# Eksploracja danych w Pythonie

## Scikit-learn i okolice

Dariusz Brzeziński

*Instytut Informatyki, Politechnika Poznańska*

# Plan zajęć

- Wstęp
- Biblioteki
  - Pandas
  - Scikit-learn
  - Plotnine
- Jupyter
- Zadanie

# Dlaczego Python?

- Popularny język programowania
- Prosta składnia
- Bogactwo gotowych bibliotek do analizy danych:
  - Numpy, Scipy, Pandas
  - Scikit-learn, PySpark, NLTK
  - Matplotlib, seaborn, plotnine, altair
- Szybszy niż R*
- Nie musi trzymać wszystkich obiektów w pamięci

# Co zainstalować

## Środowisko

- Python
- Biblioteki scipy, numpy, pandas, scikit-learn
- IPython

Powyższy pakiet bibliotek można zainstalować w ramach dystrybucji Anaconda. Opcja polecana zwłaszcza dla użytkowników Windowsa.

## Edytor

- PyCharm lub VS Code

# Pandas

- Biblioteka do wczytywania, zapisywania, filtrowania i modyfikowania zbiorów danych
- Operacje zoptymalizowane w Cythonie i C
- Podstawowe typy
  - Series
  - DataFrame
- Struktury danych i operacje inspirowane R
- Pozwala filtrować zbiory danych w sposób podobny do operatora [] w R

# Pandas - przykład

```
import pandas as pd

df = pd.read_csv('test.csv', sep=';', na_values=['?'])

df.head()   # pierwsze 6 wierszy
df.tail(10) # ostatnie 10 wierszy
df.shape    # krotka reprezentująca wymiary zbioru

df['A']     # wybiera kolumnę A
df[0:3]     # pierwsze trzy wiersze
df.loc[:,['A', 'B']] # wszystkie wiersze kolumn A i B
df.iloc[1:3,1:3]     # 2 i 3 wiersz, 2 i 3 kolumna
```

# Pandas - przykład

```
df.loc[:, 'Name'].unique()   # unikatowe wartości
df.Name.unique()     # inny sposób na operacje na kolumnie

df.loc[:, 'Noname'] = 'nada...'              # nowa kolumna
df.loc[:, 'Lowname'] = df.Name.str.lower() # str
df.loc[:, 'Oname'] = df.Name.str.contains('O')
df.drop('Noname', axis=1)                  # usuwanie (kolumn)

df[(df.Age > 50)]  # filtrowanie; tak, nawiasy są ważne...
df[~(df.Age > 50)] # ... nie, nie można korzystać z .loc[]

df.to_csv('new.csv', sep=',', na_values=['n/a']) # zapis
```

# Pandas – inne przydatne funkcje

```
df.sum()              # są też mean, median, idxmax
df.cumsum()           # min, max, count, idxmin

df.astype('float32')              # rzutowanie
df.sort_values(by='Lowname')      # sortowanie po indeksie
df.apply(lambda x: x*23)          # funkcje na osiach

# Uwaga! Poniższe funkcje z reguły działają trochę inaczej
# niż to sobie wyobrażamy...
pd.concat([df1, df2, df3])
df1.append(df2)
pd.merge(df1, df2, how='left', on=['key1', 'key2'])
```

# Scikit-learn

- Najpopularniejsza biblioteka do eksploracji danych w Pythonie
- Oferuje algorytmy do:
  - klasyfikacji, regresji, grupowania,
  - wstępnego przetwarzania danych, redukcji wymiarów,
  - optymalizacji parametrów
- Oparta na NumPy, SciPy i matplotlib
- Darmowa, otwarta, aktywnie rozwijana
- Bardzo dobra dokumentacja!

# Scikit-learn - przykład

```
from sklearn import datasets
import sklearn.model_selection as skms
from sklearn.metrics import classification_report
from sklearn.svm import SVC

digits = datasets.load_digits()

n_samples = len(digits.images)
# Uwaga! X i y muszą mieć wyłącznie wartości liczbowe!
X = digits.images.reshape((n_samples, -1))
y = digits.target

X_train, X_test, y_train, y_test =
skms.train_test_split(X, y, test_size=0.3, random_state=0)
```

# Scikit-learn - przykład

```
tuned_parameters = [{'kernel': ['rbf'], 'gamma': [1e-3,
1e-4], 'C': [1, 10, 100, 1000]}, {'kernel': ['linear'],
'C': [1, 10, 100, 1000]}]
scores = ['precision', 'recall']


for score in scores:
    clf = skms.GridSearchCV(SVC(C=1), tuned_parameters,
                            cv=5, scoring='%s_weighted' % score)
    clf.fit(X_train, y_train)
    print(clf.best_params_)
    y_true, y_pred = y_test, clf.predict(X_test)
    print(classification_report(y_true, y_pred))
```

# Plotnine

- Plotnine = ggplot w Pythonie
- Nowa biblioteka (i póki co wciąż rozwijana)

- Alternatywy to:
  - matplotlib (na tej bibliotece opierają się wszystkie inne)
  - seaborn (świetna do naukowych wizualizacji)
  - plotly (znany przyjaciel, który ożywi wykresy)

# Plotnine - przykład

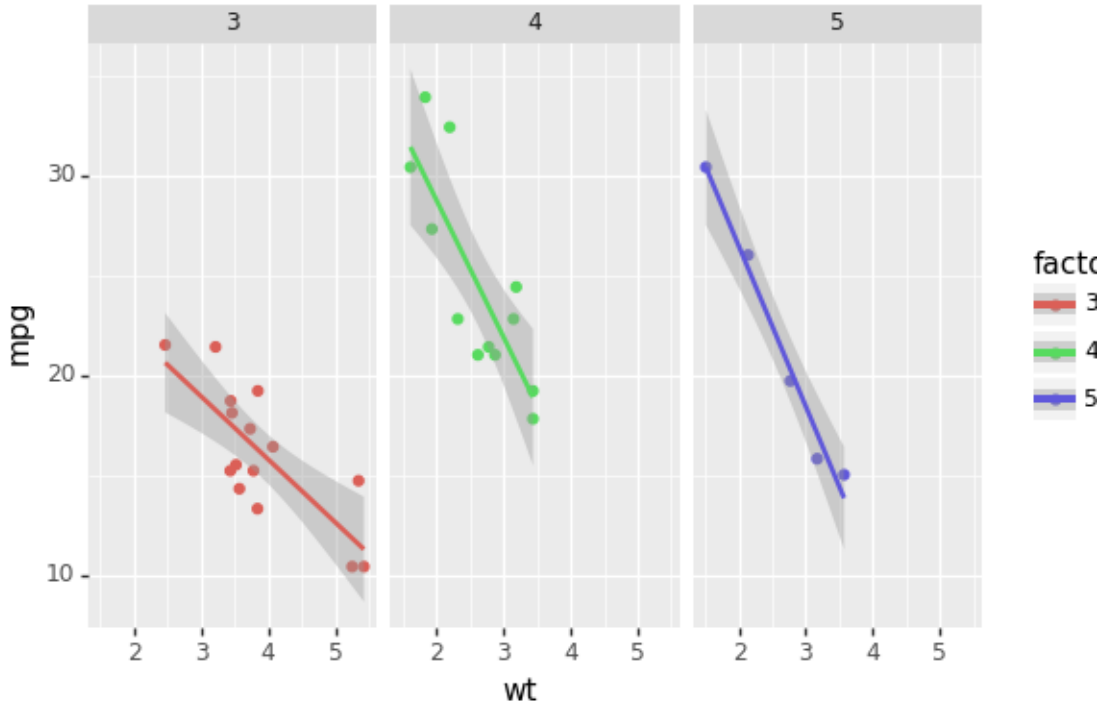
```
from plotnine import *
from plotnine.data import mtcars

(ggplot(mtcars, aes('wt', 'mpg', color='factor(gear)'))
      + geom_point()
      + stat_smooth(method='lm')
      + facet_wrap('~gear'))
```

# Plotnine - przykład

```
from plotnine import *
from plotnine.data import mtcars

(ggplot(mtcars, aes('wt', 'mpg', color='factor(gear)'))
       + geom_point()
       + stat_smooth(method 'lm')
       + facet_wrap('
```



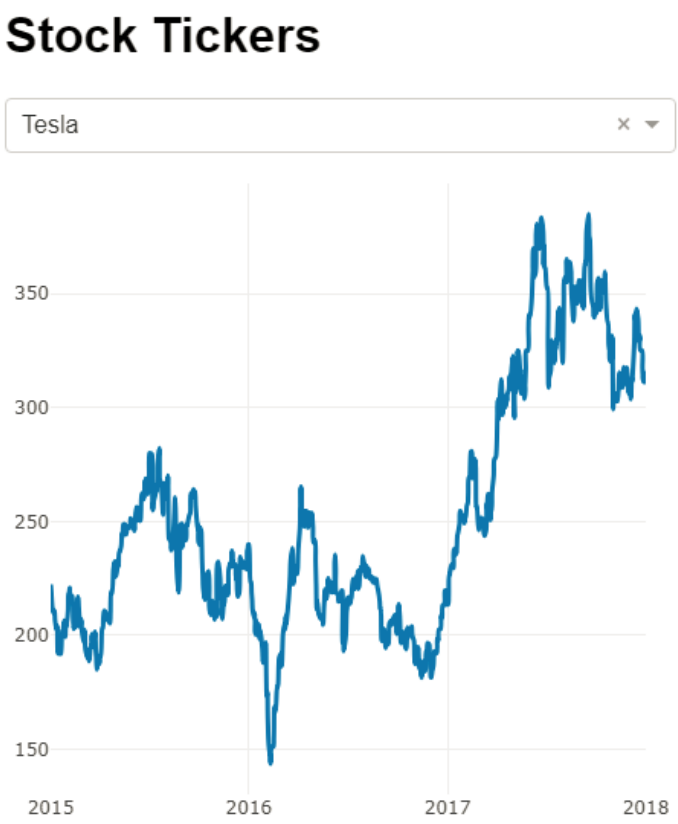
# Ipython i Jupyter Notebook

- IPython = interaktywna konsola Pythona
- Jupyter Notebook = strony z kodem wykonywalnym

1. Jupyter Notebook może być wykorzystywany wraz z językiem Markdown do łączenia kodu z opisem słownym
2. Jupyter obsługuje 40 różnych kerneli (języków) w tym kernel R
3. Do tworzenia notatników można wykorzystać przeglądarkę, Pycharm* lub JupyterLab*
4. Pliki **.ipynb* są obsługiwane przez GitHuba

# Bonus: Dash

- Dash = Shiny dla Pythona




```python
import dash
from dash.dependencies import Input, Output
import dash_core_components as dcc
import dash_html_components as html
from pandas_datareader import data as web
from datetime import datetime as dt

app = dash.Dash()

app.layout = html.Div([
    html.H1('Stock Tickers'),
    dcc.Dropdown(
        id='my-dropdown',
        options=[
            {'label': 'Coke', 'value': 'COKE'},
            {'label': 'Tesla', 'value': 'TSLA'},
            {'label': 'Apple', 'value': 'AAPL'}
        ],
        value='COKE'
    ),
    dcc.Graph(id='my-graph')
])
```

# Zadanie

1. Pobierz dane i notatnik ze strony prowadzącego
2. Uruchom notatnik i wykonuj po kolei zadania

# Przydatne linki

- http://www.learnpython.org/
- http://scikit-learn.org/stable/
- https://www.kaggle.com/learn/
- http://blog.kaggle.com/author/kevin-markham/
- http://pandas.pydata.org/pandas-docs/stable/10min.html
- http://plotnine.readthedocs.io/en/stable/
- http://seaborn.pydata.org/
- https://jupyter.org/