Warsztaty modelowania

05 – uczenie i ewaluacja modelu

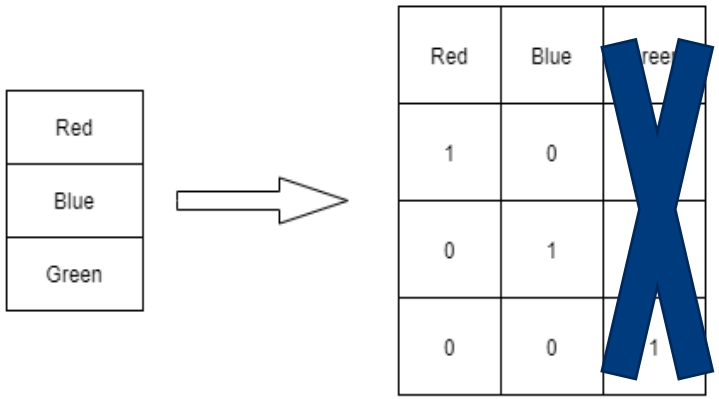
opracowała Patrycja Naumczyk

Uczenie i ewaluacja – zasady ogólne

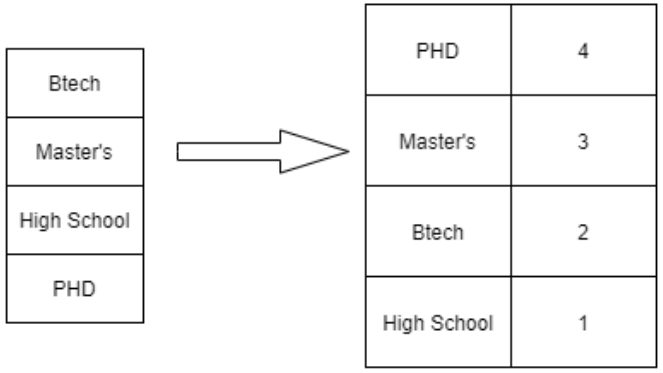
1. Uczenie
    a) Przekształcenie zmiennych kategorialnych
    b) Regularyzować, czy nie regularyzować
2. Ewaluacja
    a) Podział na set testowy i treningowy
    b) Przeciekanie danych („data leakage”)
    c) Właściwe metryki

Uczenie

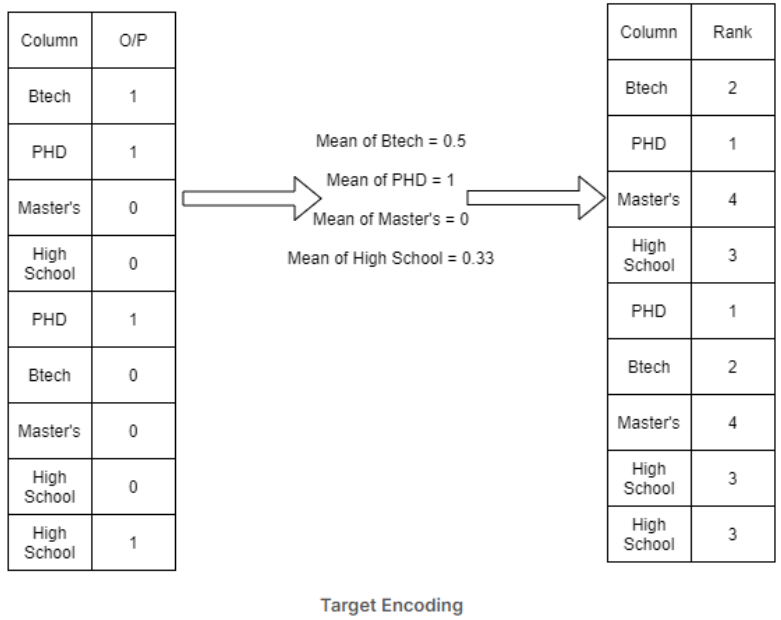
1. Przekształcenia zmiennych kategorialnych
   a) One-hot encoding
   b) Ordinal encoding
   c) Target encoding



One Hot Encoding



Label Encoding



Target Encoding

Uczenie

1. Regularyzacja:
   a) L1 – Lasso
   b) L2 – Ridge
2. Konieczne skalowanie zmiennych ciągłych (!!!)
   a) Min-Max
   b) Normalizacja (standardization / Z-score)

$$\mathrm{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

$$\mathrm{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2$$

RIDGE

$$\mathrm{RSS} + \lambda \sum_{j=1}^{p} |\beta_j|.$$

LASSO

$$\frac{\sum_{i=1}^{n} (y_i - x_i^J \hat{\beta})^2}{2n} + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^{m} \hat{\beta}_j^2 + \alpha \sum_{j=1}^{m} |\hat{\beta}_j| \right)$$

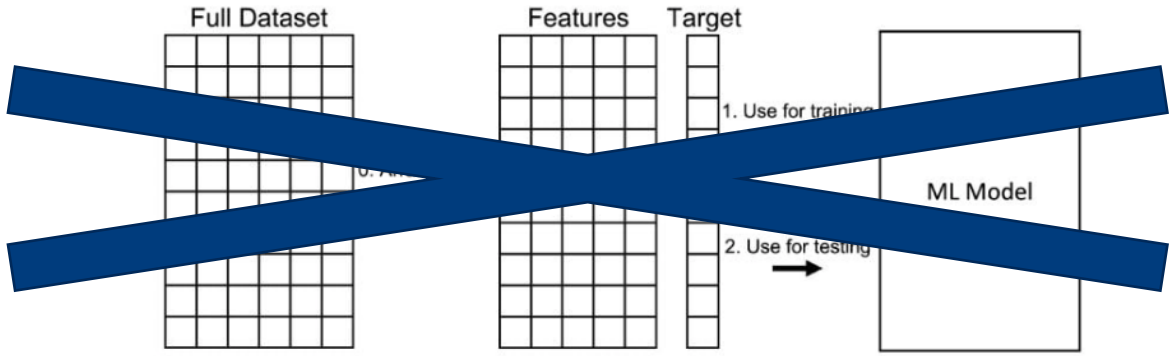
ELASTIC NET

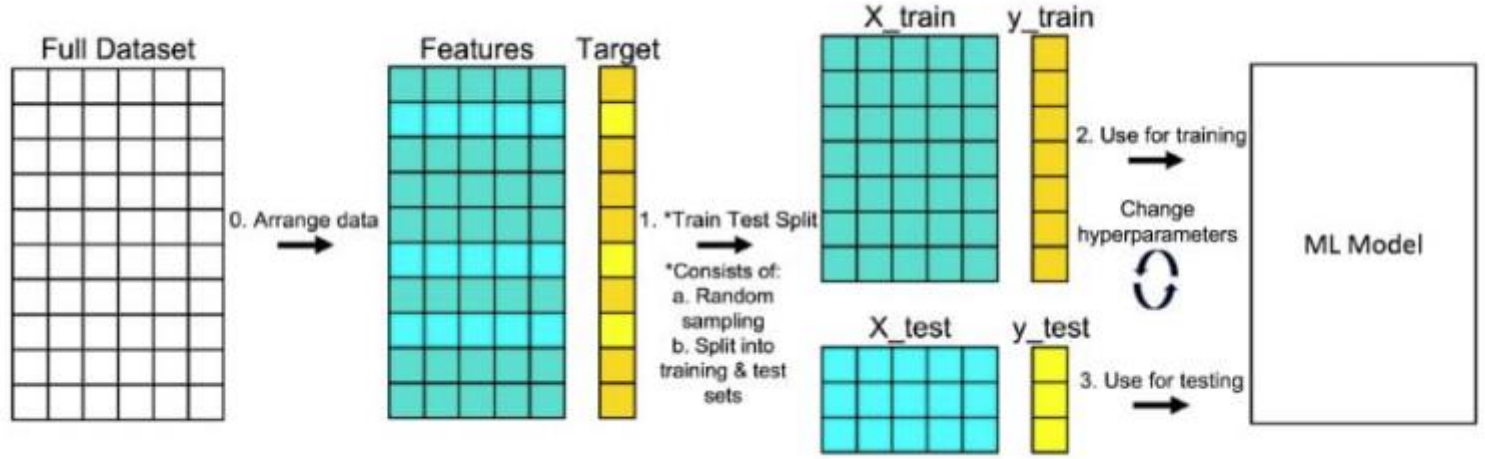
```
X_new = (X - X_min)/(X_max - X_min)
```

```
X_new = (X - mean)/Std
```

Ewaluacja

1. Podział na sety treningowy i testowy
   a) „Przeciekanie” danych
   b) Podział proporcjonalny do zmiennej



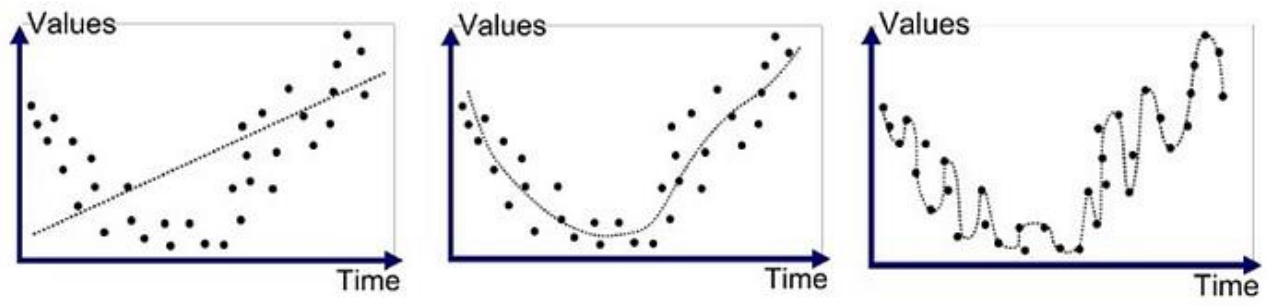


Ewaluacja

1. Podział na sety treningowy i testowy
   a) „Przeciekanie” danych
   b) Podział proporcjonalny do zmiennej
2. Metryki:
   a) R^2
   b) Odchylenie

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

$$\mathrm{D}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} \begin{cases} (y_i - \hat{y}_i)^2, & \text{for } p = 0 \text{ (Normal)} \\ 2(y_i \log(y_i/\hat{y}_i) + \hat{y}_i - y_i), & \text{for } p = 1 \text{ (Poisson)} \\ 2(\log(\hat{y}_i/y_i) + y_i/\hat{y}_i - 1), & \text{for } p = 2 \text{ (Gamma)} \\ 2\left(\frac{\max(y_i,0)^{2-p}}{(1-p)(2-p)} - \frac{y_i \hat{y}_i^{1-p}}{1-p} + \frac{\hat{y}_i^{2-p}}{2-p}\right), & \text{otherwise} \end{cases}$$



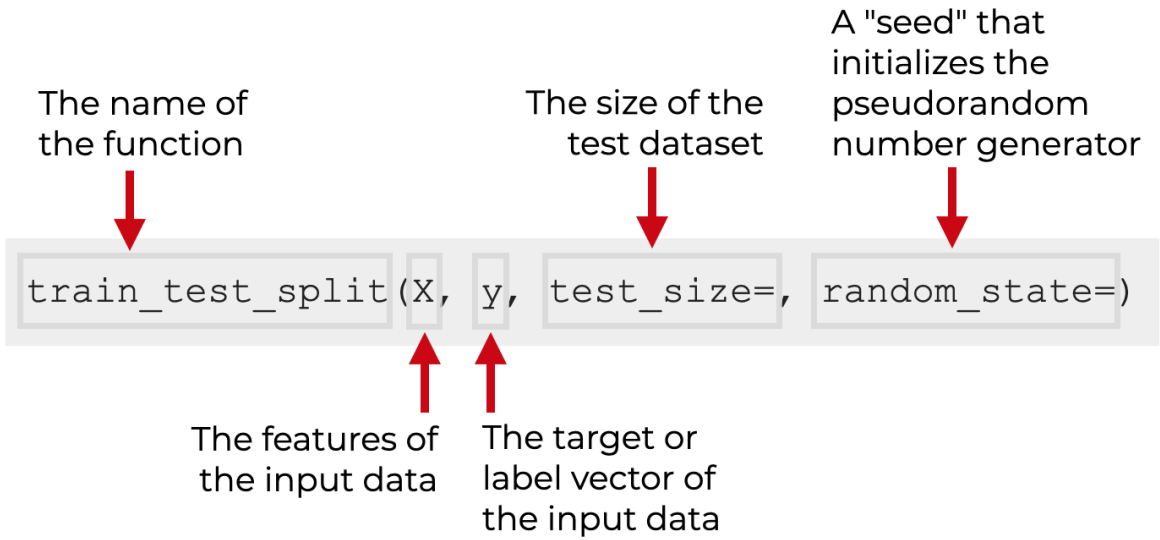
Underfitted | Good Fit/Robust | Overfitted

Uczenie i ewaluacja – scikit-learn vs statsmodels

1. Podział na set treningowy i testowy – scikit-learn:
   a) funkcja train_test_split()
2. Przekształcanie zmiennych – sikit-learn (metody .fit() .tranform() .fit_transform()):
   a) zmienne kategorialne: kodowanie
      i. obiekt OneHotEncoder()
      ii. obiekt OrdinalEncoder()
   b) zmienne ciągłe: skalowanie
      i. obiekt MinMaxScaler()
      ii. obiekt StandardScaler()
3. Uczenie modelu:
   a) scikit-learn (metody .fit() .predict()):
      i. obiekt PoissonRegressor()
      ii. obiekt GammaRegressor()
   b) statsmodels (metody .fit() .fit_regularized() .predict()):
      i. obiekt GLM(family=sm.families.Poisson())
      ii. obiekt GLM(family=sm.families.Gamma())
4. Ewaluacja modelu – scikit-learn:
   a) funkcje poszczególnych metryk
      https://scikit-learn.org/stable/modules/model_evaluation.html

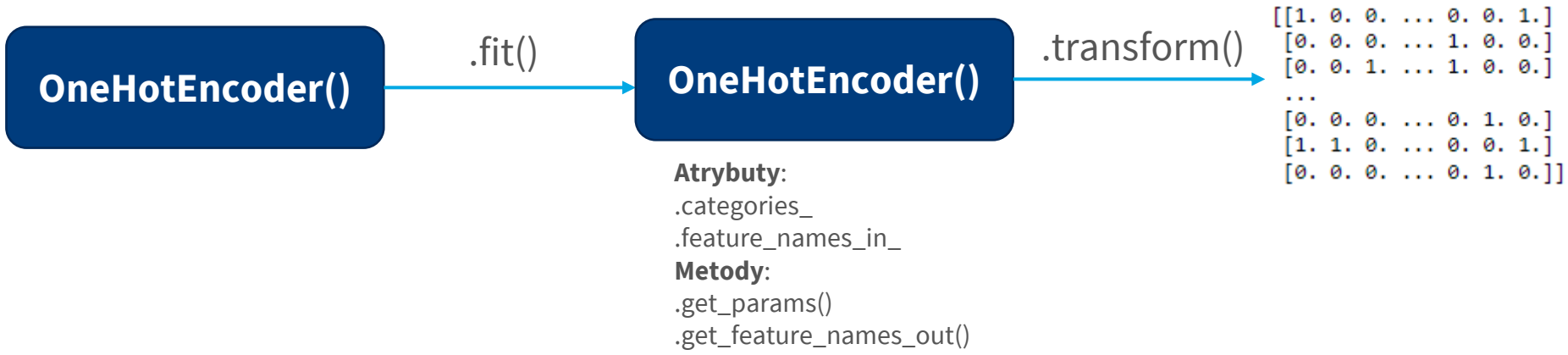
Funkcja train_test_split()

1. Podział na próbkę treningową i testową
2. Jeśli jest potrzeba wydzielenia osobno próbki „hold-out”, konieczne jest dwukrotne przepuszczenie danych przez funkcję
3. Ważne parametry (patrz: dokumentacja):
   a) **test_size** – jaki odsetek zbioru ma być w próbce testowej (default: 0.25)
   b) **random_state** – zapamiętanie stanu losowego, użytego przy podziale (powtarzalność podziałów!)
   c) **stratify** – podział proporcjonalny do zmiennej
4. Zachowuje typ danych (dataframe -> dataframe, numpy array -> numpy array)



Przekształcanie zmiennych

1. Zmienne kategorialne -> kodowanie (np. OneHotEncoder() OrdinalEncoder())
2. Zmienne ciągłe -> skalowanie (np. MinMaxScaler() StandardScaler())
3. Podstawowe metody na obiekcie:
   a) .**fit**()
   b) .**transform**()
   c) .**fit_transform**()
4. NIE zachowuje typu danych (!)
   a) Wynikiem numpy array
   b) Konieczność przekształcenia do dataframe



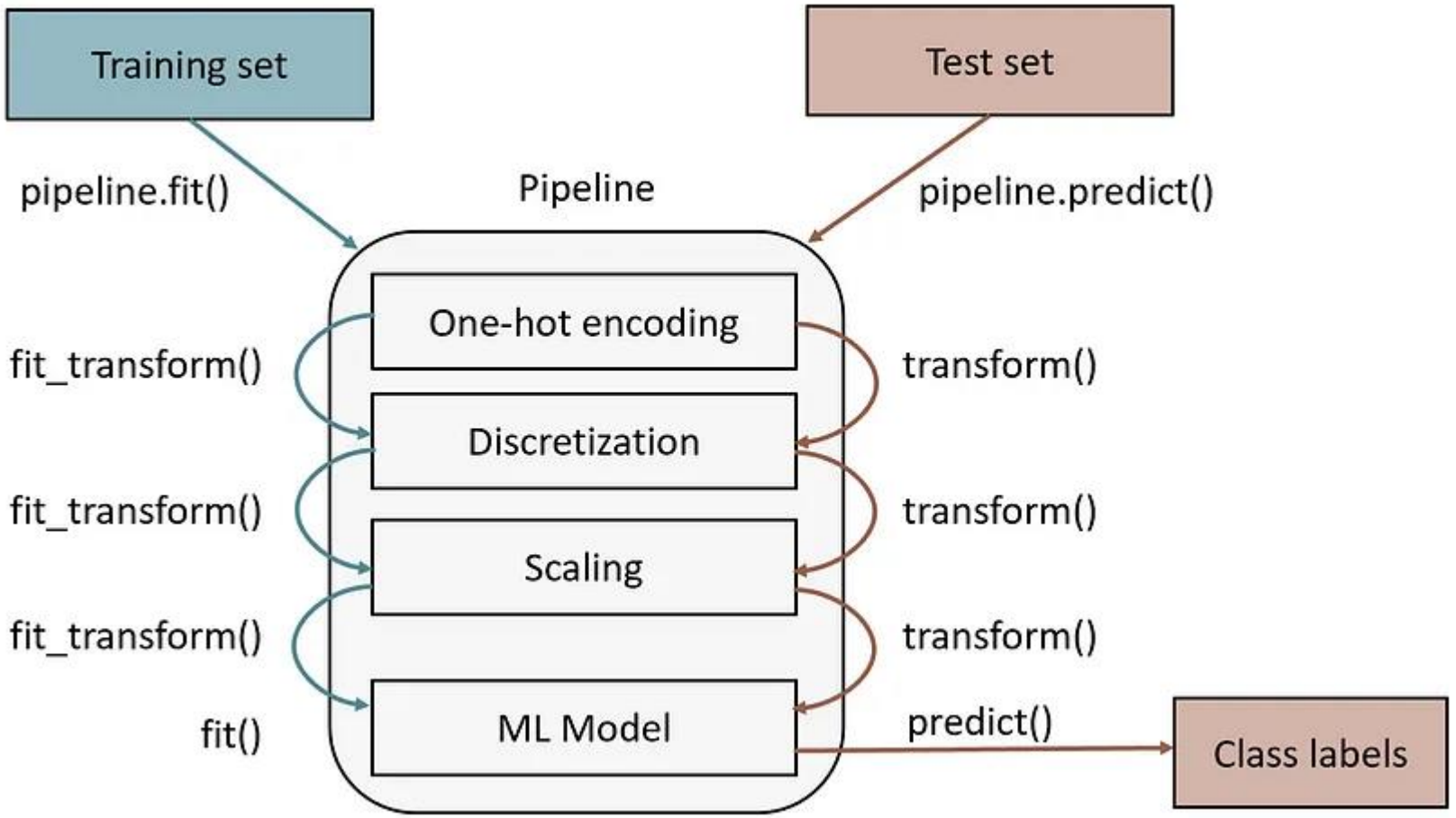
Uczenie modelu – scikit-learn

1. Model częstości -> PoissonRegressor()
2. Model średniej szkody -> GammaRegressor()
3. Podstawowe metody na obiekcie:
   a) .**fit**()
   b) .**predict**()
4. NIE zachowuje typu danych (!)
   a) Wynikiem numpy array
   b) Konieczność przekształcenia do dataframe
5. NIE wylicza prawdopodobieństwa poszczególnych zmiennych wyjaśniających, ani F całego modelu
6. Ocena modelu tylko indywidualna przez użytkownika (domyślnie D^2)
7. Regularyzacja tylko L2 (parametr alpha$\in$[0.0, inf) domyślnie wartość 1.0)

Atrybuty:
.coef_
.feature_names_in_
.intercept_

```
array([0.40164084, 0.49078604, 0.46763855, 0.47875   , 0.41495275,
       0.47448992, 0.44240534, 0.4262428 , 0.37862659, 0.49555404,
       0.47602527, 0.37936182, 0.40186752, 0.38349234, 0.49078604,
       0.49729195, 0.42151945, 0.46085436, 0.38349234, 0.38956118,
       0.47547504, 0.47641806, 0.49729195, 0.48774034, 0.38956118,
       0.41096026, 0.3818036 , 0.37606936, 0.42151945, 0.45990434,
       0.49643638, 0.44736766, 0.47514809, 0.45711726, 0.49555404,
       0.41495275, 0.47221076, 0.47804511, 0.40749027, 0.4860358 ,
       0.49763413, 0.47875   , 0.37503606, 0.46955453, 0.48135939,
       0.41096026, 0.49729195, 0.45460398, 0.46853901, 0.39414875,
       0.48774034, 0.45711726, 0.44736766, 0.38370878, 0.51788313,
       0.37862659, 0.45711726, 0.44240534, 0.37503606, 0.42416317,
       0.38370878, 0.38956118, 0.40186752, 0.47736295, 0.43750289,
       0.47221076, 0.46893189, 0.49729195, 0.48058266, 0.47875   ,
       0.39469136, 0.45711726])
```

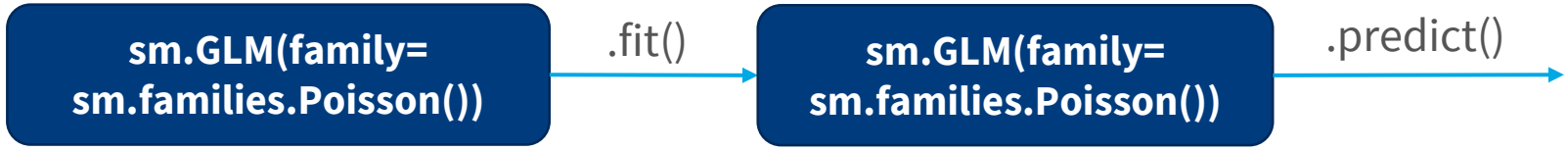
Co na którym zbiorze?



Uczenie modelu – statsmodels

1. Równanie modelu zapisane tekstowo (notacja R, biblioteka patsy):
   zmienna_wyjaśniana ~ zmienna_wyjaśniająca1 + zmienna_wyjaśniająca2 + …

   y ~ x + C(x1) + x:x1 + x2*x3
2. Formatowanie danych wejściowych (funkcja dmatrices())
3. Jeden obiekt dla GLMs:
   a) statsmodels.api.GLM(), wskazanie rodziny poprzez parametr *family*
   b) dla rodziny Poisson i Gamma konieczne wskazanie zmiennej wagowej (exporyz dla częstości, liczba szkód dla średniej szkody)
4. Podstawowe metody na obiekcie:
   a) .**fit**() *ALE! .fit() zwraca nowy obiekt(!)*
   b) .**predict**()
   c) regularyzacja poprzez metodę **fit_regularized()**
      i. parametr alpha∈[0.0, inf), domyślnie 0
      ii. parametr L1_w ∈[0.0, 1.0], L1 = L1_w, L2 = 1-L1_w

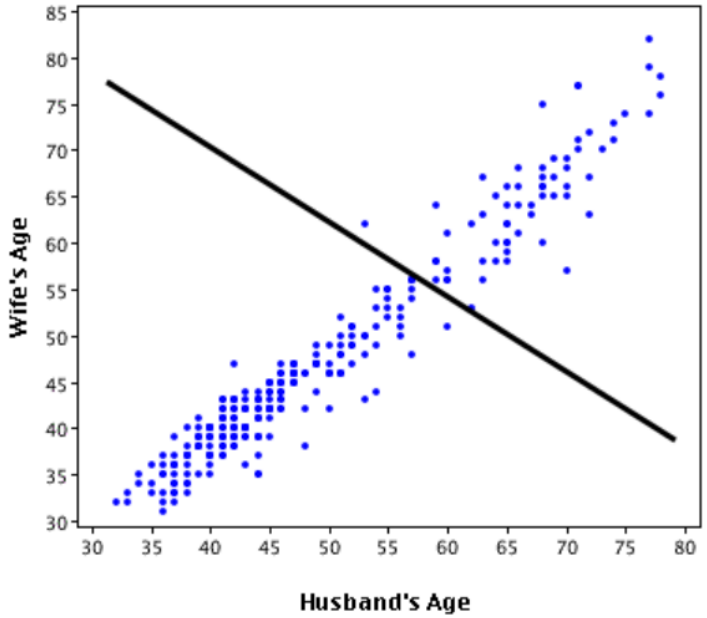


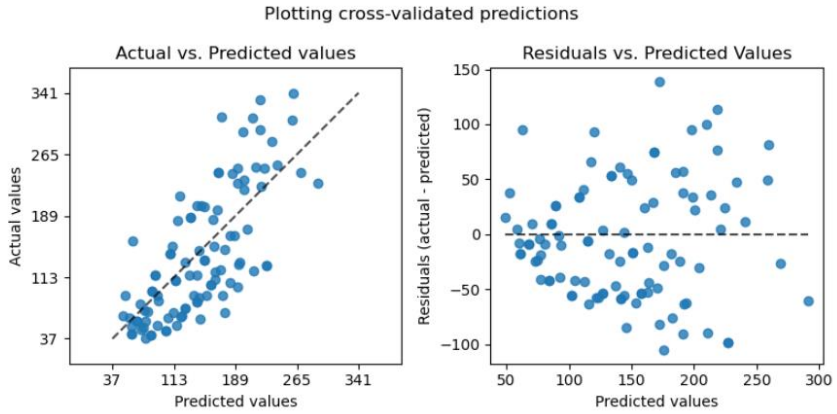
Metody:
.summary()

```
array([0.40164084, 0.49078604, 0.46763855, 0.47875   , 0.41495275,
       0.47448992, 0.44240534, 0.4262428 , 0.37862659, 0.49555404,
       0.47602527, 0.37936182, 0.40186752, 0.38349234, 0.49078604,
       0.49729195, 0.42151945, 0.46085436, 0.38349234, 0.38956118,
       0.47547504, 0.47641806, 0.49729195, 0.48774034, 0.38956118,
       0.41096026, 0.3818036 , 0.37606936, 0.42151945, 0.45990434,
       0.49643638, 0.44736766, 0.47514809, 0.45711726, 0.49555404,
       0.41495275, 0.47221076, 0.47804511, 0.40749027, 0.4860358 ,
       0.49763413, 0.47875   , 0.37503606, 0.46955453, 0.48135939,
       0.41096026, 0.49729195, 0.45460398, 0.46853901, 0.39414875,
       0.48774034, 0.45711726, 0.44736766, 0.38370878, 0.51788313,
       0.37862659, 0.45711726, 0.44240534, 0.37503606, 0.42416317,
       0.38370878, 0.38956118, 0.40186752, 0.47736295, 0.43750289,
       0.47221076, 0.46893189, 0.49729195, 0.48058266, 0.47875   ,
       0.39469136, 0.45711726])
```

Funkcje ewaluacyjne… i jedna metoda… i kilka wykresów

1. Funkcje ewaluacyjne scikit-learn
   a) r2_score()
   b) mean_poisson_deviance()
   c) mean_gamma_deviance()
2. Metoda .score()
   a) Wykonywana na obiekcie z dopasowanym modelem
   b) W zależności od modelu, różna metryka -> sprawdź dokumentację!
   c) Dla regresji Poissona i Gamma: D^2
3. Wykresy ewaluacyjne
   a) Funkcjonalność dostępna od scikit-learn v.1.2
   b) Zestawienia „actual-predicted” oraz „residuals-predicted”





$$D^2(y, \hat{y}) = 1 - \frac{\text{dev}(y, \hat{y})}{\text{dev}(y, y_{\text{null}})}$$