Dominik Wróbel

Inżynieria oprogramowania i systemów

Informatyka, II stopień, 2018/19

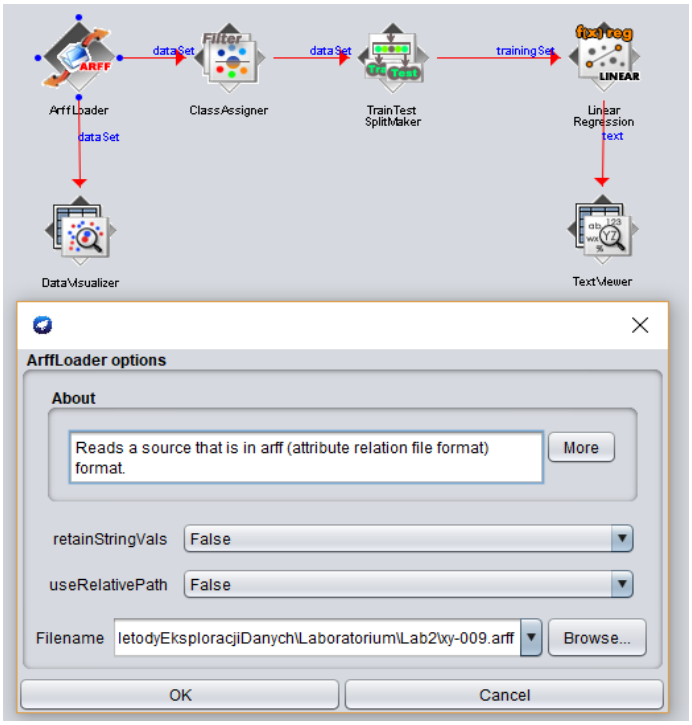
Metody eksploracji danych

Laboratorium 2 – 12.03.2019

Weka + Python + regresja

2.1

W oprogramowaniu Weka przy użyciu narzędzia KnowledgeFlow zbudowano model prostego procesu. Dane wejściowe skonfigurowano w bloczku ArffLoader, użyto zbiór xy-009.arf.

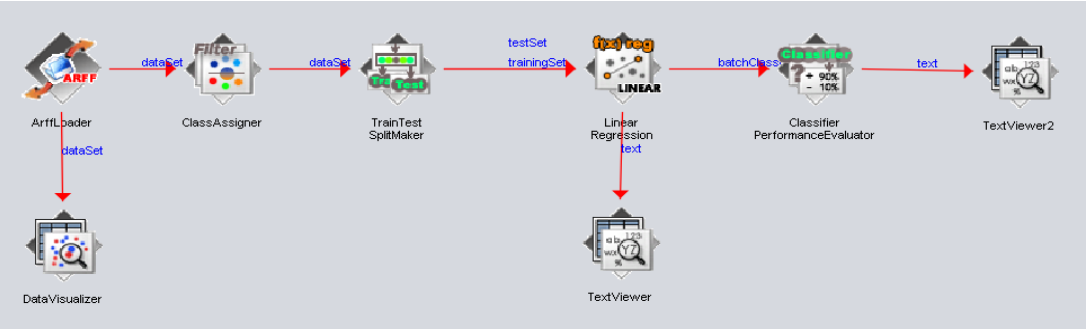


Zgodnie z oczekiwaniami otrzymano równanie prostej Y = -1.6615*X + 13.2711

```
=== Classifier model ===

Scheme:   LinearRegression
Relation: XY-(ellipse)-weka.filters.unsupervised.attribute.ClassAssigner-Clast


Linear Regression Model

Y =

     -1.6615 * X +
     13.2711
```

2.2

Do przepływu dołączono bloczek ClassifierPerformanceEvaluator, który oblicza wskaźniki informujące o jakości działania algorytmu.



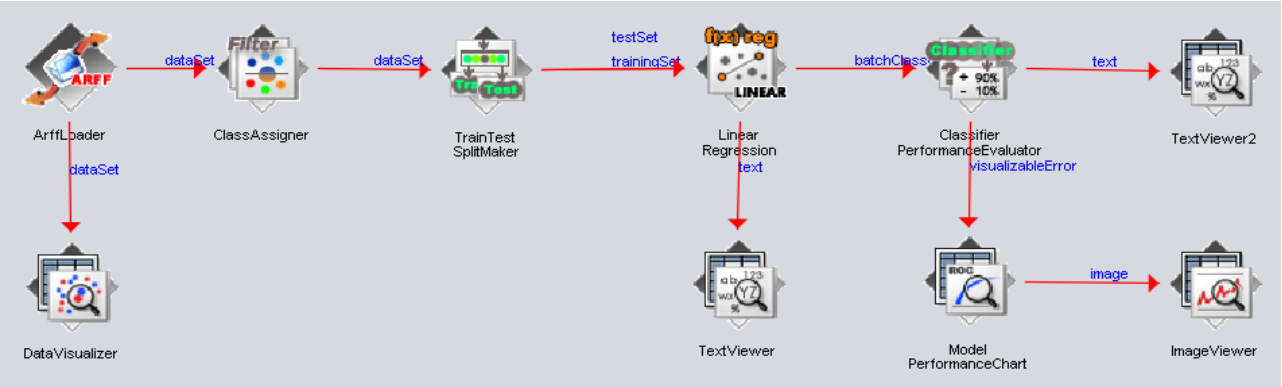
Bloczek ten dał następujące rezultaty:

```
=== Evaluation result ===

Scheme: LinearRegression
Options: -S 0 -R 1.0E-8 -num-decimal-places 4
Relation: XY-(ellipse)-weka.filters.unsupervised.attribute.ClassAssigner-Clast


Correlation coefficient                  0.9766
Mean absolute error                      0.826
Root mean squared error                  0.9689
Relative absolute error                 21.5026 %
Root relative squared error             21.5537 %
Total Number of Instances              340
```

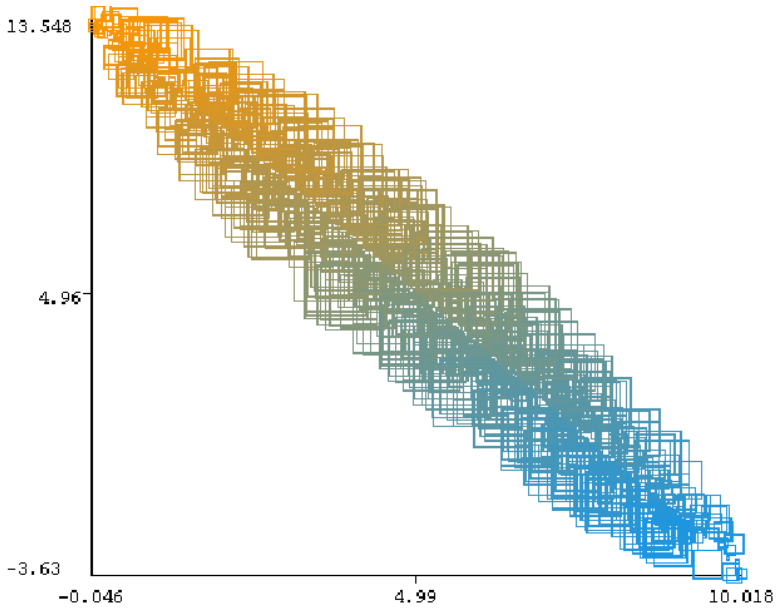
- Pierwszy ze wskaźników (współczynnik korelacji) informuje o tym jak silna zależność jest pomiędzy wyznaczonym wynikiem, a danymi wejściowymi, 1 oznacza bardzo silną pozytywną korelacje (wartości danych rosną i maleją razem), 0 oznacza brak jakiejkolwiek korelacji, a wartości ujemne oznaczyłyby, że korelacja zachodzi, ale jest negatywna, co oznacza, że wraz ze wzrostem danych wejściowych, dane wyznaczone przez algorytm maleją – tak samo odwrotnie.
- Kolejne wskaźniki określają stopień błędu, rozbieżności pomiędzy wyznaczonymi danymi, a danymi wejściowymi.
- Total number of instances, to liczba danych testowych. Bloczek TrainTestSplitMaker dzieli zbiór danych wejściowych (1000 próbek) na dane uczące i testowe, dane uczące stanowią 66%(zapisane w konfiguracji bloczka), czyli 660 próbek, pozostałe 340 próbek to dane testowe.

2.3

Do przepływu dodano bloczki w celu wizualizacji obliczonych błędów. Dodatkowo przetestowano dwa różne sposoby ewaluacji w przepływie.



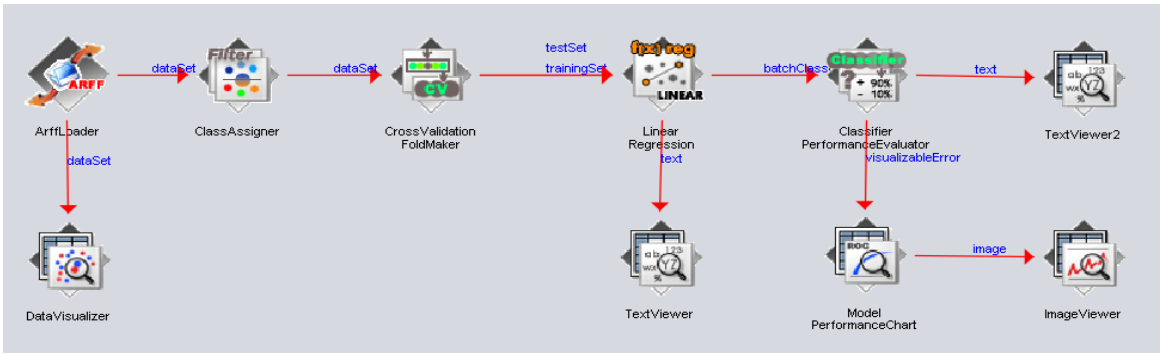
Correlation coefficient	0.9766
Mean absolute error	0.826
Root mean squared error	0.9689
Relative absolute error	21.5026 %
Root relative squared error	21.5537 %
Total Number of Instances	340



- Prostokąty układają się w elipsę, ponieważ odpowiadają one punktom danych wejściowych. Prostokąty są miarą odległości od prostej wyznaczonej przez algorytm, są tym większe im błąd jest większy.
- Po środku widoczna jest biała linia, ponieważ jest to miejsce w którym błędy są minimalnie małe lub zerowe. Wzdłuż prostej, punkty danych wejściowych pokrywają się z wyznaczonymi przez algorytm.

2.4

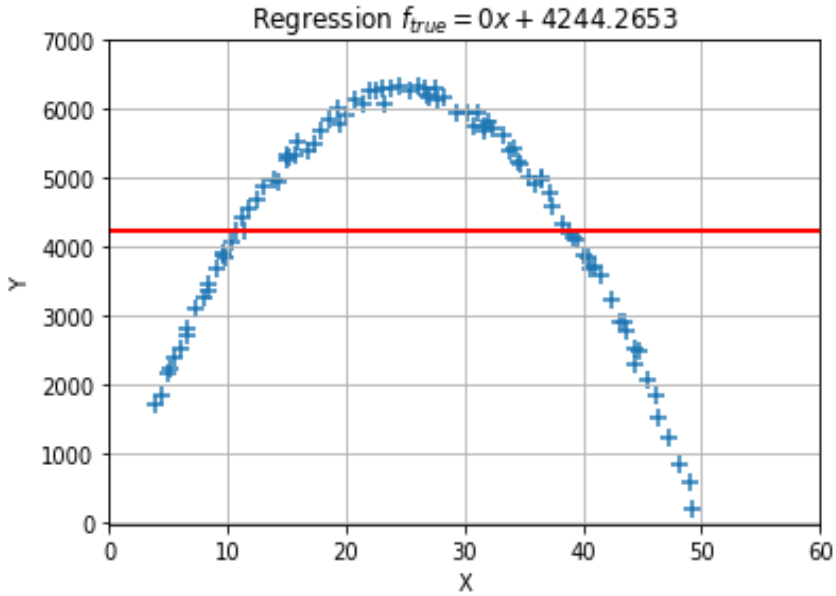
Drugi ze sposobów ewaluacji dał wyniki o nieco innej wartości:



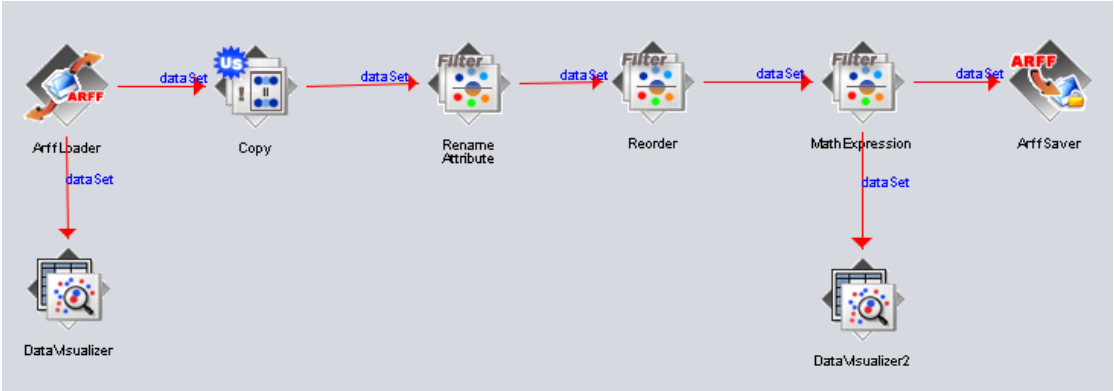
Correlation coefficient	0.975
Mean absolute error	0.825
Root mean squared error	0.9704
Relative absolute error	22.1431 %
Root relative squared error	22.1937 %
Total Number of Instances	1000

2.5

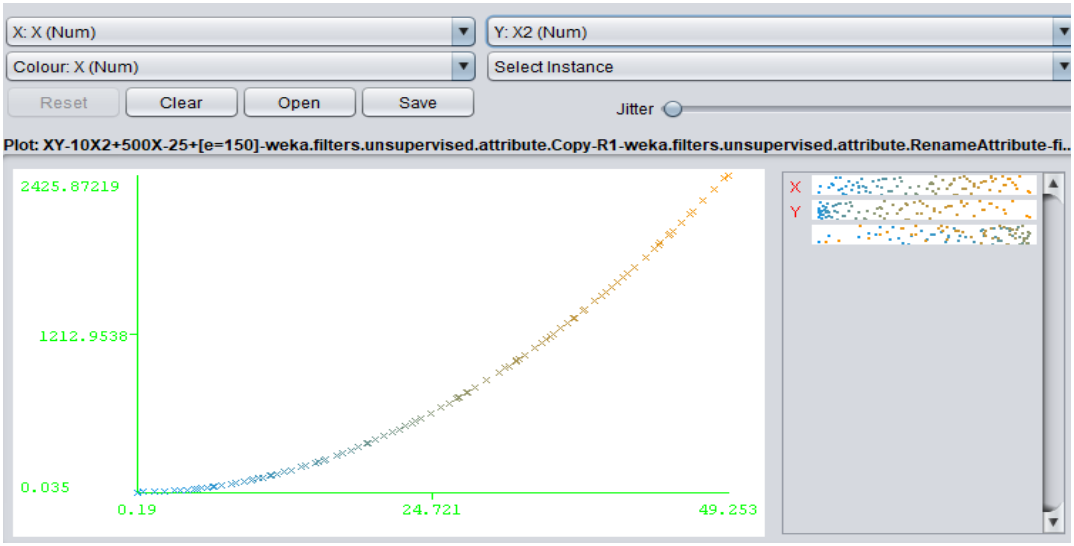
- W oprogramowaniu Weka wykonano regresje liniową dla danych z pliku xy-004.arff.
- Otrzymano równanie krzywej: Y = 0*X + 4244.2653
- Dane wraz z otrzymaną prostą przedstawia wykres poniżej:



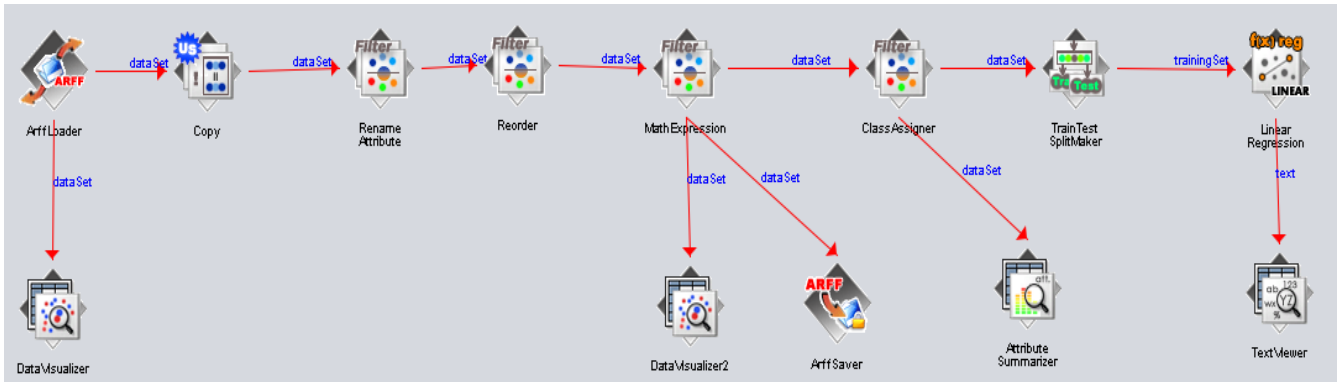
- Jak widać, wynik ten jest daleki od poprawnego, wprowadzono więc modyfikację poprzez rozszerzenie zbioru cech.



- Zbudowany przepływ sprawdzono przy pomocy wykresu w oprogramowaniu Weka oraz pliku zapisanego przez ArffSaver:



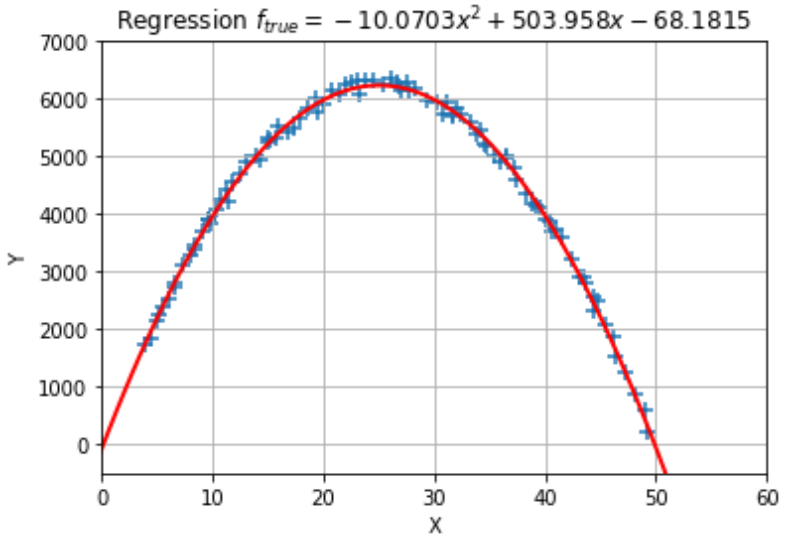
- Dodano komponenty odpowiedzialne za przeprowadzenie regresji



- Po wykonaniu algorytmu otrzymano następujący wynik

```
Linear Regression Model

Y =

      503.958   * X +
      -10.0703  * X2 +
      -68.1815
```

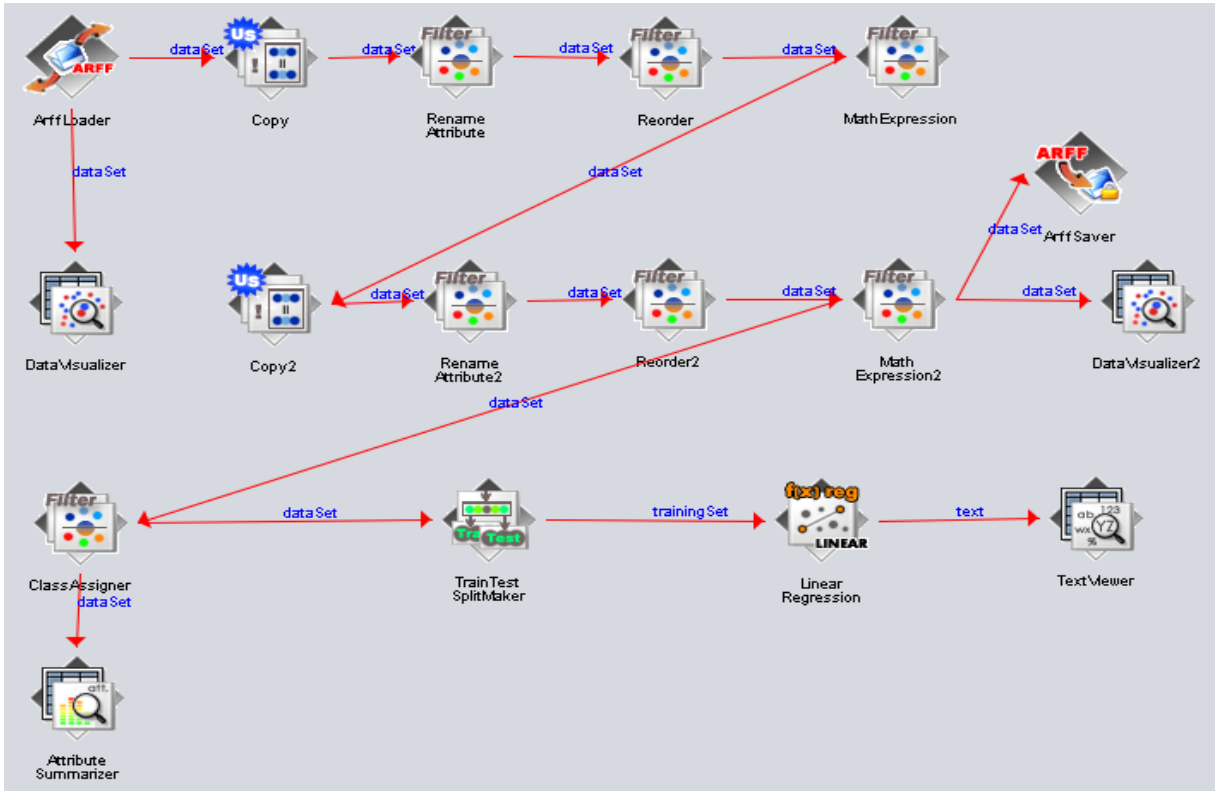
- Wykres otrzymanej funkcji przedstawiono na wykresie razem z danymi wejściowymi



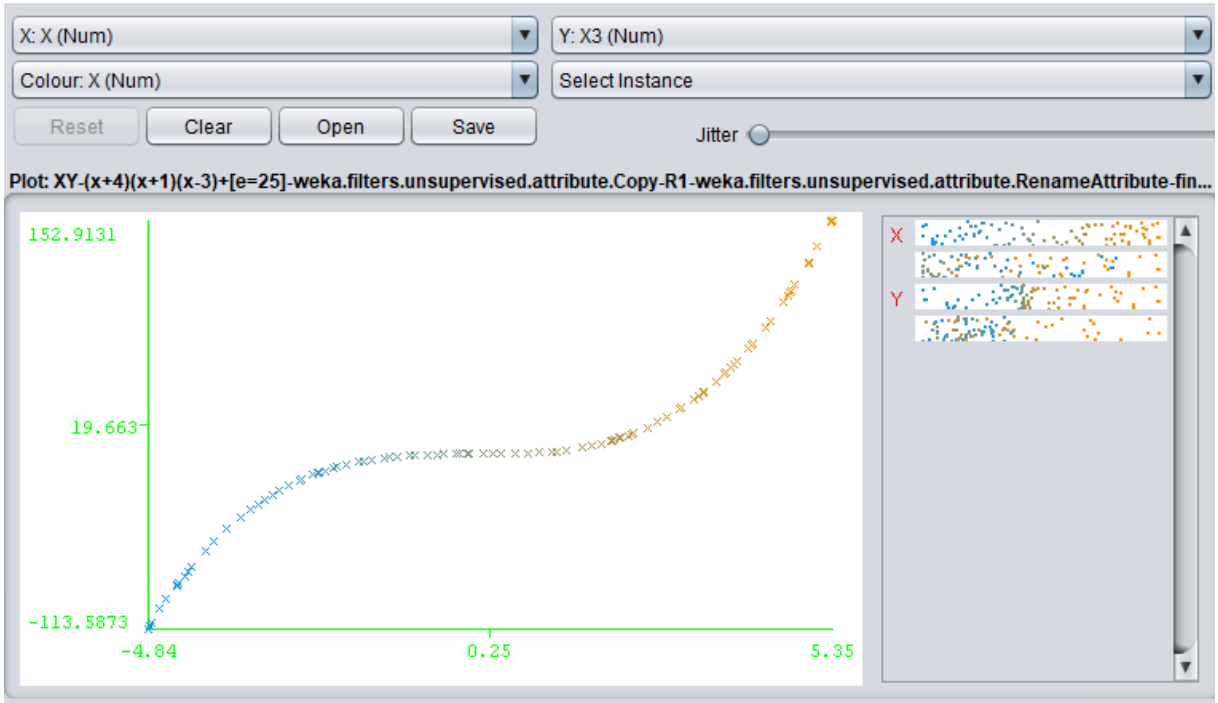
Jak widać rozszerzenie zbioru cech przyniosło znaczną poprawę jakości regresji.

2.6

- Wprowadzono cechy trzeciego stopnia analogicznie jak w poprzednim zadaniu. Tor rozszerzono o kolejną kopię argumentu X, zmianę argumentów według kolejności X,X2,X3,Y oraz podniesienie X3 do trzeciej potęgi. (Przepływ na kolejnej stronie)



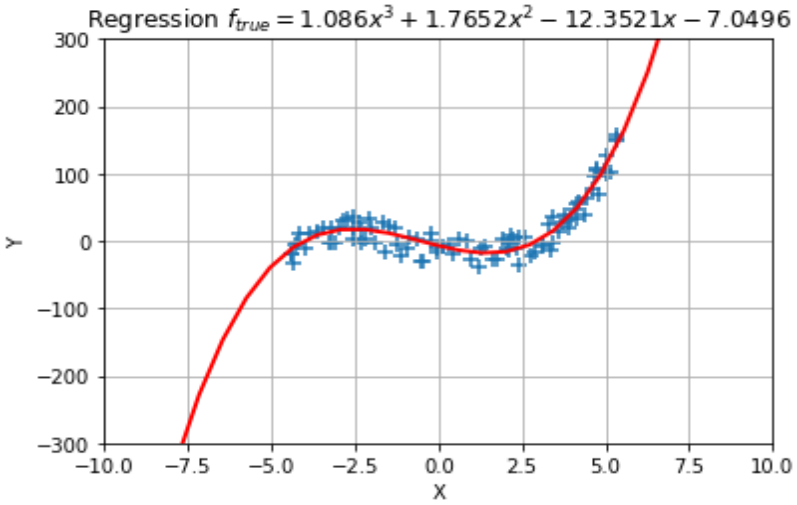
Poprawność obliczania X3 do trzeciej potęgi sprawdzono w bloczku DataVisualizer2, gdzie otrzymano wykres:



- Znalezione współczynniki regresji to

```
Linear Regression Model

Y =

    -12.3521 * X +
      1.7652 * X2 +
      1.086  * X3 +
     -7.0496
```

- Dane wejściowe wraz z otrzymaną krzywą przedstawia wykres

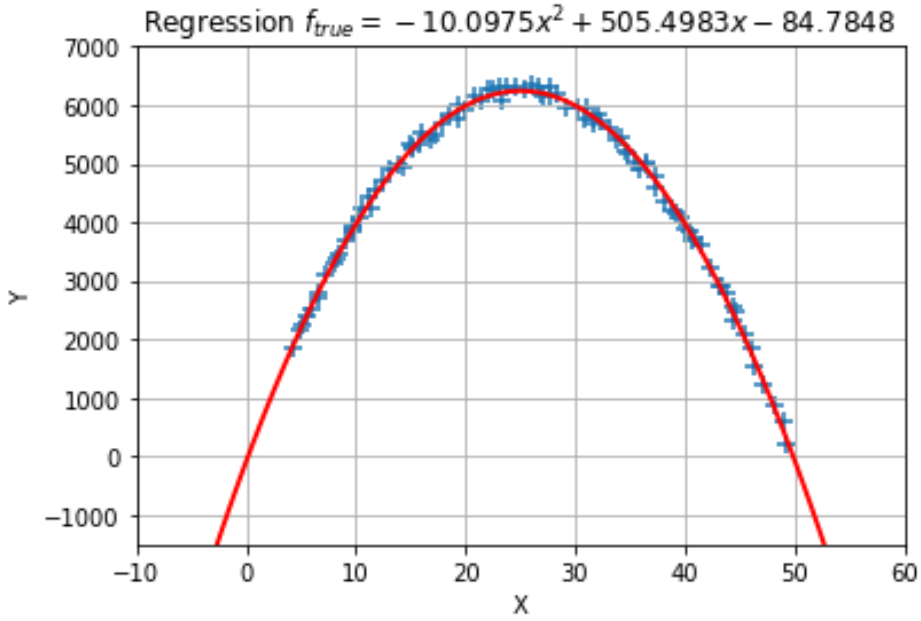


2.7

- W języku Python utworzono skrypt obliczający współczynniki na postawie regresji

```
inp =  StringIO(data)
x, x2, y = np.loadtxt(inp, delimiter=',', usecols=(0, 1, 2), unpack=True,skiprows=7)


plt.scatter(x,y,s=80, marker='+')

features= np.stack((x,x2),axis=-1)
regr = linear_model.LinearRegression()


regr.fit(features, y)

print('Coefficients: ', regr.coef_, ' Intercept: ',regr.intercept_)
```

- Otrzymane równanie regresji oraz wykres przedstawione są na poniższym wykresie.



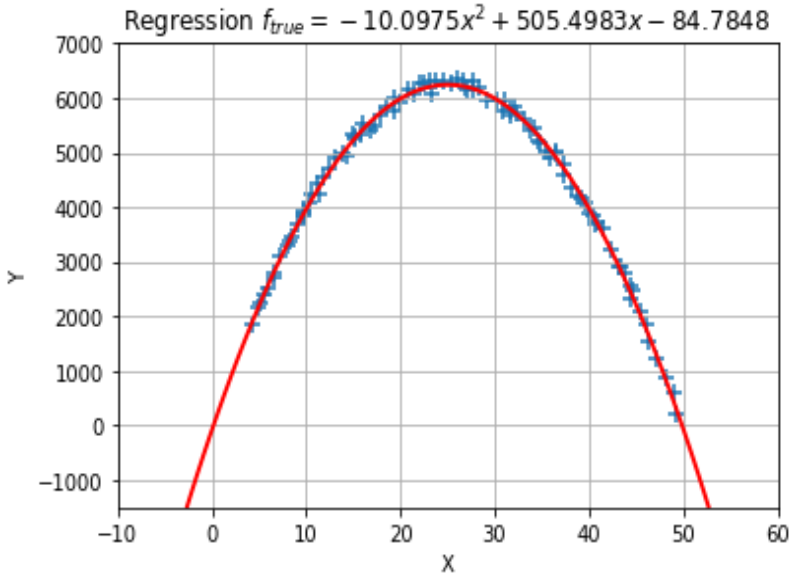
2.8

W tym zadaniu wykonano te same czynności co w zadaniu 2.7, tym razem nie czytano jednak od razu zmodyfikowanych danych, a wykorzystano dane wejściowe z plików.

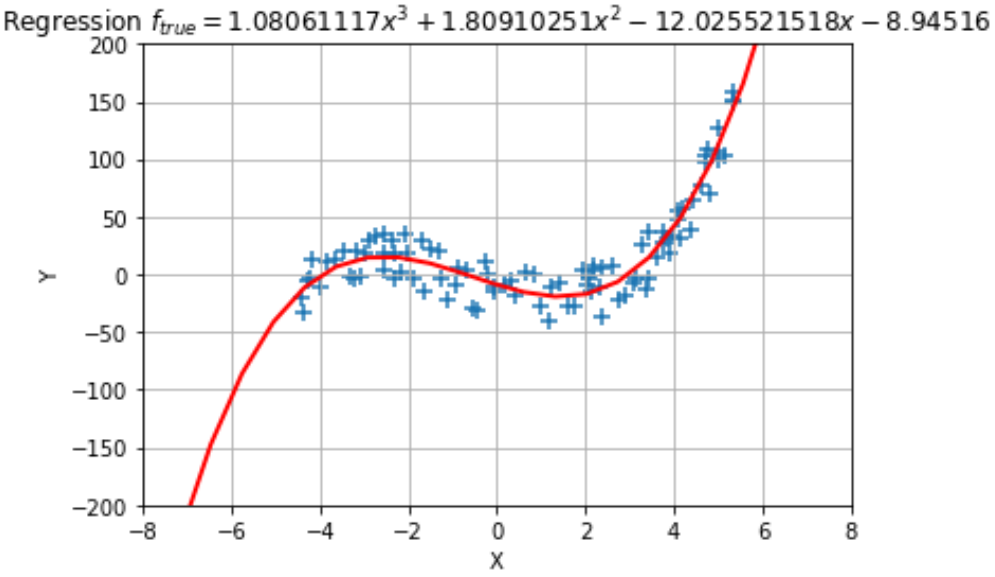
- Utworzono skrypt Python obliczający równanie regresji dla pliku xy-004.arff

```
inp =  StringIO(data)
x, y = np.loadtxt(inp, delimiter=',', usecols=(0, 1), unpack=True,skiprows=7)


plt.scatter(x,y,s=80, marker='+')

features= np.stack((x,x*x),axis=-1)
regr = linear_model.LinearRegression()


regr.fit(features, y)

print('Coefficients: ', regr.coef_, ' Intercept: ',regr.intercept_)
```

Zgodnie z oczekiwaniami, otrzymano te same wyniki jak w zadaniu 2.7



- Następnie wszystkie czynności powtórzono dla pliku xy-005.arff. Wykres i równanie regresji są przedstawione poniżej




```
inp =  StringIO(data)
x, y = np.loadtxt(inp, delimiter=',', usecols=(0, 1), unpack=True,skiprows=7)


plt.scatter(x,y,s=80, marker='+')

features= np.stack((x,x**2,x**3),axis=-1)
regr = linear_model.LinearRegression()


regr.fit(features, y)

print('Coefficients: ', regr.coef_, ' Intercept: ',regr.intercept_)
```