WUM_PD1_Szymon_Recko

March 8, 2021

[2]:
```
import pandas as pd
import numpy as np
import csv
from matplotlib import pyplot as plt
import seaborn as sns
from scipy import stats
from pandas_profiling import ProfileReport
import warnings
warnings.filterwarnings('ignore')
```

[3]:
```
forest_fires_df=pd.read_csv("forest_fires_dataset.csv")
cols=['FFMC', 'DMC', 'DC', 'ISI', 'temp', 'RH',
       'wind', 'rain', 'area']
```

[3]:
```
forest_fires_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
```

Widzimy, że dane nie są wybrakowane, czyli nie mamy żadnych null'i.

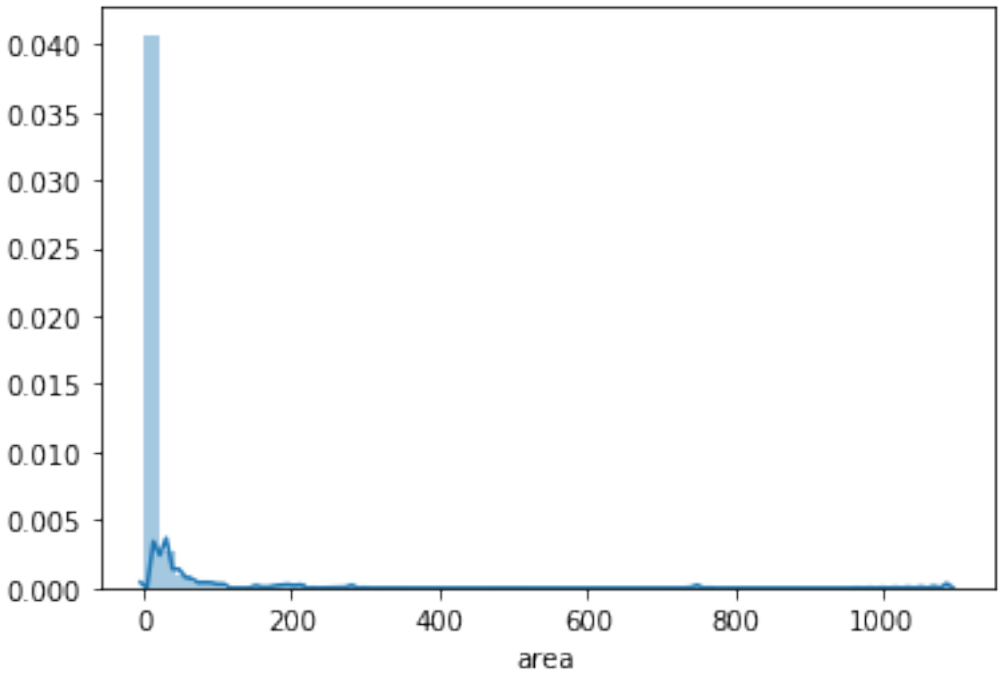
[4]:
```
forest_fires_df['area'].describe()
```

[4]:
```
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
```

Eksploracje zaczynamy od sprawdzenia informacji na temat 'area' czyli targetu, który chcielibyśmy modelować. Od razu widać, że większośc danych znajduje się w bliskiej odległości 0.

[5]:
```
sns.distplot(forest_fires_df['area'])
print("Skośność: %f" % forest_fires_df['area'].skew())
print("Kurtoza: %f" % forest_fires_df['area'].kurt())
```

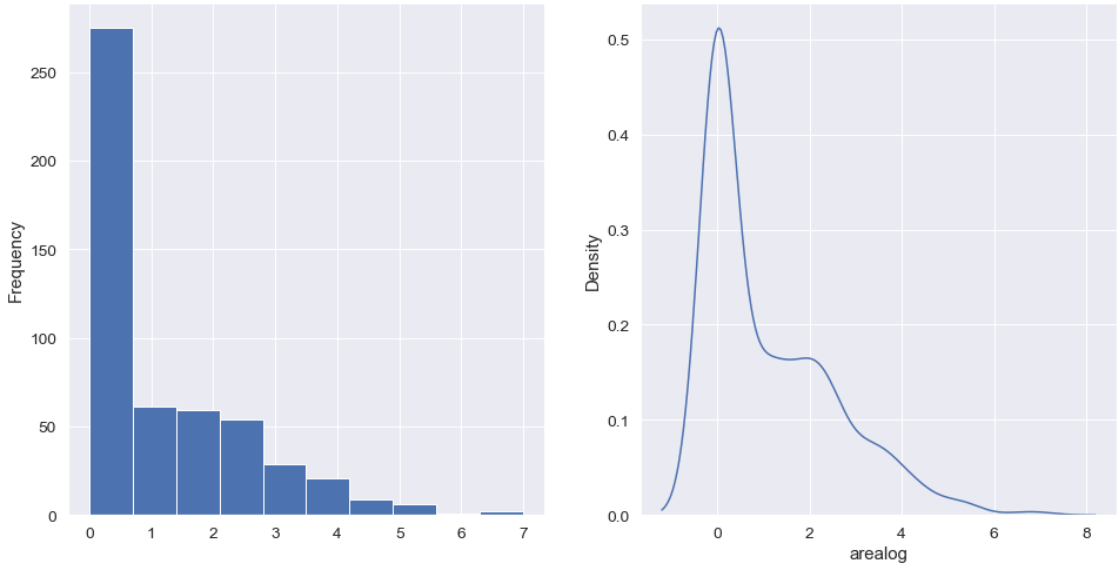
```
Skośność: 12.846934
Kurtoza: 194.140721
```



Wykres, jak można się było spodziewać jest bardzo specyficzny.

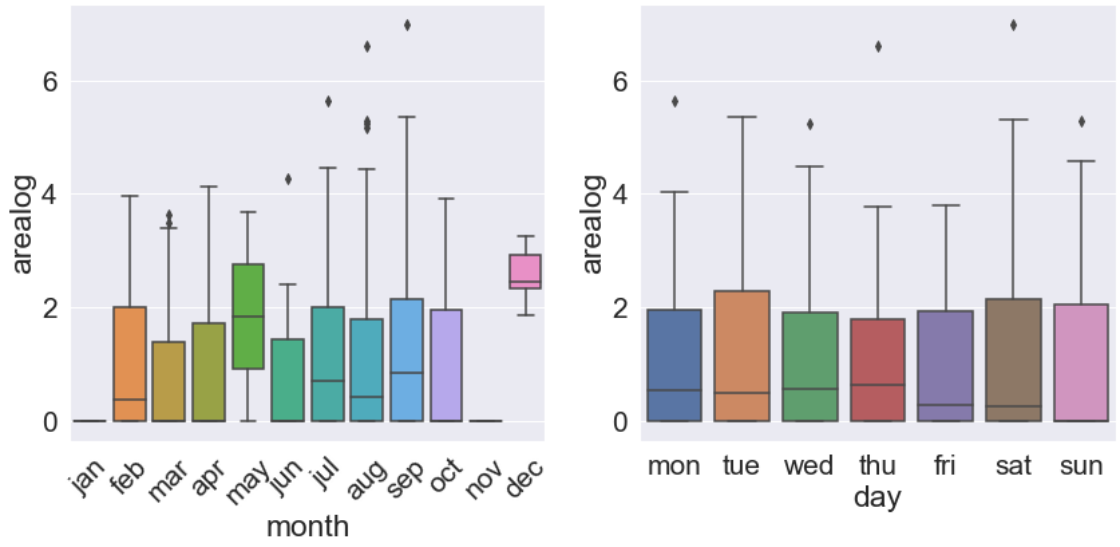
[19]:
```
forest_fires_df['arealog'] = forest_fires_df['area'].map(lambda x: np.log(x+1))
fig, (ax1,ax2)=plt.subplots(1,2,figsize=(16, 8))
forest_fires_df['arealog'].plot.hist(ax=ax1)
sns.distplot(forest_fires_df['arealog'],ax=ax2,hist=False)
print("Skośność: %f" % forest_fires_df['arealog'].skew())
print("Kurtoza: %f" % forest_fires_df['arealog'].kurt())
```

```
Skośność: 1.217838
Kurtoza: 0.945668
```



Autorzy danych zasugerowali użycie log(x) na 'area', co wydaje się bardzo dobrym pomysłem do analizy tej danej.

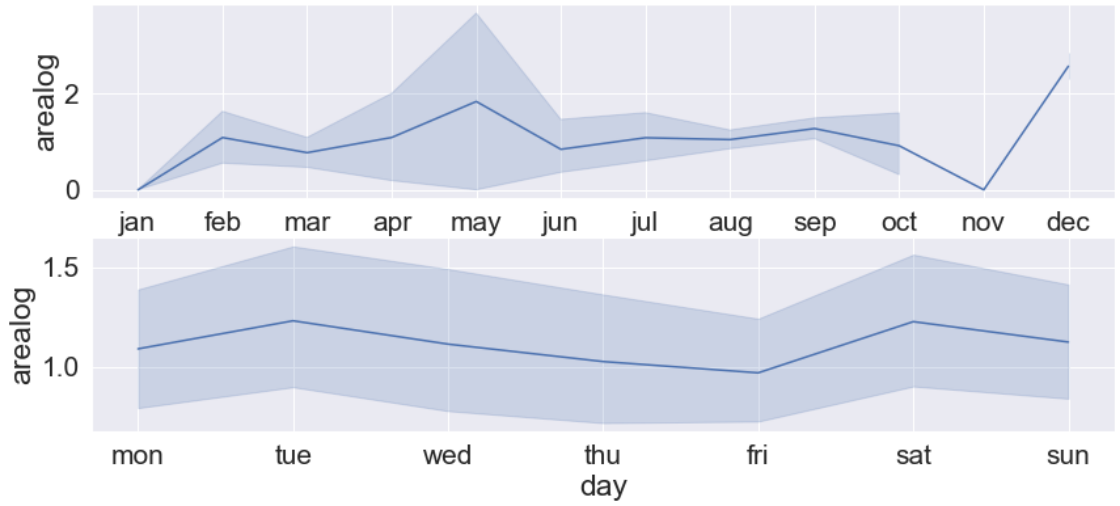
[72]:
```
f, axs = plt.subplots(1,2,figsize=(14, 6))
ax=sns.boxplot(x='month', y="arealog",␣
 ↪data=forest_fires_df,ax=axs[0],order=['jan','feb','mar','apr','may','jun','jul','aug','sep'
plt.setp(ax.get_xticklabels(), rotation=45)
ax2=sns.boxplot(x='day', y="arealog",␣
 ↪data=forest_fires_df,ax=axs[1],order=['mon','tue','wed','thu','fri','sat','sun'])
```



[80]:
```
f, axs = plt.subplots(2,1,figsize=(14, 6))
list_ordering =␣
 ↪['jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec']
forest_fires_df['month'] = pd.Categorical(forest_fires_df["month"],␣
 ↪categories=list_ordering)
sns.lineplot(x='month', y="arealog", data=forest_fires_df,ax=axs[0])

list_ordering = ['mon','tue','wed','thu','fri','sat','sun']
forest_fires_df['day'] = pd.Categorical(forest_fires_df["day"],␣
 ↪categories=list_ordering)
sns.lineplot(x='day', y="arealog", data=forest_fires_df,ax=axs[1])
```

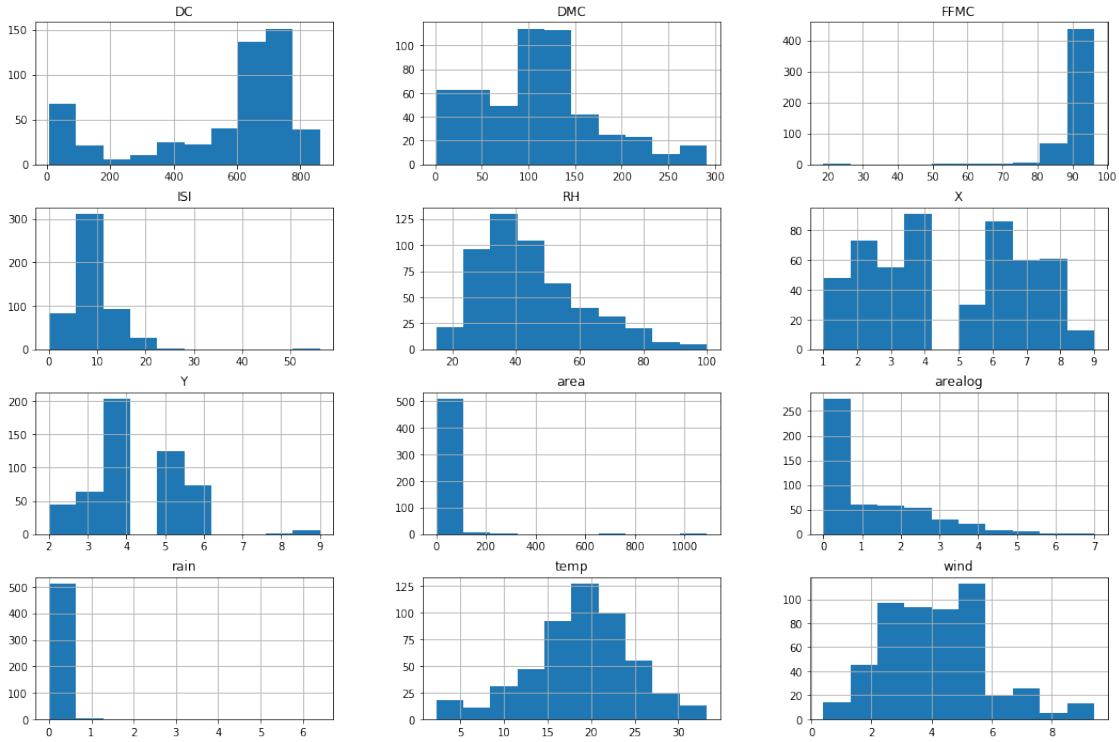
[80]: <AxesSubplot:xlabel='day', ylabel='arealog'>



Z boxplotów i lineplotów zrobionych na zmiennnych kategorycznych (miesiąc, dzień) nie widać jasnej zależności. Warto zauważyć relatywnie duży rozrzut w maju.

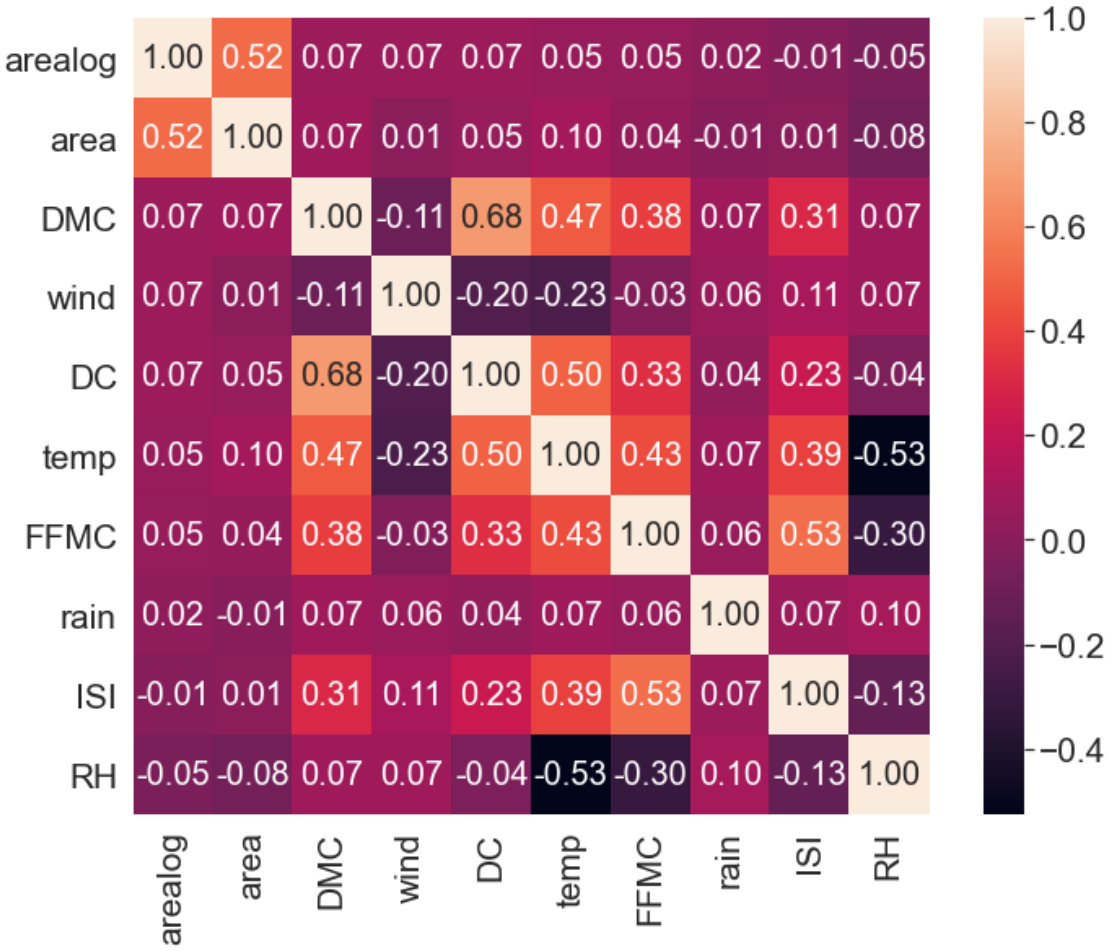
Warto jednak przyjżeć się jak rozkładają się cechy ciągłe, ponieważ już w tym miejscu może być widoczna jakaś zależność.

[8]:
```
forest_fires_df.hist(figsize=(18, 12))
plt.show()
```



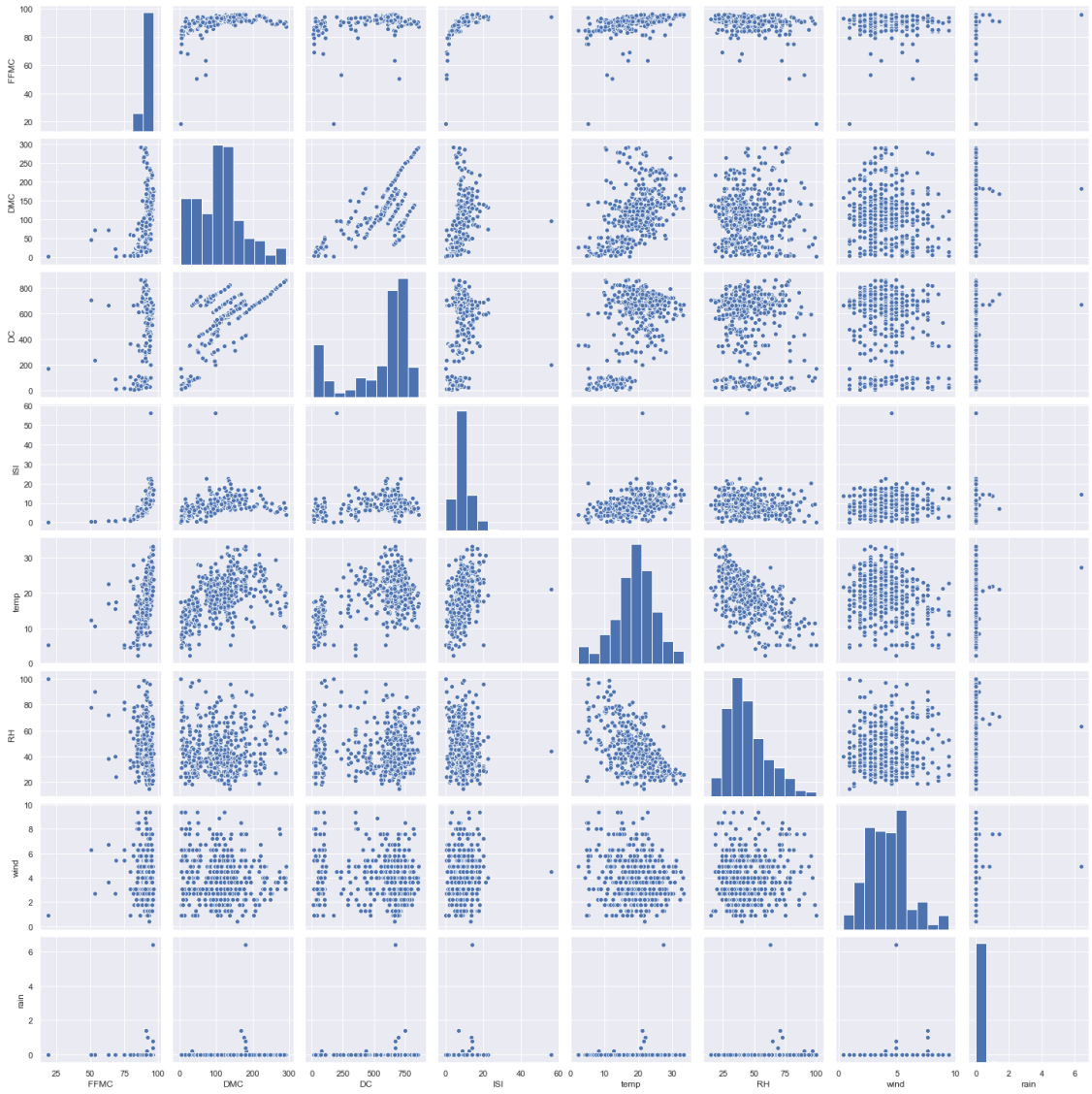
Na pierwszy rzut oka można zacząć już coś podejrzewać, ale trzeba oczywiście to dokładnie zbadać i do tego posłuży nam heatmapa.

[37]:
```
corrmat=forest_fires_df[['FFMC', 'DMC', 'DC', 'ISI', 'temp', 'RH','wind',␣
 ↪'rain', 'area','arealog']].corr()
k = 10
cols = corrmat.nlargest(k, 'arealog')['arealog'].index
cm = np.corrcoef(forest_fires_df[cols].values.T)
sns.set(font_scale=1.9)
f, ax = plt.subplots(figsize=(12, 9))
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f',␣
 ↪annot_kws={'size': 20}, yticklabels=cols.values, xticklabels=cols.values)
plt.show()
```



Pierwsza rzecz, którą widzimy to to, że ani 'area', ani 'arealog' nie jest wyraźnie mocniej skorelowana z którąkolwiek z innych cech. Druga rzecz jest taka, że na heatmapie z naniesionymi wartościami korelacji, można już bardziej zdecydowanie wskazać cechy na które trzeba zwrócić uwagę.

[10]:
```
sns.set()
cols=['FFMC', 'DMC', 'DC', 'ISI', 'temp', 'RH',
       'wind', 'rain']
sns.pairplot(forest_fires_df[cols])
plt.show();
```



Wiedząc już gdzie występują zależności możemy teraz spróbować znaleźć ich 'kształt' np. punkty na wykresie DC i DMC wyglądają jakby były skupione wokół jakiejś prostej, wykres FFMC i ISI wygląda podobnie do log(x).

[4]:
```
ProfileReport(forest_fires_df)
```

```
HBox(children=(HTML(value='Summarize dataset'), FloatProgress(value=0.0, max=27.
↪0), HTML(value='')))
```

```
HBox(children=(HTML(value='Generate report structure'), FloatProgress(value=0.0,␣
↪max=1.0), HTML(value='')))
```

```
HBox(children=(HTML(value='Render HTML'), FloatProgress(value=0.0, max=1.0),␣
 ↪HTML(value='')))
```

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

[4]:

Takie narzędzie wydaje się bardzo przydatne w podstawowej analizie, ale na pewno nie jest wystarczające. Normalnie zobaczylibyśmy więcej wykresów takich jak boxploty i barploty. Podejrzewam również, że przy dużych zbiorach danych tworzenie takiego raportu może zająć znaczącą ilość czasu.