Untitled

March 6, 2021

1 HW 1 - Patryk Słowakiewcz

[173]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from matplotlib import axes as ax
import seaborn as sns
from scipy import stats
import datetime
import plotly.graph_objs as go
import plotly.express as px
import pandas_profiling
```

1.1 Wczytanie i wstępna eksploracja

[174]:
```
df = pd.read_csv("forest_fires_dataset.csv")
```

[175]:
```
df.head(10)
```

[175]:

	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
5	8	6	aug	sun	92.3	85.3	488.0	14.7	22.2	29.0	5.4	0.0	0.0
6	8	6	aug	mon	92.3	88.9	495.6	8.5	24.1	27.0	3.1	0.0	0.0
7	8	6	aug	mon	91.5	145.4	608.2	10.7	8.0	86.0	2.2	0.0	0.0
8	8	6	sep	tue	91.0	129.5	692.6	7.0	13.1	63.0	5.4	0.0	0.0
9	7	5	sep	sat	92.5	88.0	698.6	7.1	22.8	40.0	4.0	0.0	0.0

[176]:
```
#sprawdzam braki w danych
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
```

[177]:
```
# zamieniamy dni i miesiące na liczby

#from sklearn.preprocessing import LabelEncoder
#le = LabelEncoder()
#df['month_int'] = le.fit_transform(df['month']) + 1
#df['day_int'] = le.fit_transform(df['day']) + 1

# ale to niestety nie działa bo są nie pokolei
```

1.2 Zmiana dni i miesięcy na liczby

[181]:
```
def day_change(weekday):
    if weekday == 'mon':
        return 1
    if weekday == 'tue':
        return 2
    if weekday == 'wed':
        return 3
    if weekday == 'thu':
        return 4
    if weekday == 'fri':
        return 5
    if weekday == 'sat':
        return 6
    if weekday == 'sun':
        return 7
```

[182]:
```
m_list = [datetime.datetime.strptime(i, '%b') for i in df['month']]
df['month_int'] = [i.month for i in m_list]
```

```
df['day_int'] = [day_change(i) for i in df['day']]
```

[183]:
```
df.describe()
```

[183]:

	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

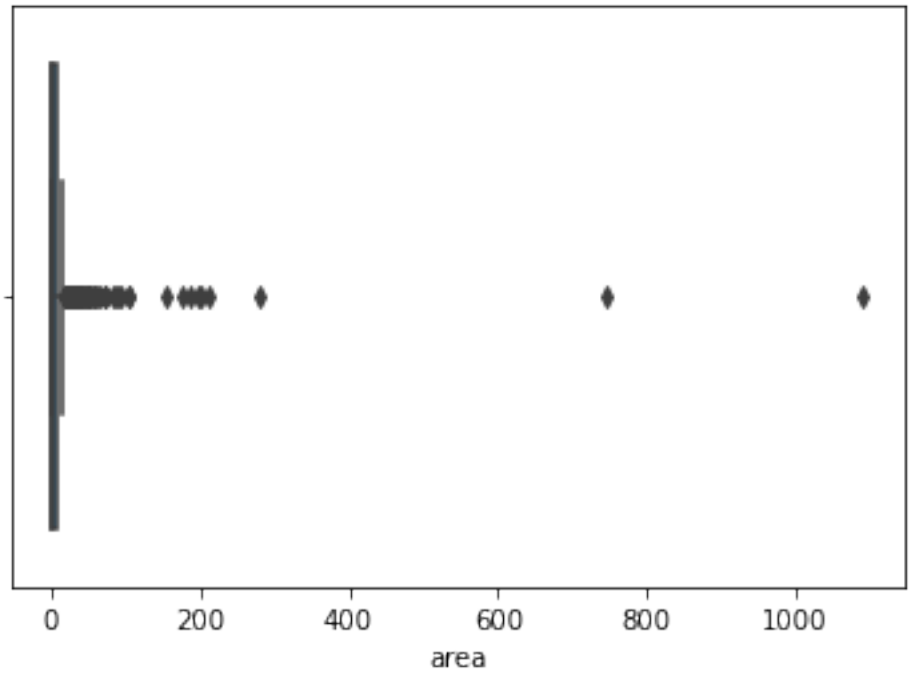
	temp	RH	wind	rain	area
count	517.000000	517.000000	517.000000	517.000000	517.000000
mean	18.889168	44.288201	4.017602	0.021663	12.847292
std	5.806625	16.317469	1.791653	0.295959	63.655818
min	2.200000	15.000000	0.400000	0.000000	0.000000
25%	15.500000	33.000000	2.700000	0.000000	0.000000
50%	19.300000	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

	month_int	day_int
count	517.000000	517.000000
mean	7.475822	4.259188
std	2.275990	2.072929
min	1.000000	1.000000
25%	7.000000	2.000000
50%	8.000000	5.000000
75%	9.000000	6.000000
max	12.000000	7.000000

1.3 Badanie targetu (area)

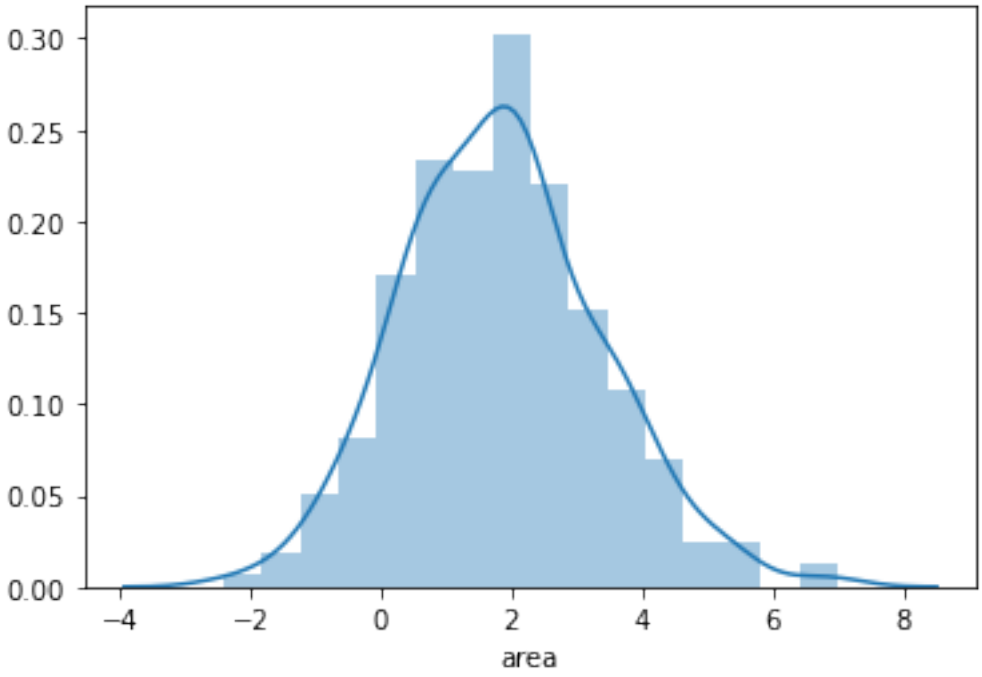
[184]:
```
#Spardzam rozkład naszego y którym jest 'area'
sns.boxplot(df['area'])
# widzimy bardzo duże odstępstwa od mediany na prawo (jest to spowodowane dużą␣
 ↪ilością zartości 0 i bliskich)
```

[184]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x26b34f6e940>
```



[185]:
```
# po usunięciu wartości 0 i zastosowaniu skali logarytmicznej otrzymujemy␣
↪rozkład bliski rozkładowi normalnemu
cdf2 = np.isclose(df['area'], 0)
cdf = np.logical_not(cdf2)
sns.distplot(np.log(df[cdf]['area']))
```

[185]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x26b35001e20>
```

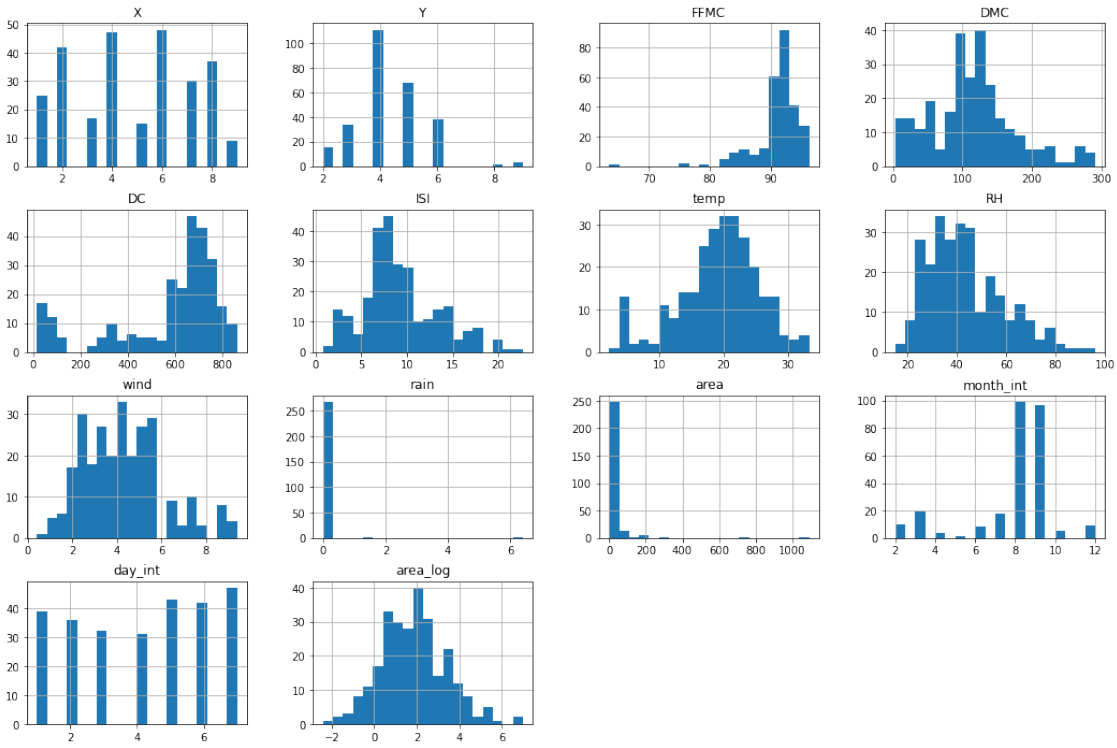


[186]:
```
#dodajemy kolumnę z wartościamu log(df('area')) aby łatwiej ją było analizować
↪( dla 0 jest None)
l = [np.log(i) if i != 0 else None for i in df['area']]
df['area_log'] = l
```

1.3.1 Histogramy dla każdej zmiennej przy czym bierzemy obserwacje jedynie dla przypadków pożarów.

Widzimy z poniższych wykresów, że ['DC', 'DMC', 'FFMC', 'ISI', 'RH', 'temp', 'wind'], mogą być w pewien sposób zależne co wynika również ze strony kanadyjskiego leśnictwa poniważ wskaźniki te są wyliczane na podstawie warunków atmosferycznych (https://cwfis.cfs.nrcan.gc.ca/background/summary/fwi).

[187]:
```
df[cdf].hist(bins= 20,figsize=(18,12))
plt.show()
```



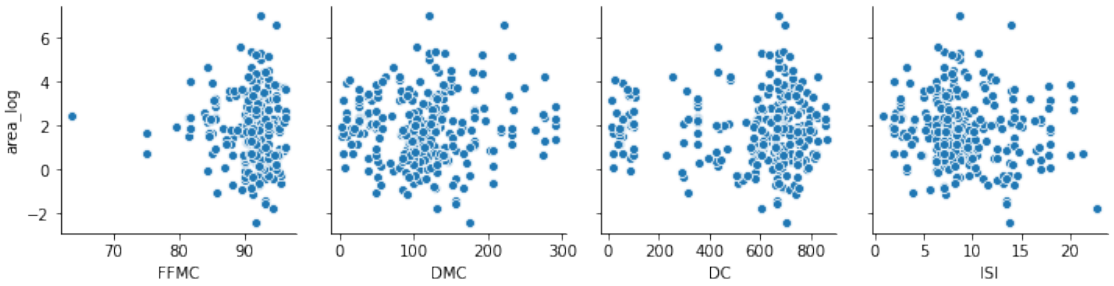
1.3.2 Pairplot

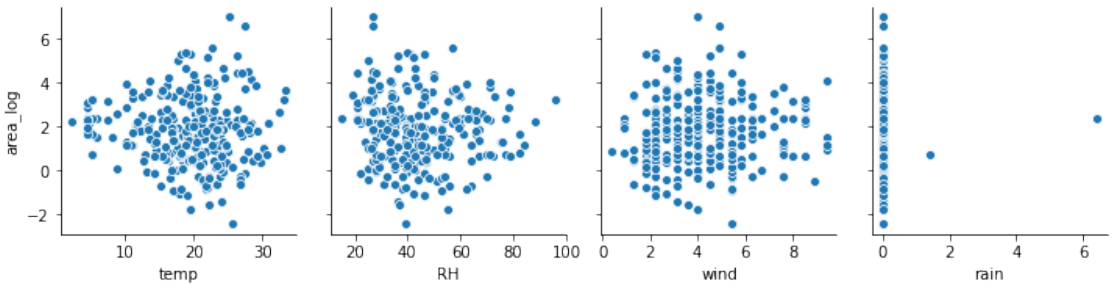
Sprawdzamy możliwe korelacje pomiędzy wielkością pożarów a innymi zmiennymi.

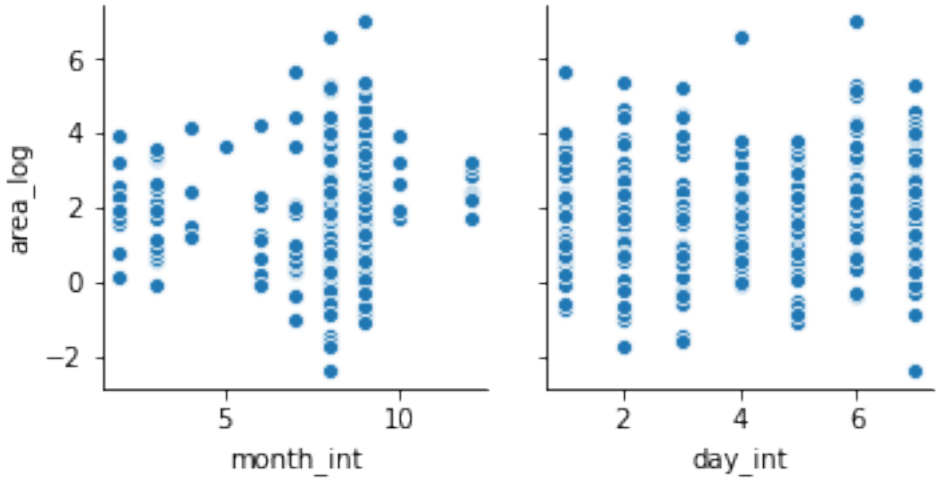
Nie widać, żadnych liniowych zależności ale można wyciągnąć wnioski, że pożary pojawiają się kiedy ilość opdaów jest zerowa co jest dość intuicyjne.

[188]:
```
sns.pairplot(df[cdf], y_vars="area_log", x_vars=df.columns.values[4:8])
sns.pairplot(df[cdf], y_vars="area_log", x_vars=df.columns.values[8:12])
sns.pairplot(df[cdf], y_vars="area_log", x_vars=df.columns.values[13:15])

plt.show()
```





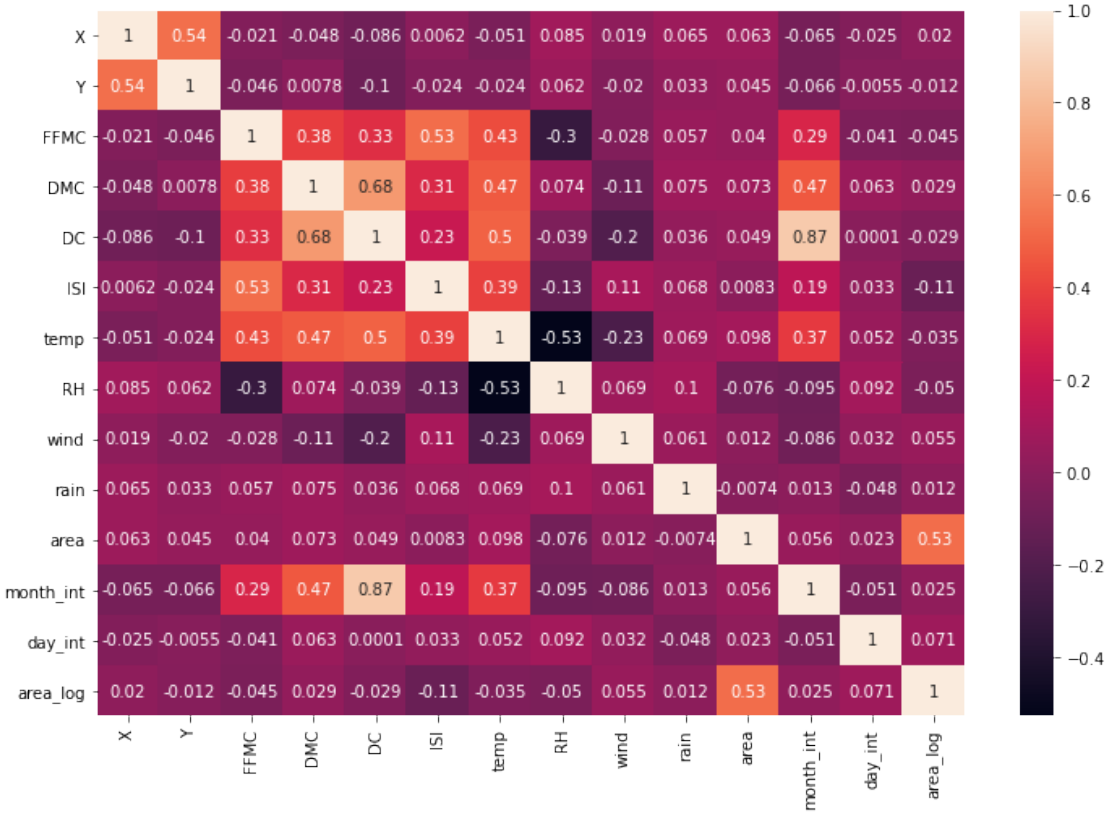


1.3.3 Heatmapa korelacji

Widzimy, że podane w ramce wskaźniki są ze sobą trochę skorelowane (w szczególności DC oraz DMC) ciekawa jest również wysoka korelacja DC z miesiącami pomimo, że data nie ma bezpośredniego wpływu na wyliczenia DC

[189]:
```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), annot=True)

plt.show()
```



1.3.4 Umiejscownie pożarów

Sprawdzimy jak rozkładaja się średnia pożarów z zależności od zmiennych X oraz Y

[190]:
```
#
fire_matrix = np.zeros((10,11))
for i in range(1,len(df)):
    k = df['area'][i]
    if k != 0:
        k = np.log(k)
    fire_matrix[df.iloc[i, 1], df.iloc[i, 0]] += k
```

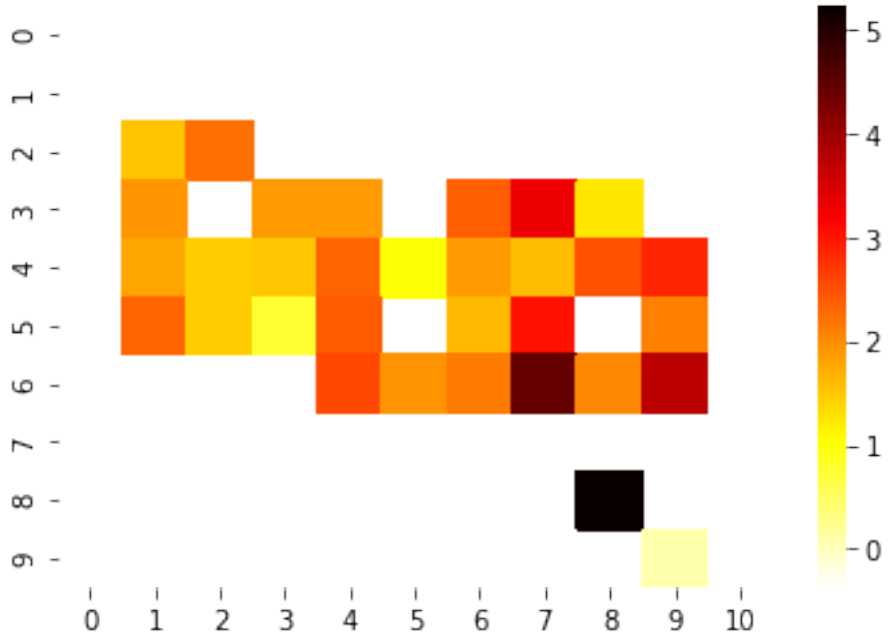
[191]:
```
freq_matrix = np.zeros((10,11))
for i in range(1,len(df)):
    k = df['area'][i]
    if k != 0:
        freq_matrix[df.iloc[i, 1], df.iloc[i, 0]] += 1
```

[192]:
```
sns.heatmap((fire_matrix/freq_matrix), cmap='hot_r')
```

```
<ipython-input-192-79840e7e671c>:1: RuntimeWarning:
```

```
invalid value encountered in true_divide
```

[192]: `<matplotlib.axes._subplots.AxesSubplot at 0x26b30369730>`



Widać pewną reguralność wzdłuż na powyższej 'mapie'. Pożary rozkładają się wzdłuż Y = 4

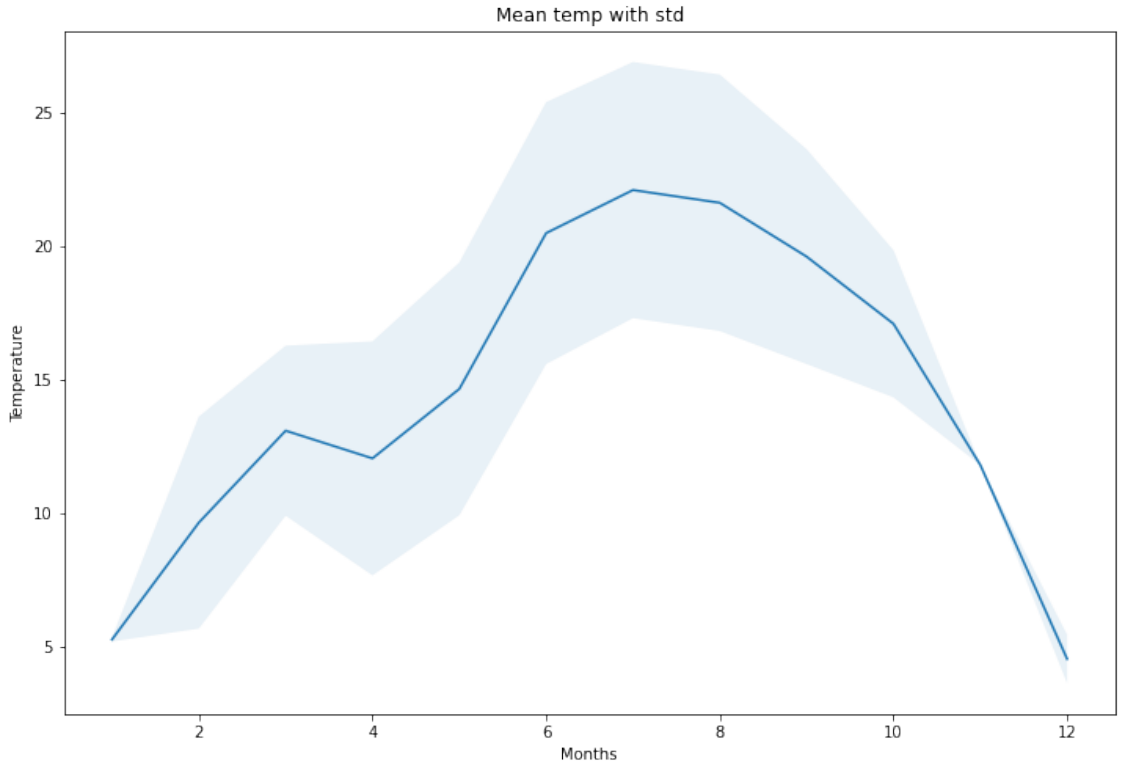
1.4 Rozkład średniej temperatury według miesięcy wraz z odchyleniem standardowm. Poniżej widać, wzrost w miesiądach letnich co jest spodziewaną zależnością

[193]:
```
months = df.groupby(['month_int']).mean()
months_std =  df.groupby(['month_int']).std()
months_std.loc[11] = 0
months_max =  df.groupby(['month_int']).max()
months_min =  df.groupby(['month_int']).min()
```

[194]:
```
x = months.index
y = months['temp']
y_upper = y + months_std['temp']
y_lower = y - months_std['temp']

plt.figure(figsize=(12,8))
plt.plot(x, y)
plt.fill_between(x, y_upper, y_lower, alpha = 0.1)
plt.title("Mean temp with std")
```

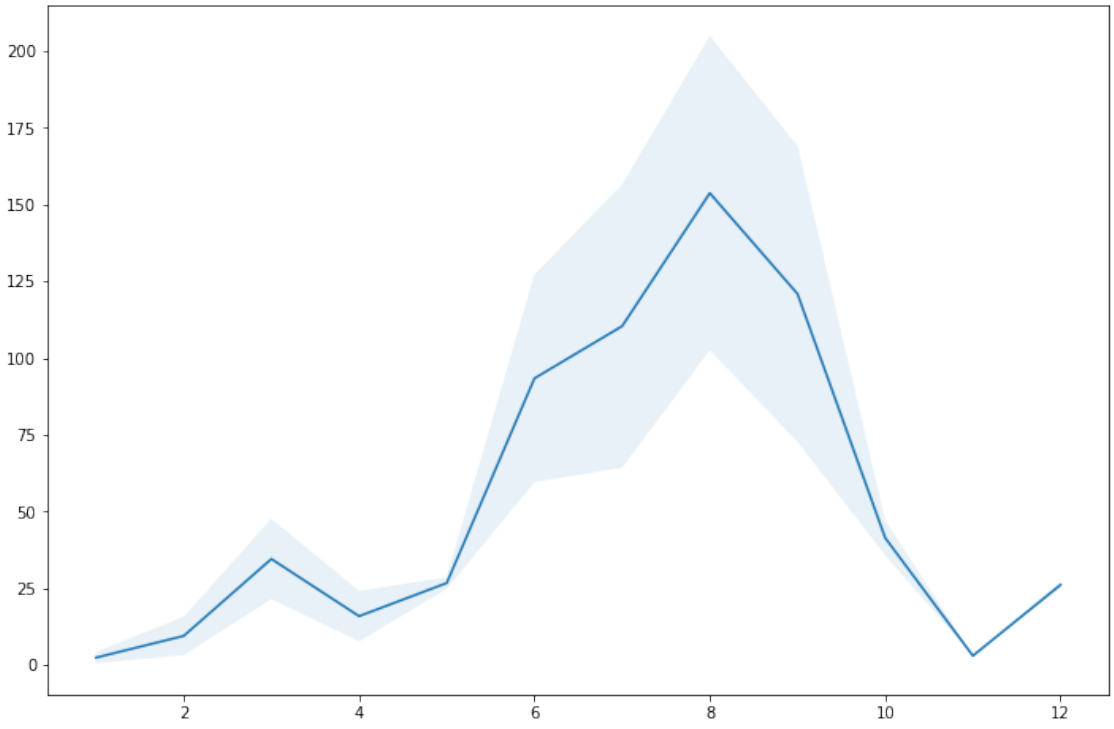
```
plt.xlabel("Months")
plt.ylabel('Temperature')
plt.show()
```



[195]:
```
what = 'DMC'
x = months.index
y = months[what]
y_upper = y + months_std[what]
y_lower = y - months_std[what]

plt.figure(figsize=(12,8))
plt.plot(x, y)
plt.fill_between(x, y_upper, y_lower, alpha = 0.1)

plt.show()
```

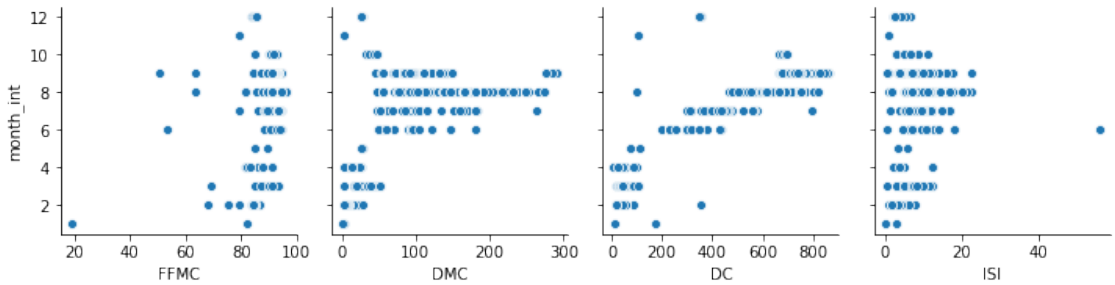


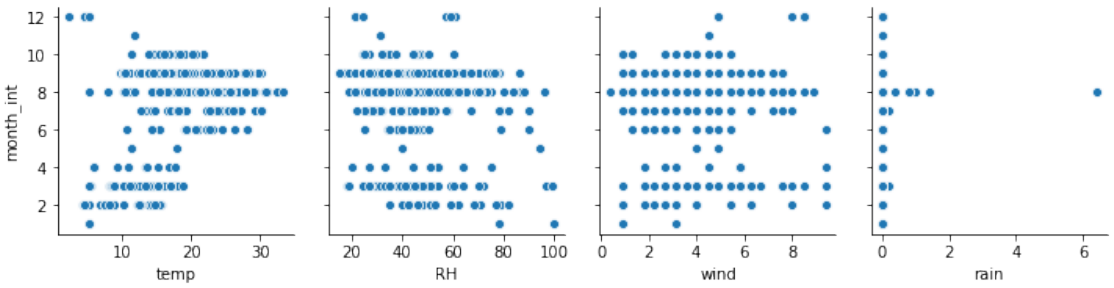
Badając te dwie zmienne można zauważyć, że obie rosną w okresie letnim. Warto zatem zbadać jak inne zmienne zachowują się na przestrzenii roku

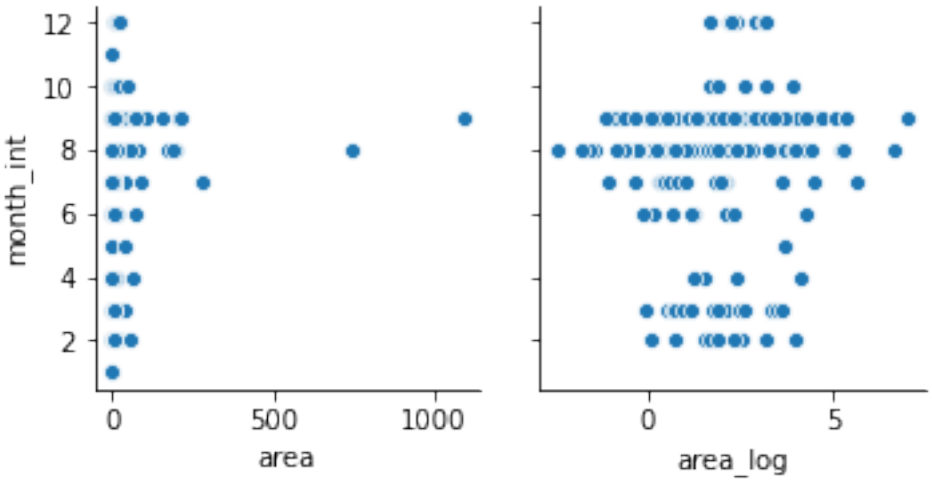
1.5 Pairploty ze zwzględu na miesiące

[196]:
```
sns.pairplot(df, y_vars=["month_int"], x_vars=df.columns.values[4:8])
sns.pairplot(df, y_vars='month_int', x_vars=df.columns.values[8:12])
sns.pairplot(df, y_vars='month_int', x_vars=df.columns.values[[12,15]])
```

[196]:
```
<seaborn.axisgrid.PairGrid at 0x26b18363b20>
```







Na powyższych wykresach widać jak zmienne DMC oraz temp zwiększają wartość w okresie letnim ale to już zaobserwowaliśmy na wcześniejszych wykresach. Ale dzięki tym pairplotom widzimy, że również DC oraz ISI są mocno zależne od miesiąca.

1.6 Python-profilig

Jest to dobre i wygodne narzędzie do szybkiej eksploracji danych dzięki któremu możemy zbadać rozkłady poszczególnych zmiennych. Dzięki heatmapie korelacji można też zauważyć które zmienne warto odrzucić. Wadami które mogą sprawiać problemy to brak możliwości wyboru jak gęste mają być histogramy pownieważ w pewnych przypadkach uniemożliwia to dopasowanie odpowiedniego rozkładu. Badane są wszystkie dane więc ręcznie trzeba wykonywać pewne przekształcenia np. zmienna 'area' jest mocno skoncentrowana przy 0 więc bez przekształceń nie możliwe jest zaobserwowanie jej właściwego rozkładu. Tak samo korelacja wyłapuje jedynie liniowe zależności, więc jeżeli chcemy badać bardziej złożone zależności trzeba to robić ręcznie.

[ ]:
```
df.profile_report()
```