AKADEMIA GÓRNICZO-HUTNICZA
im. Stanisława Staszica w Krakowie

WYDZIAŁ ZARZĄDZANIA

Przetwarzanie i Analiza Danych w Pythonie

Autor: Wiktoria Szczypka
Tytuł ćwiczenia: Sprawozdanie
Z eksperymentów symulacyjnych

Kraków, 11.05.2020

1. Cel ćwiczenia:

   Tematem tego ćwiczenia są eksperymenty symulacyjne. Jego celem jest stworzenie modeli i wybranie najlepszego dla wygenerwoanych danych z zaburzeniami.

2. Propozycja rozwiązania zadania:

   Do rozwiązania powyższego zadania zostanie wykorzystana biblioteka numpy w celu generacji danych. Z wygenerowanych danych zostanie losowo wybrane 30 danych, a następnie w celu dopasowania dla nich odpowiednich modeli zostanie użyta biblioteka sklearn. Powstałe modele zostaną porównane przy pomocy błędów MAE, MSE, RMSE oraz współczynnika determinacji R^2. Do wizualizacji otrzymanych wyników posłuży biblioteka matplotlib.

3. Opis przebiegu ćwiczenia:

   3.1. Wygenerowanie danych z zaburzeniami.

   3.2. Stworzenie i weryfikacja trzech modeli.

   3.3. Wybór najlepszego z modeli.

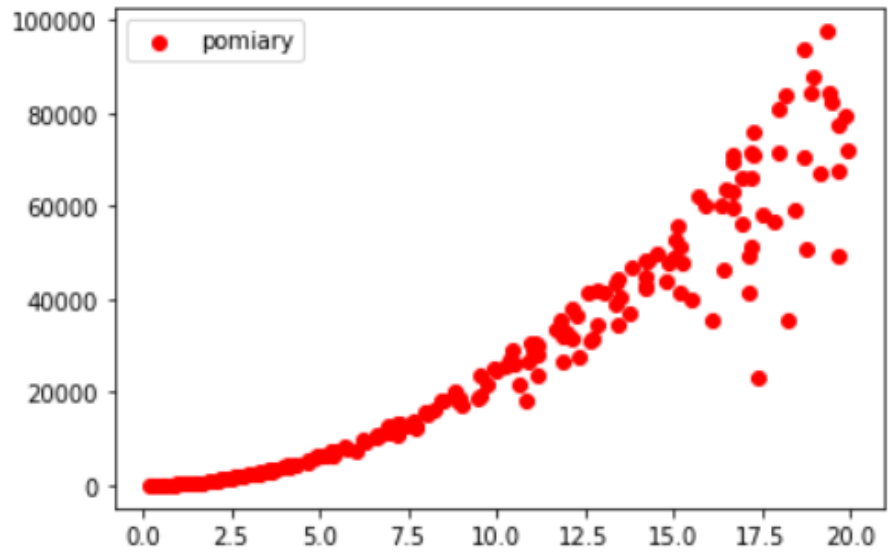
4. Przeprowadzenie badania:

   4.1. Generacja danych

   W celu niezmienności wyników ziarno generatora zostało ustawione na 19. X został wygenerowany jako 200 danych z rozkładu jednostajnego od 0 do 20, następnie zostały one posortowane. Zdefiniowano również funkcję, która przyjmuje dwa parametry x oraz n (długość x). Funkcja ta przypisuje każdemu x zdefiniowaną wartość. Do zaburzenia tych danych użyto funkcji cosinus, modelu hiperbolicznego oraz wielomianów, a także rozkładów: jednostajnego, normalnego, F oraz chi-kwadrat.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline

np.random.seed(19)

x = np.random.uniform(0,20,200)
x.sort()

def f(x,n):
    return 23 + 4*np.cos(x) + np.random.f(3, n) * (1/x**2) - \
        np.random.chisquare(3, n)*x**3 + np.random.uniform(250,269,n)*x**2 -\
        np.random.uniform(5,12,n)*x + np.random.normal(0,1,n)

y = f(x, len(x))
```

Poniżej przedstawiono wygenerowane dane:

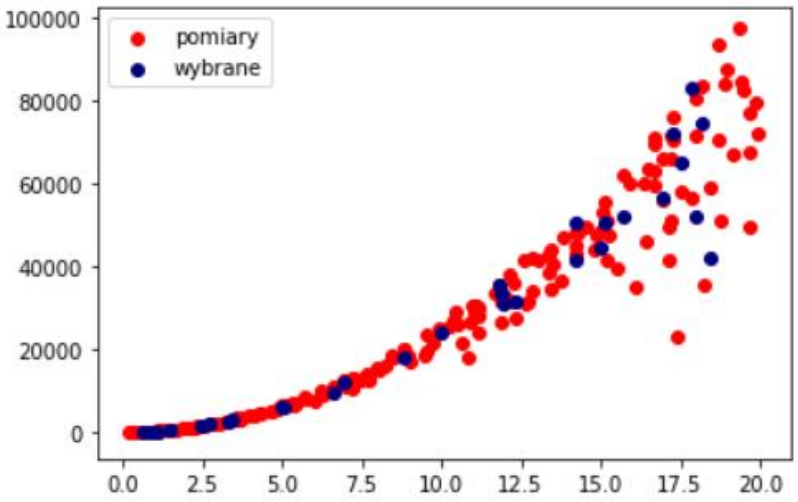


4.2. Wybranie losowo 30 danych

Za pomocą funckcji RandomState oraz shuffle wybrano 30 losowych danych, dla których tworzone będą modele.

```
rng = np.random.RandomState(0)
rng.shuffle(x)
x_wybr = np.sort(x[:30])
y_wybr = f(x_wybr, len(x_wybr))
x=np.sort(x)
```

Poniżej wybrane dane:



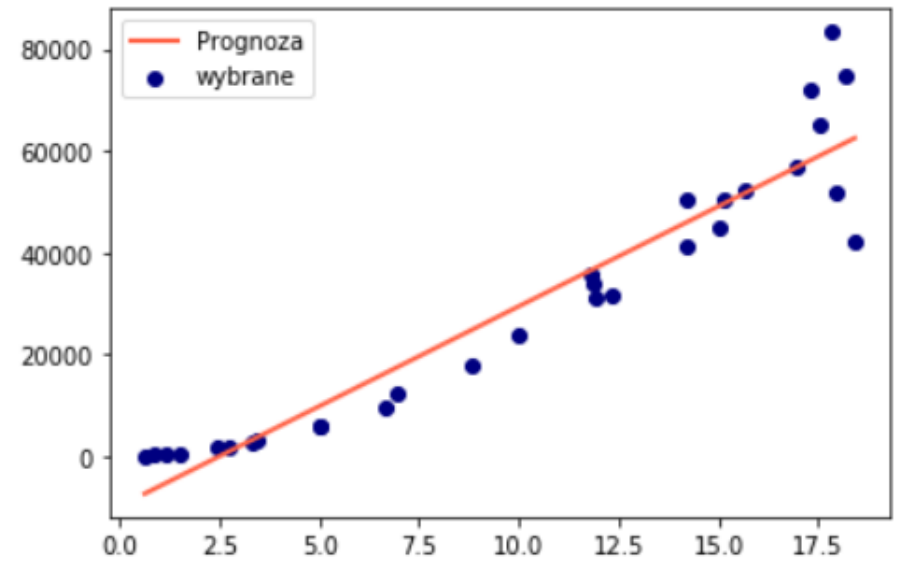
Można zauważyć, iż dane zostały wybrane losowo i dość dobrze obrazują rzeczywiste pomiary.

4.3. Stworzenie i weryfikacja modeli

Pierwszy przedstawiany model to regresja liniowa. Został stworzony przy pomocji funkcji LinearRegression.

```
X = x[:, np.newaxis]
X_wybr = x_wybr[:, np.newaxis]

reg = linear_model.LinearRegression()
reg.fit(X_wybr,y_wybr)
y_pred1 =reg.predict(X_wybr)
```

Poniżej dopasowana do wybranych danych linia regresji.



Poniżej przedstawione obliczenia w celu przedstawienia współczynników, błędów oraz współczynnika determinacji.

```
# Wspolczynniki
print("Wspolczynniki:\n", reg.coef_, reg.intercept_)
# Bledy
print("MAE: %.2f" % mean_absolute_error(y_wybr,y_pred1))
print("MSE: %.2f" % mean_squared_error(y_wybr,y_pred1))
print("RMSE: %.2f" %np.sqrt(mean_squared_error(y_wybr,y_pred1)))
# R^2
print("Wspolczynnik determinacji: %.2f" % r2_score(y_wybr, y_pred1))
```

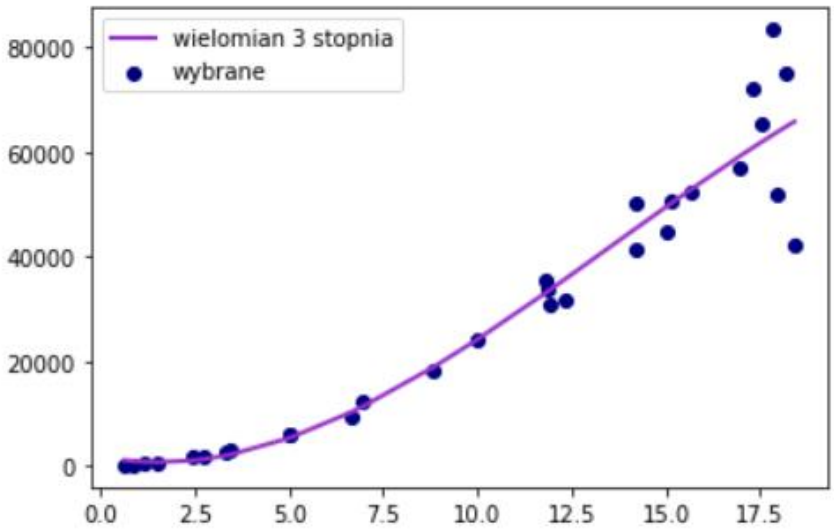
```
Wspolczynniki:
 [3924.18554556] -9785.304891654443
MAE: 5908.24
MSE: 64048417.82
RMSE: 8003.03
Wspolczynnik determinacji: 0.90
```

Na pierwszy rzut oka prosta y = -9785.3 + 3924.19x dobrze opisuje wybrane dane. Współczynnik determinacji, który wynosi 0.9 potwierdza to - wskazuje, iż model dobrze opisuje zmienność danych.

Drugim stworzonym modelem będzie dopasowany wielomian 3 stopnia. Został on stworzony za pomocą funkcji Pipeline oraz PolynomialFeatures.

```
degree = 3
model = Pipeline([('mpoly', PolynomialFeatures(degree)),
                  ('mlinear', linear_model.LinearRegression(fit_intercept=True))])
results = model.fit(X_wybr, y_wybr)
y_pred2 = model.predict(X_wybr)
```

Poniżej wizualizacja.



Można przypuszczać, iż model ten w lepszym stopniu wyjaśnia dane lepiej jest do nich dopasowany niż prosta regresji liniowej. Poniżej przedstawione obliczenia w celu przedstawienia współczynników, błędów oraz współczynnika determinacji.

```
print("Współczynniki:", model.named_steps['mlinear'].coef_)
# Bledy
print("MAE: %.2f" % mean_absolute_error(y_wybr,y_pred2))
print("MSE: %.2f" % mean_squared_error(y_wybr,y_pred2))
print("RMSE: %.2f" %np.sqrt(mean_squared_error(y_wybr,y_pred2)))
# R^2
print("Wspolczynnik determinacji: %.2f" % r2_score(y_wybr, y_pred2))
```

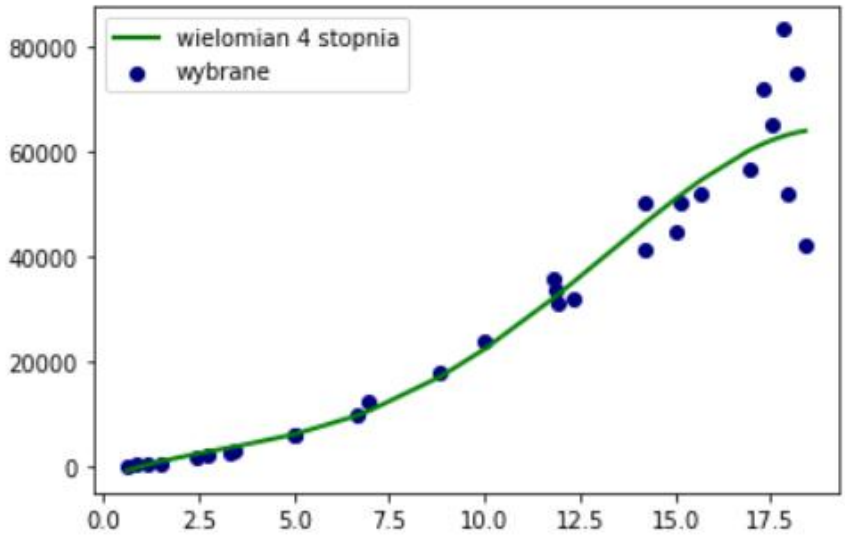
```
Współczynniki: [    0.         -1433.30752031   486.29669981   -11.93576434]
MAE: 3842.81
MSE: 48238052.17
RMSE: 6945.36
Wspolczynnik determinacji: 0.93
```

Można teraz z pewnością stwierdzić, iż model y = -1433.31x + 486.3x^2 – 11.94x^3 w lepszym stopniu wyjaśnia dane. Zarówno wartości błędów są mniejsze jak i współczynnik determinacji wynosi teraz, aż 0.93.

Stworzony zostanie teraz trzeci model, również będzie to wielomian, lecz stopnia 4. Został on stworzony analogicznie do poprzedniego.

```
degree = 4
model = Pipeline([('mpoly', PolynomialFeatures(degree)),
                  ('mlinear', linear_model.LinearRegression(fit_intercept=True))])
model.fit(X_wybr, y_wybr)
y_pred3 = model.predict(X_wybr)
```

Poniżej wizualizacja.



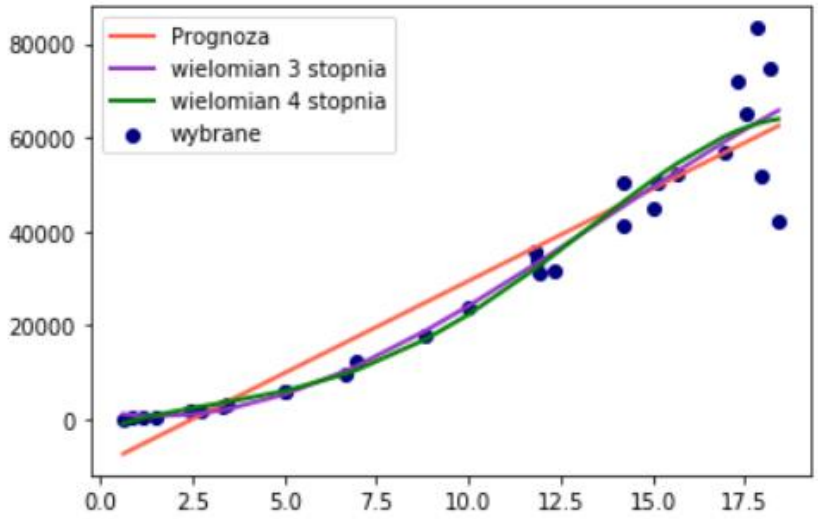
Na pierwszy rzut oka można stwierdzić, iż różnica między tym, a poprzednim modelem jest niewielka, lecz można się spodziewać, że będzie on lepszy. Poniżej przedstawione obliczenia w celu przedstawienia współczynników, błędów oraz współczynnika determinacji.

```
print("Współczynniki:",model.named_steps['mlinear'].coef_)
# Bledy
print("MAE: %.2f" % mean_absolute_error(y_wybr,y_pred3))
print("MSE: %.2f" % mean_squared_error(y_wybr,y_pred3))
print("RMSE: %.2f" %np.sqrt(mean_squared_error(y_wybr,y_pred3)))
# R^2
print("Wspolczynnik determinacji: %.2f" % r2_score(y_wybr, y_pred3))
```

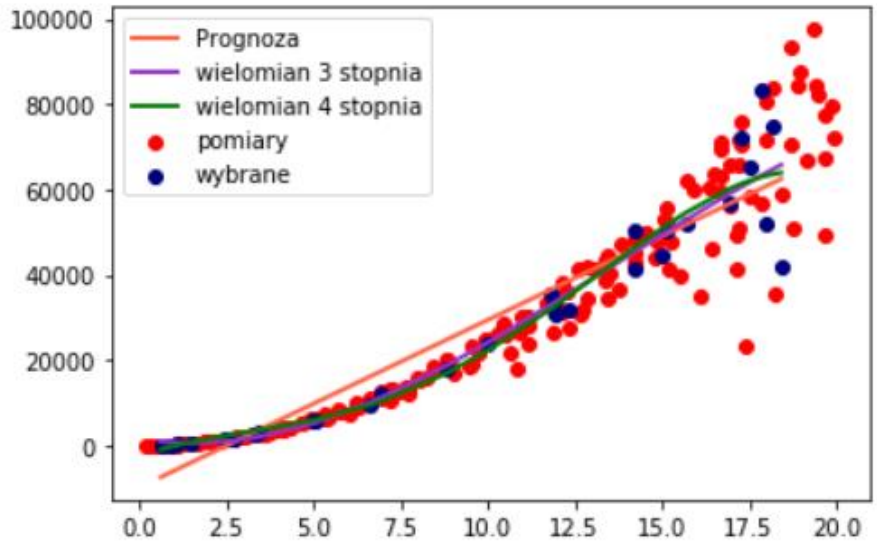
```
Współczynniki: [ 0.00000000e+00  2.92196295e+03 -5.55430629e+02  7.35980940e+01
 -2.24905178e+00]
MAE: 4009.10
MSE: 46806647.54
RMSE: 6841.54
Wspolczynnik determinacji: 0.93
```

Model y = 2921.96x – 555.43x^2 –73.6x^3 – 2.25x^4 jest nieco lepszy od poprzedniego – błędy są mniejsze, współczynnik determinacji nie zmienił się.

Poniżej przedstawione wszystkie modele wraz z wybranymi danymi.



Poniżej również wszystkie modele z wybranymi danymi oraz ze wszystkimi pomiarami.



4.4. Wnioski

Można jednoznacznie stwierdzić, iż modele wielomianowe okazały się być lepsze od regresji liniowej. Błędy w kolejnych dwóch modelach okazały się być mniejsze, a współczynnik determinacji większy.

Warto się jednak zastanowić czy konieczne jest tworzenie modelu opartego na wielomianie 4 stopnia, który jest bardziej skomplikowanym modelem niż ten 3 stopnia skoro błędy okazały się być niewiele mniejsze, a R^2 pozostało bez zmian. Jak widać na wykresie wielomiany te są bardzo podobne i prawie nakładają się. Jako najlepszy model wybieram, więc ten mniej skomplikowany – wielomian 3 stopnia.