

# Praktyczny wstęp do Machine Learning

**Mateusz Rogowski**  
Pythonist & Data Scientist  
m.rogowski90@gmail.com



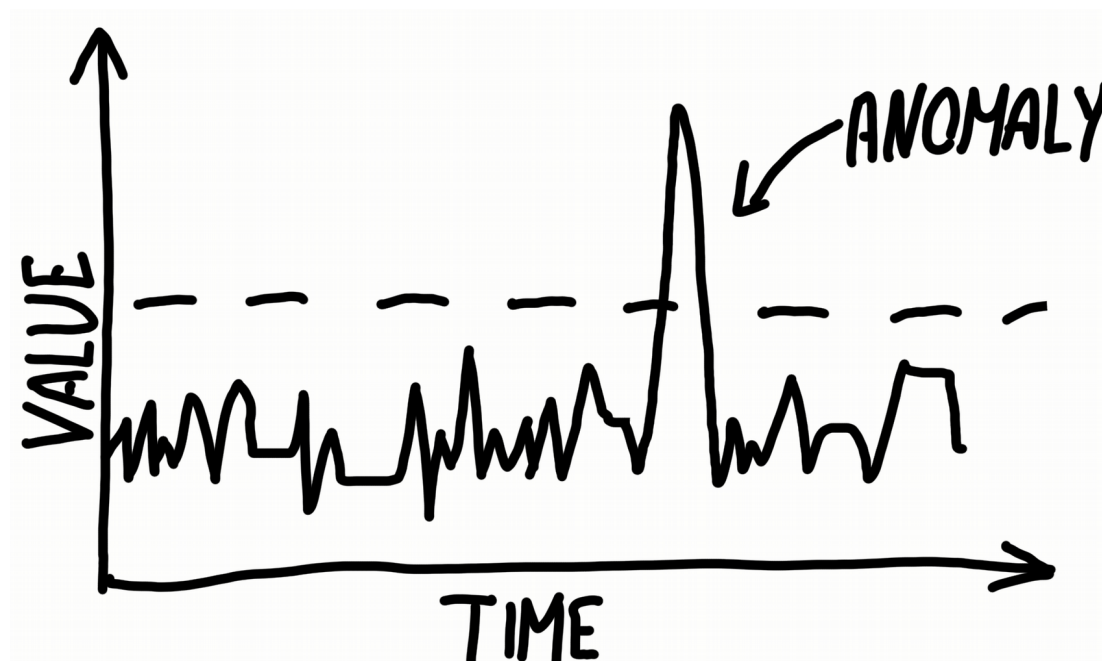
**#DataScience**

**#MachineLearning**

**#DeepLearning**

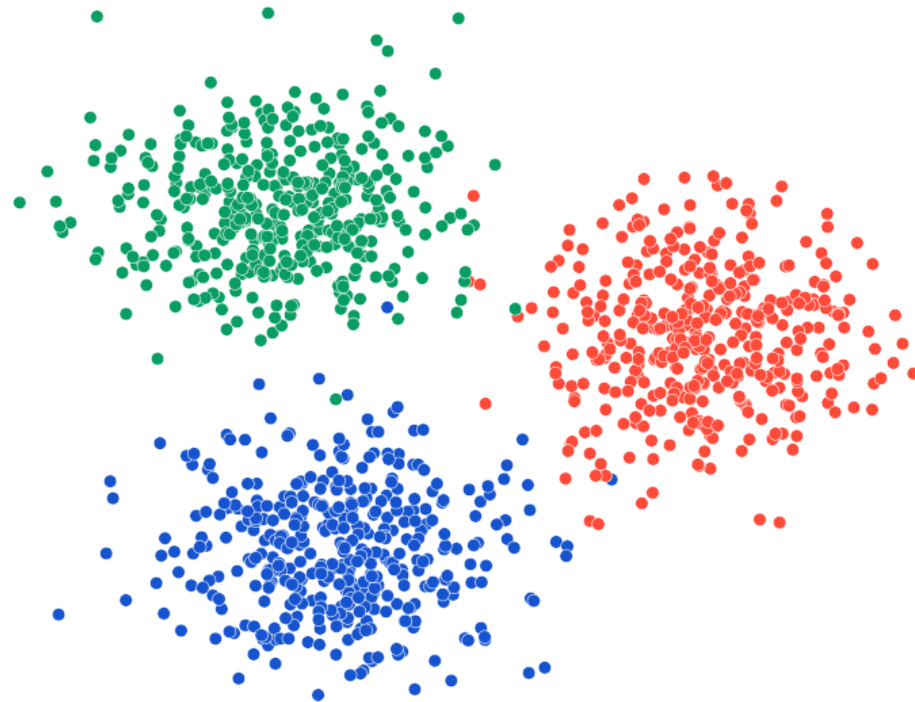
## Wykrywanie anomalii

- Oszustwa
- Włamania
- Katastrofy naturalne
- Defekty przemysłowe
- Monitoring zdrowia



## Klasyfikacja

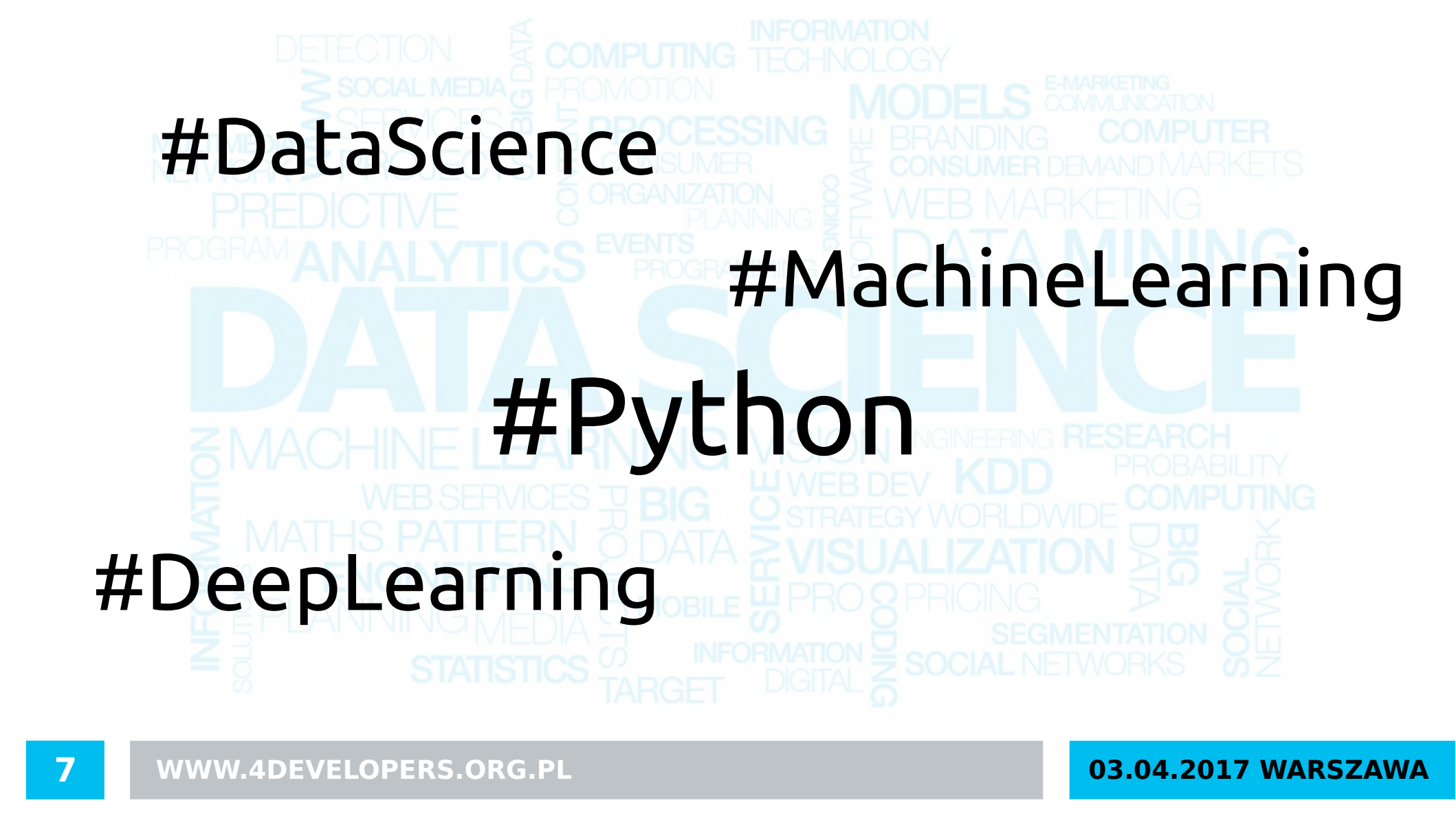
- Automatyczne kategorie postów
- Wykrywanie typu choroby
- Kategoryzacja obrazów
- Wykrywanie twarzy



## Predykcja

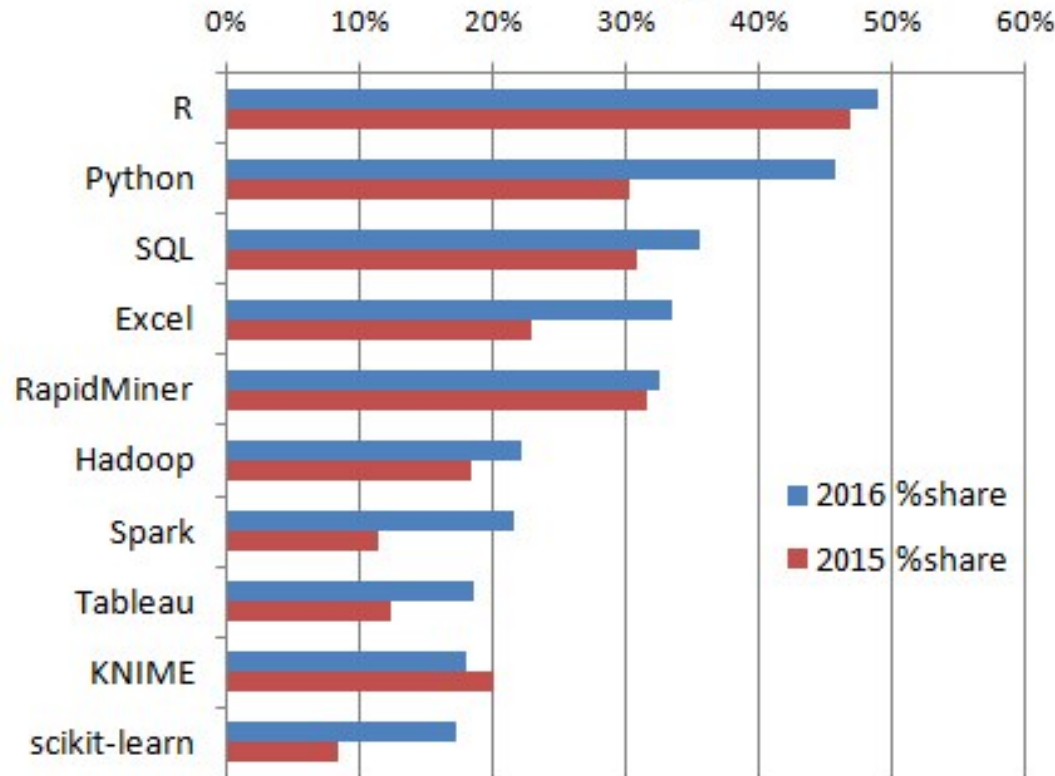
- Wartości akcji
- Nakład produkcji
- Obciążenie serwerów
- Termin wyjścia ze szpitala



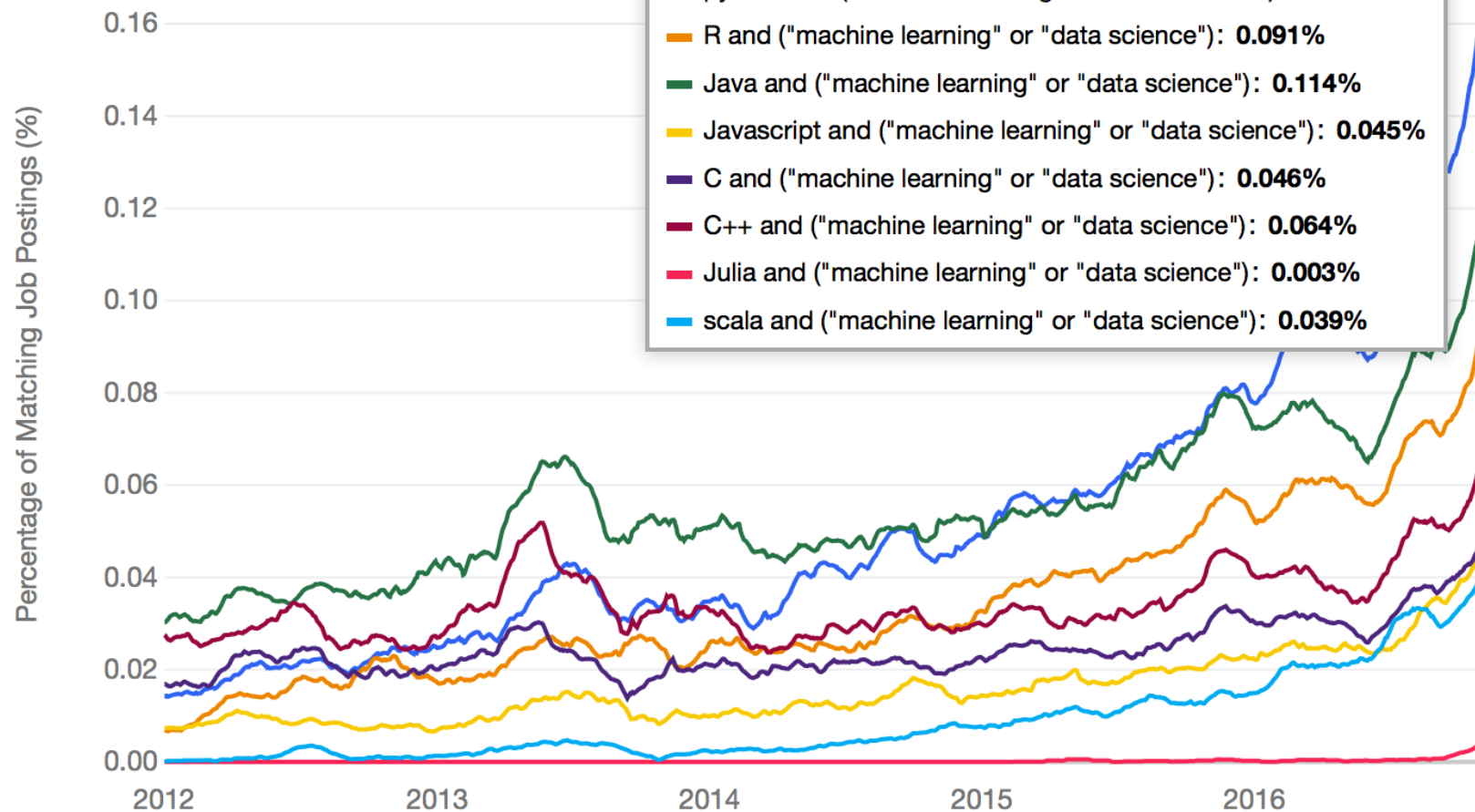


**#DataScience**  
**#MachineLearning**  
**#Python**  
**#DeepLearning**

## KDnuggets Analytics/Data Science 2016 Software Poll, top 10 tools

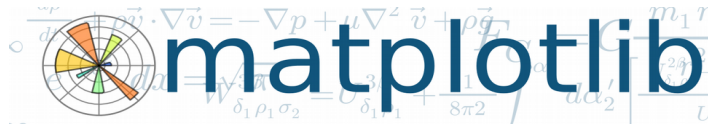


<http://www.kdnuggets.com/2016/06/r-python-top-analytics-data-mining-data-science-software.html>



[https://www.ibm.com/developerworks/community/blogs/jfp/entry/What\\_Language\\_Is\\_Best\\_For\\_Machine\\_Learning\\_And\\_Data\\_Science](https://www.ibm.com/developerworks/community/blogs/jfp/entry/What_Language_Is_Best_For_Machine_Learning_And_Data_Science)





theano



- Proste i wydajne narzędzia do analizowania danych
- Powszechnie dostępne, uniwersalne
- Zbudowane w oparciu o biblioteki NumPy, SciPy i matplotlib
- Open source na licencji BSD – można wykorzystywać komercyjnie
- Optymalna implementacja – C + Fortran + Cython
- Wielowątkowość

# scikit-learn – główne moduły

- Klasyfikacja
- Regresja
- Klastrowanie
- Redukcja wymiarów
- Selekcja modelu
- Wstępne przetwarzanie



# scikit-learn – spójny interfejs

```
def fit(self, X, y=None):
```

```
def predict(X)
```

```
def predict_proba(X)
```

```
def transform(self, X, y=None):
```

```
def fit_transform(self, X, y=None, **fit_params):
```

# scikit-learn – spójny interfejs

```
cls = SVC(kernel='rbf')
```

```
cls = RandomForestClassifier(n_estimators=20)
```

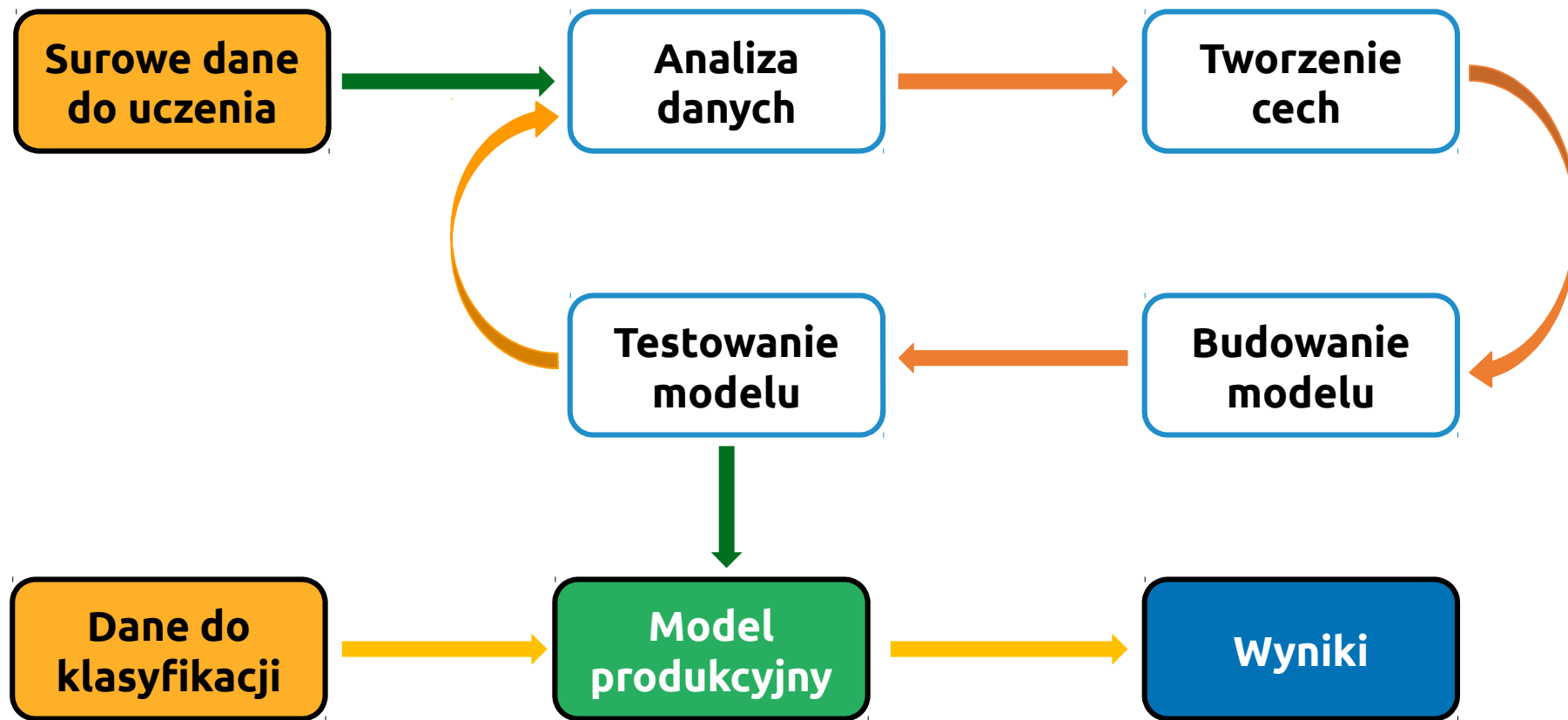
```
cls.fit(X, y)
```

```
cls.predict(x)
```

# scikit-learn - duży zakres dostępnych funkcjonalności

- gotowe zestawy testowe,
- próbkowanie zestawów danych (`train_test_split`, `StratifiedShuffleSplit`)
- automatyczne dopasowanie parametrów (`grid_search`)
- wiele sposobów oceniania wyników (`accuracy`, `precision`, `recall`, `F1`)
- wiele dostępnych algorytmów (klasyfikatory, regresory, grupowanie)

# Przebieg tworzenia modelu



## **# PRZYKŁADOWY NOTEBOOK**

**[https://github.com/Ritsuki/4Developers\\_2017](https://github.com/Ritsuki/4Developers_2017)**



# Q&A

**Mateusz Rogowski**  
Pythonist & Data Scientist  
m.rogowski90@gmail.com

