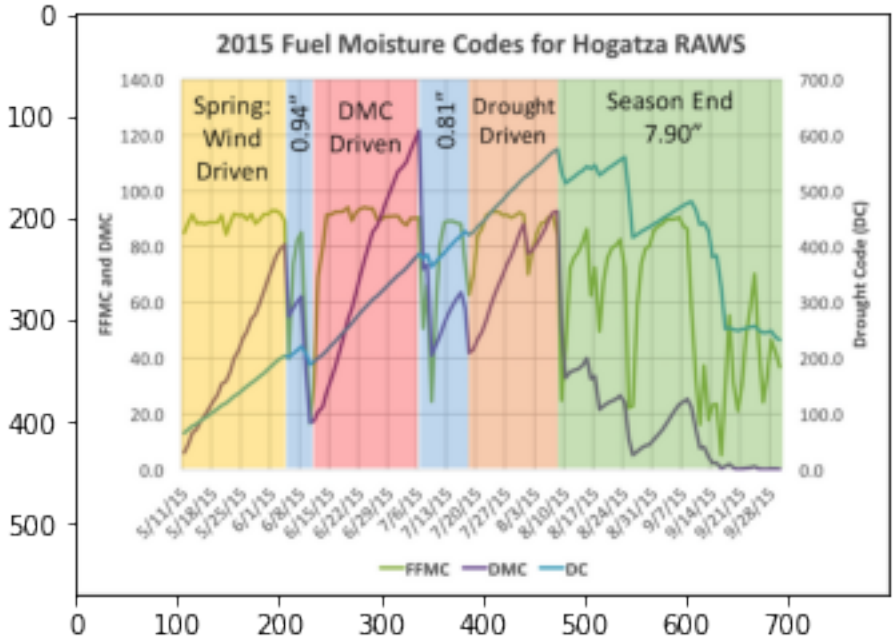

std of areas by coordinates
X
1
2
3
4
5
6
7
8
9
Y
2
3
4
5
6
8
9
140
120
100
80
60
40
20
0

Widać, że największe i najczęstsze pożary są w prostokątnym pasie leżącym wzdłuż przekątnej terenu. Stąd też współczynnik korelacji `X` i `Y` to 0.54. Największa średnia powierzchnia pożaru jest w prostokącie (8,8), gdzie odnotowano mniej niż 10 pożarów. Odchylenie standardowe spalonej powierzchni w tym rejonie też jest niewielkie. Sugeruje to, że było tam kilka dużych pożarów.

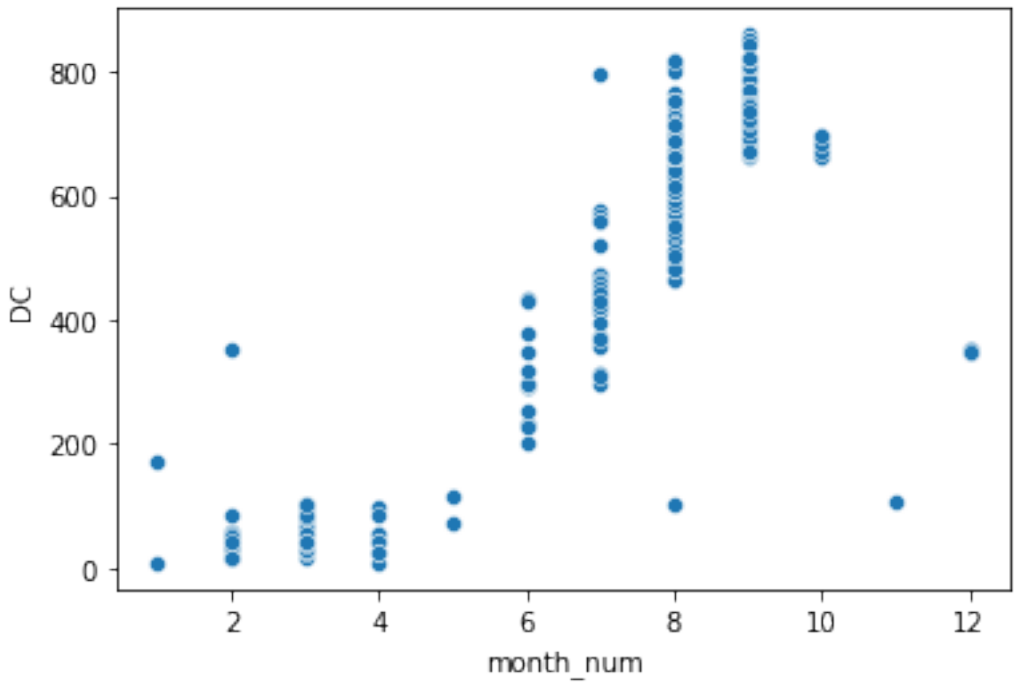
1.8 Dane w czasie

`DC` opisuje poziom suszy. Logiczne jest, że wraz z nadejściem lata ten wskaźnik rośnie

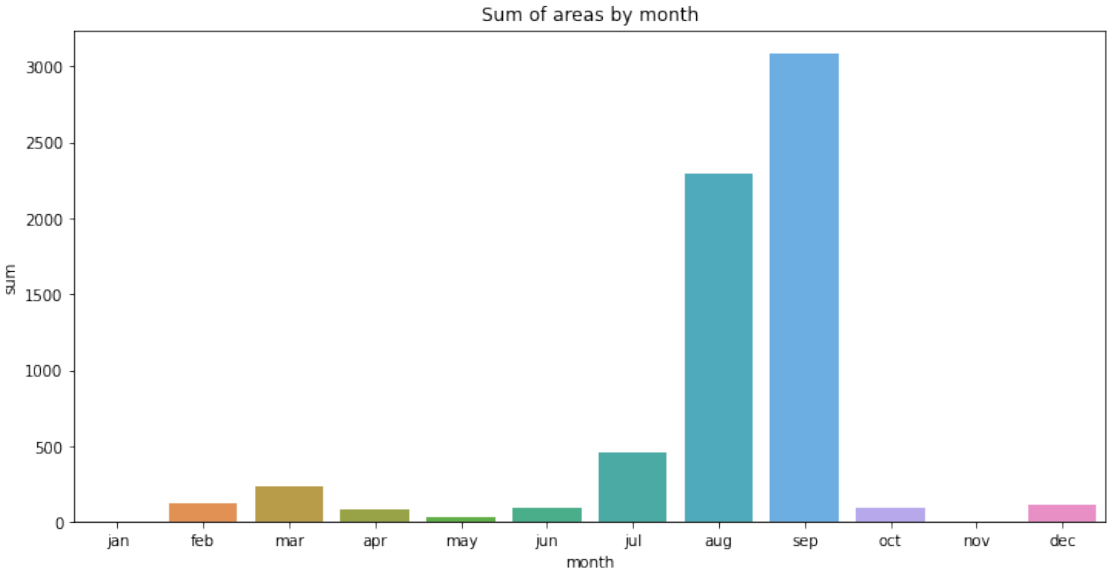
[27]:
```
import matplotlib.image as mpimg
image = mpimg.imread("images/437-cffdrs-fuel-moisture-codes-graph.png")
plt.imshow(image)
plt.show()
```

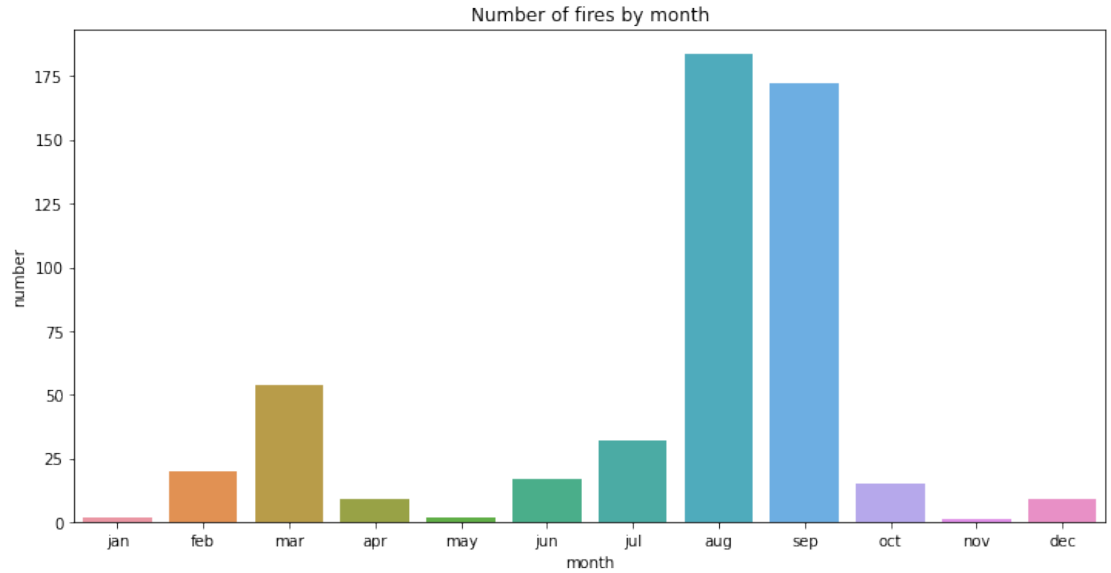


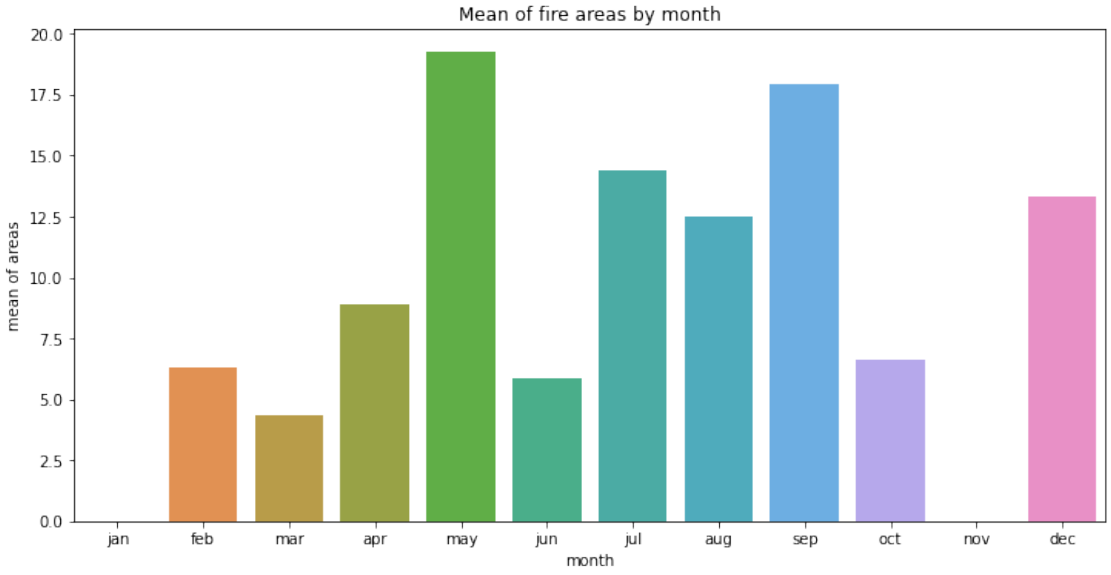
[21]:
```
sns.scatterplot(x = df['month_num'], y = df['DC'])
plt.show()
```



[22]:
```
month_df = df.loc[:,['month_num', 'month', 'area']].groupby(['month_num',␣
↪'month']).agg([np.sum, np.size, np.mean]).reset_index()
month_df.loc[:,'area']

f, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(12, 20))

sns.barplot(x = month_df['month'], y = month_df.loc[:,'area']['sum'], ax = ax1)
ax1.title.set_text('Sum of areas by month')
ax2.set_ylabel('sum of areas')

sns.barplot(x = month_df['month'], y = month_df.loc[:,'area']['size'], ax = ax2)
ax2.title.set_text('Number of fires by month')
ax2.set_ylabel('number')

sns.barplot(x = month_df['month'], y = month_df.loc[:,'area']['mean'], ax = ax3)
ax3.title.set_text('Mean of fire areas by month')
ax3.set_ylabel('mean of areas')

plt.show()
```

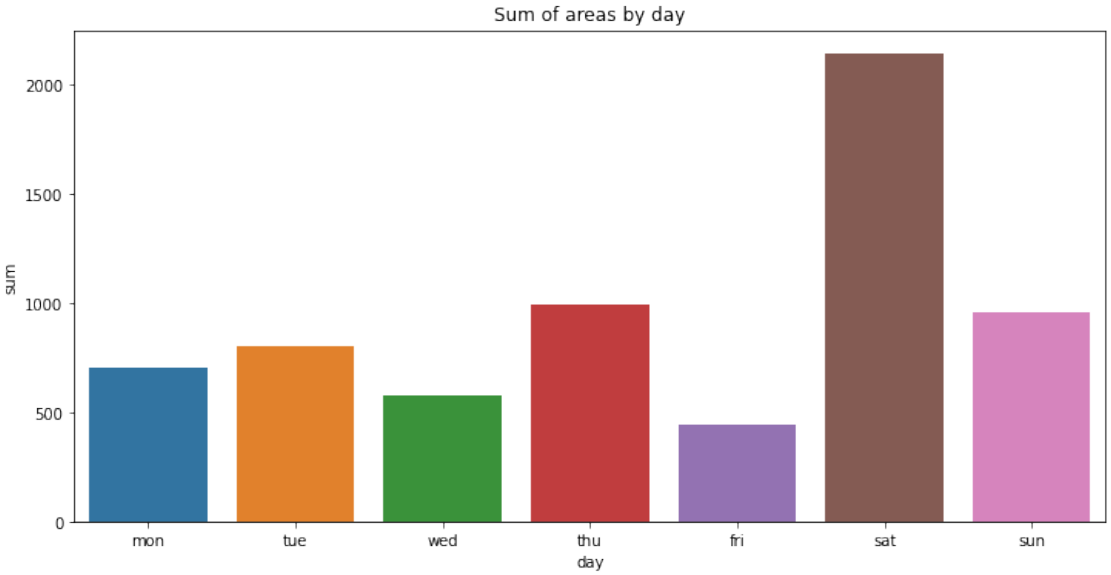

Sum of areas by month
sum
month
jan
feb
mar
apr
may
jun
jul
aug
sep
oct
nov
dec
0
500
1000
1500
2000
2500
3000

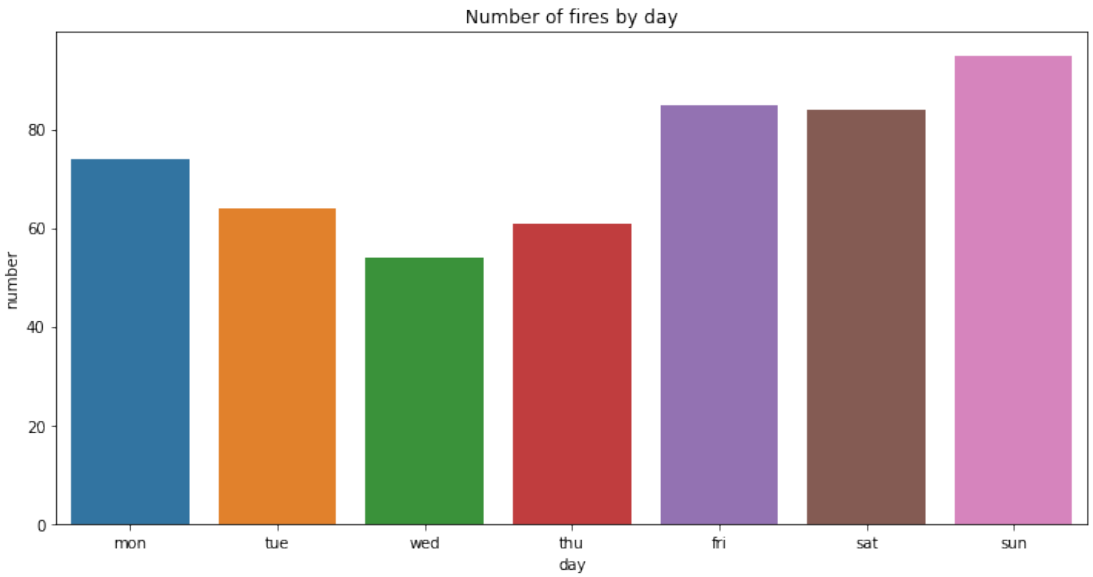

Number of fires by month
number
month
jan
feb
mar
apr
may
jun
jul
aug
sep
oct
nov
dec
0
25
50
75
100
125
150
175

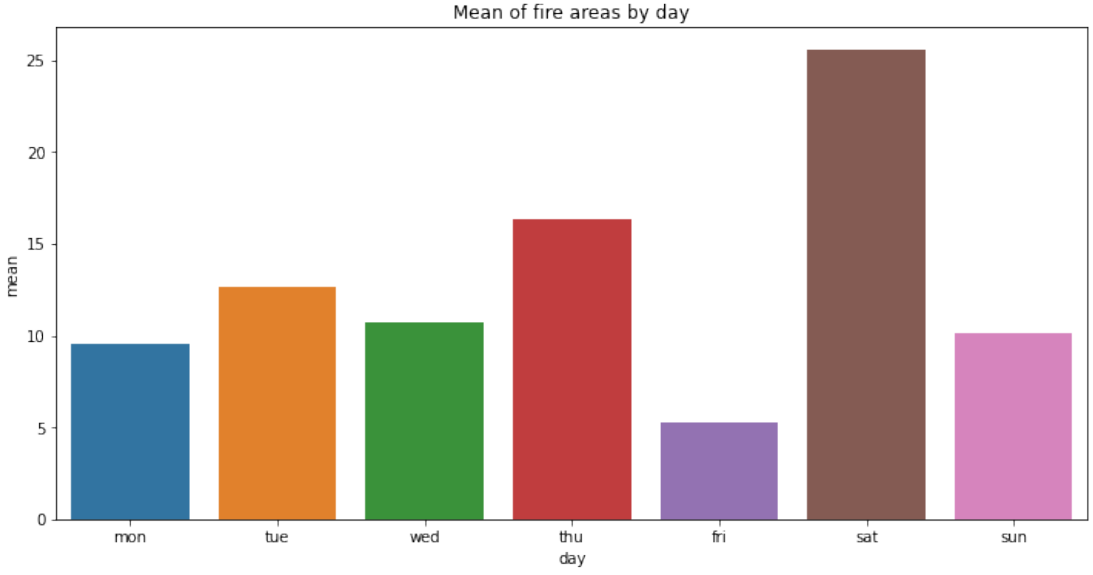

Mean of fire areas by month
mean of areas
month
jan
feb
mar
apr
may
jun
jul
aug
sep
oct
nov
dec
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

Nie jest zaskoczeniem, że późnym latem pożarów jest najwięcej i suma spalonych obszarów jest największa. Warto zauważyć, że średnia powierzchnia pożaru nie zależy aż tak bardzo od pory roku.

[23]:
```
weekday_df = df.loc[:,['day_num', 'day', 'area']].groupby(['day_num', 'day']).
 ↪agg([np.sum, np.size, np.mean]).reset_index()
weekday_df.loc[:,'area']

f, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(12, 20))

sns.barplot(x = weekday_df['day'], y = weekday_df.loc[:,'area']['sum'], ax =␣
 ↪ax1)
ax1.title.set_text('Sum of areas by day')
ax2.set_ylabel('sum of areas')

sns.barplot(x = weekday_df['day'], y = weekday_df.loc[:,'area']['size'], ax =␣
 ↪ax2)
ax2.title.set_text('Number of fires by day')
ax2.set_ylabel('number')


sns.barplot(x = weekday_df['day'], y = weekday_df.loc[:,'area']['mean'], ax =␣
 ↪ax3)
ax3.title.set_text('Mean of fire areas by day')
ax3.set_ylabel('mean')

plt.show()
```


Sum of areas by day
sum
2000
1500
1000
500
0
mon
tue
wed
thu
fri
sat
sun
day


Number of fires by day
number
80
60
40
20
0
mon
tue
wed
thu
fri
sat
sun
day


Mean of fire areas by day
mean
25
20
15
10
5
0
mon
tue
wed
thu
fri
sat
sun
day

Najwięcej pożarów wybucha w okolicach weekendów. Może to wynikać ze wzmożonej obecności turystów w dni wolne.

1.9 Narzędzie do automatycznej eksploracji - pandas profiler

[24]:
```
profile = ProfileReport(df, title='Pandas Profiling Report', explorative=True)
```

[25]:
```
profile.to_notebook_iframe()
```

```
Summarize dataset:   0%|          | 0/29 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]

<IPython.core.display.HTML object>
```

1.9.1 Wnioski z raportu

- są 4 zdublowane obserwacje. Prawdopodobnie jest to wynik błędu i lepiej je usunąć.

1.9.2 Opinia o narzędziu

- automatyzuje wstępny etap eksploracji
- wyłapuje anomalia w danych, takie jak dużo zerowych wartości, duplikaty
- po przeczytaniu raportu można zyskać intuicję co do dalszej analizy
- profiler nie zrobi za nas bardziej zaawansowanych wykresów i podsumowań