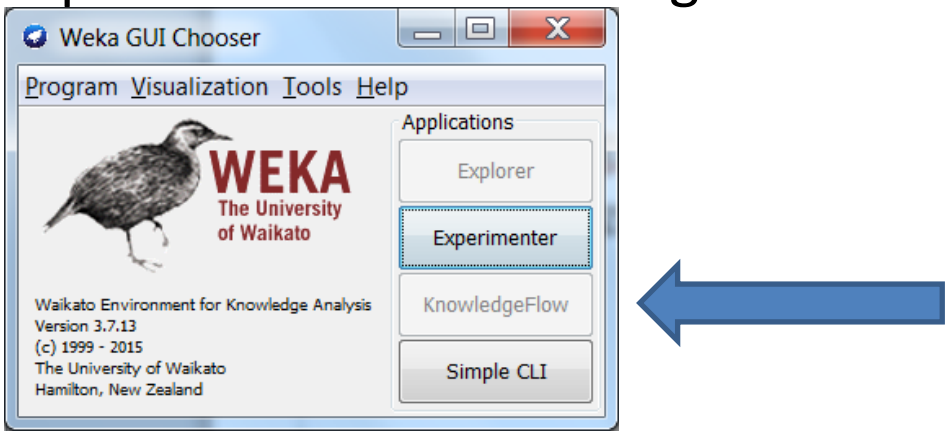
Metody eksploracji danych
Laboratorium 2

Weka + Python + regresja

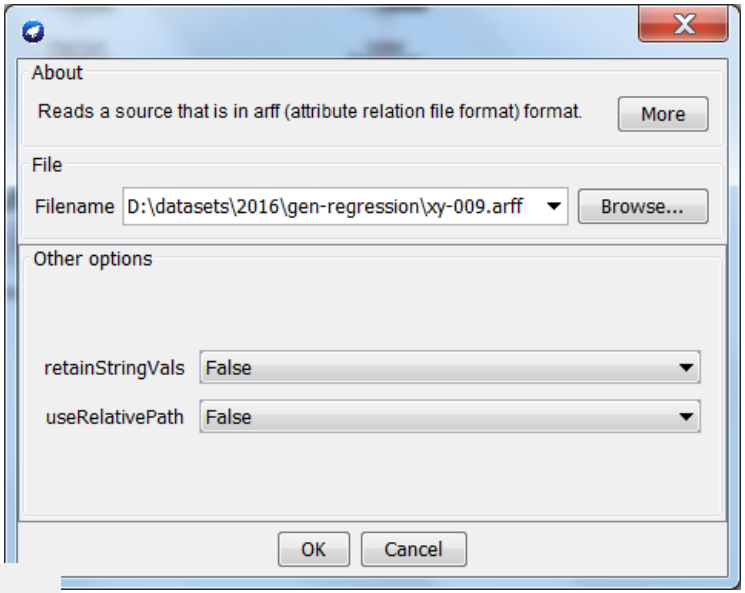
KnowledgeFlow

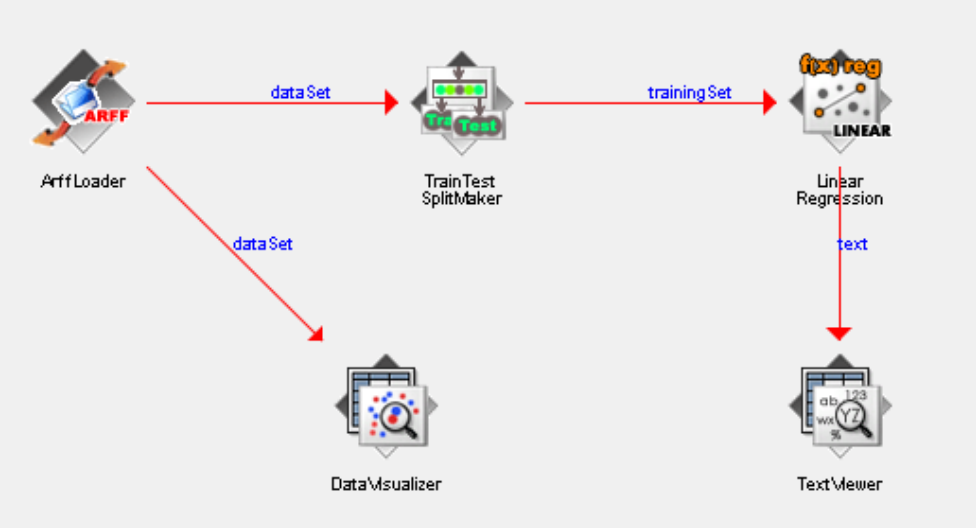
- KnowledgeFlow pozwala na zdefiniowanie procesu przetwarzania danych
- Komponenty realizujące poszczególne czynności można konfigurować, np. podać nazwę pliku do odczytu, parametry operacji lub algorytmu
- Edytor zapewnia poprawność połączeń
- Uruchom Weka
- Uruchom z poziomu okna startowego



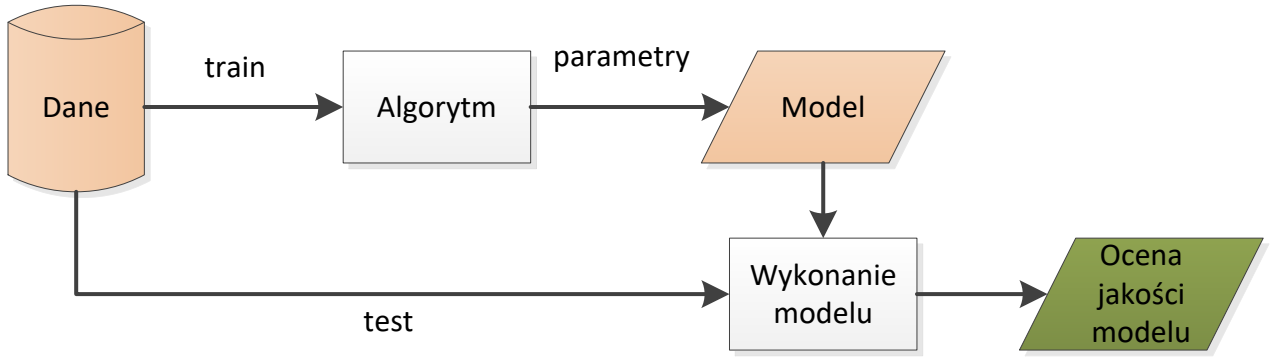
2.1 KnowledgeFlow – prosty proces

- Spróbuj zdefiniować pokazany proces
- Skonfiguruj ArffLoader tak, aby czytał zbiór danych **xy-009.arff**
- Kliknij prawym klawiszem na źródłowym komponencie, aby utworzyć połączenia
- Po uruchomieniu procesu ▶ możesz kliknąć na komponent i wybrać prawym klawiszem dostępne polecenia





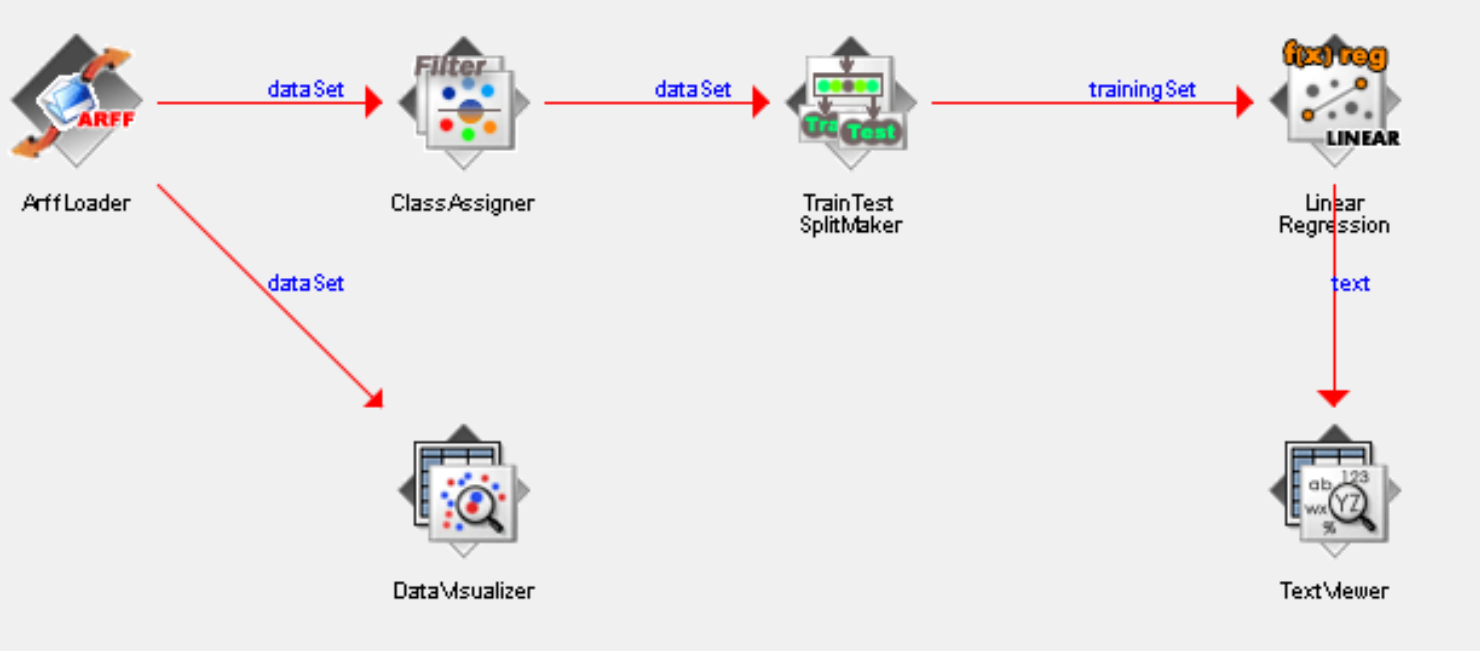
KnowledgeFlow - po co TrainTestSplitMaker?



- Wyznaczanie modelu nie powinno ograniczać się wyłącznie do uczenia. Cały proces obejmuje także jego ocenę przeprowadzaną z użyciem innych danych niż użyte do uczenia.
- Edytor KnowledgeFlow nie pozwala na dostarczenie całego zbioru na wejście algorytmu uczenia (musimy dostarczyć trainingSet)

KnowledgeFlow – poprawiony proces

- Po uruchomieniu pojawiają się błędy? Algorytm nie wie, który atrybut jest wyjściowy.
- Zmień workflow



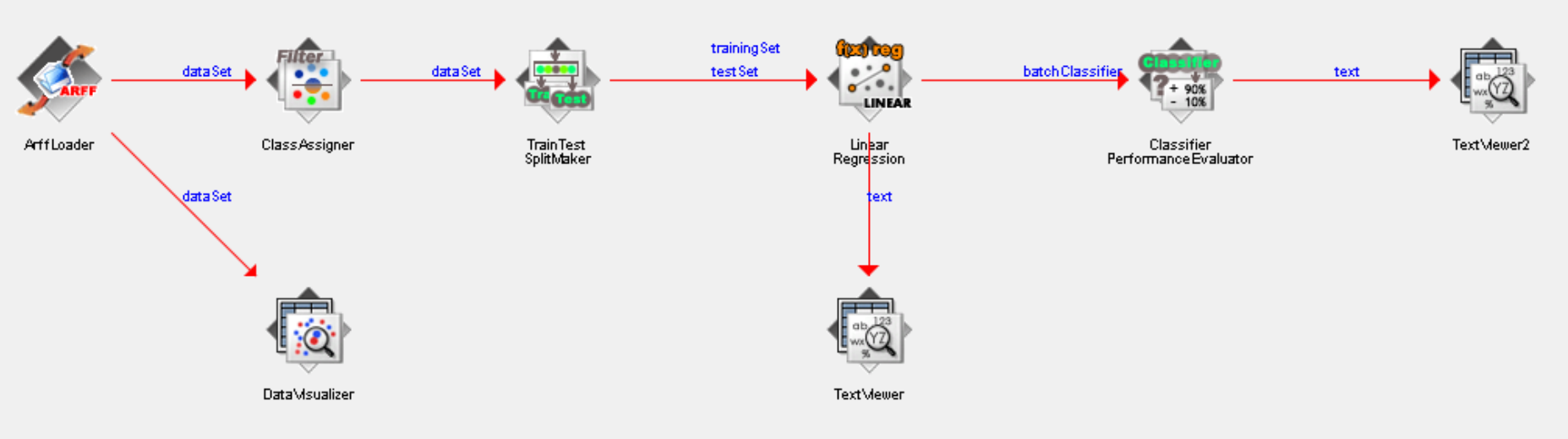
✓ Oczekiwany wynik:

Linear Regression Model

Y = -1.6615 * X + 13.2711

2.2 KnowledgeFlow – dodajemy ocenę

- Rozszerz workflow
- Nie zapomnij dołączyć przepływu testSet



KnowledgeFlow – wyniki

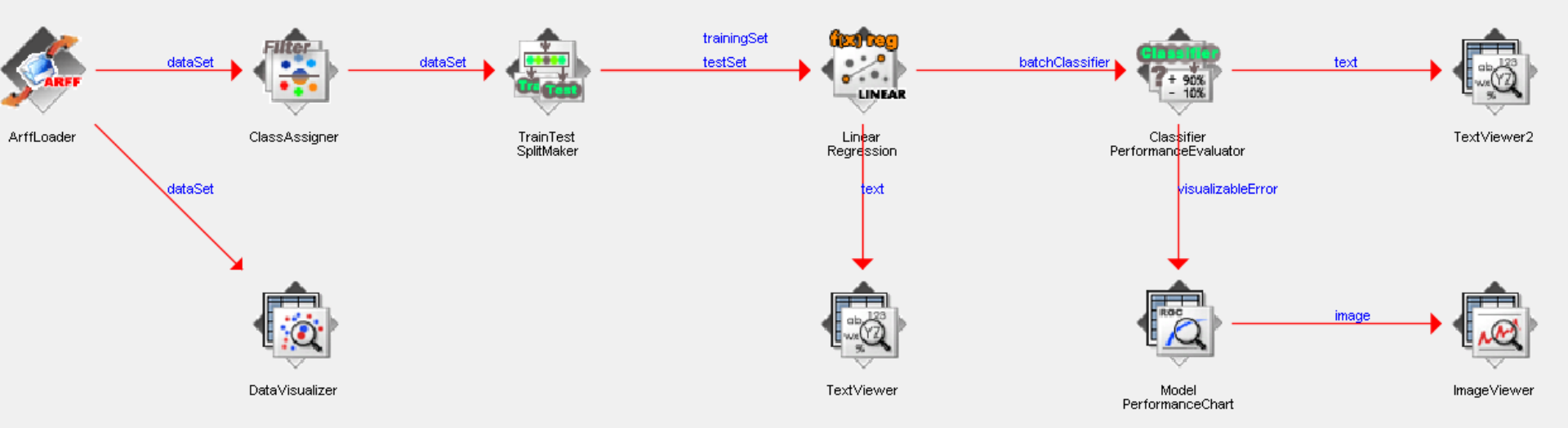
Text

```
=== 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
```

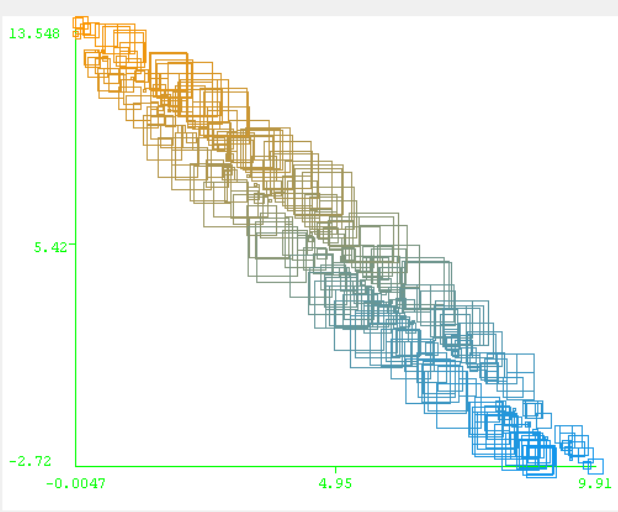
- Sprawdź, co oznaczają poszczególne wskaźniki: http://stats.stackexchange.com/questions/131267/how-to-interpret-error-measures-in-weka-output
- Dlaczego Total Number of Instances = 340 ?

2.3 KnowledgeFlow – dodatkowa wizualizacja

- Dodaj kolejne elementy i uruchom.

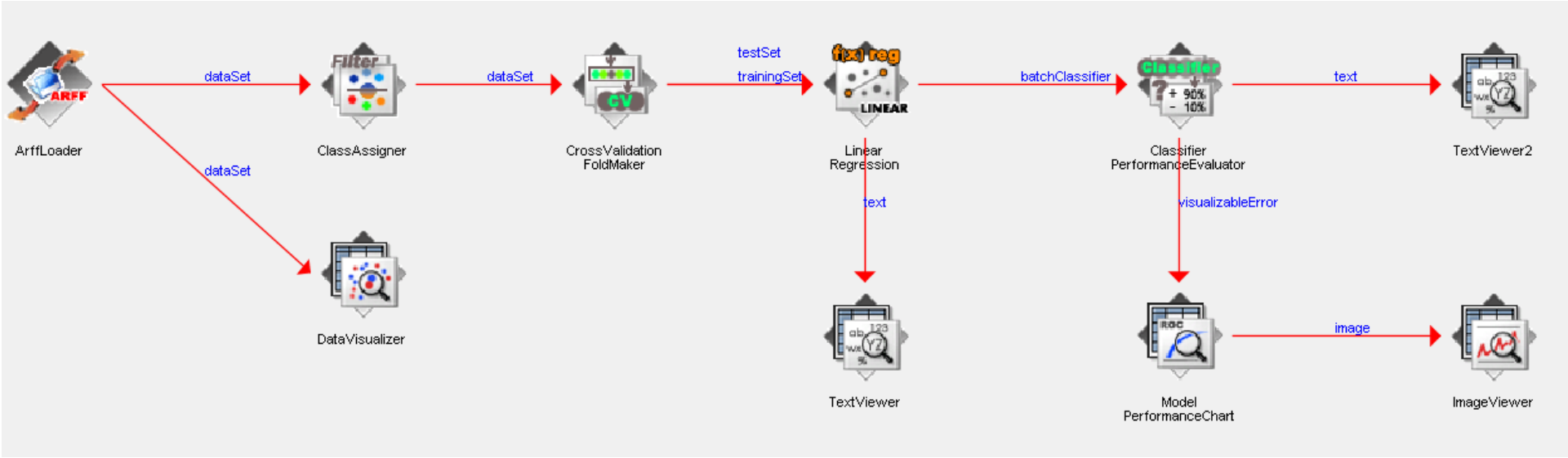


- Wyświetl wynik
- Dlaczego prostokąty oznaczające błędy układają się w elipsę?
- Skąd biały pasek pośrodku?



2.4 KnowledgeFlow – zmień sposób ewaluacji

- Użyj elementu CrossValidationFoldMaker



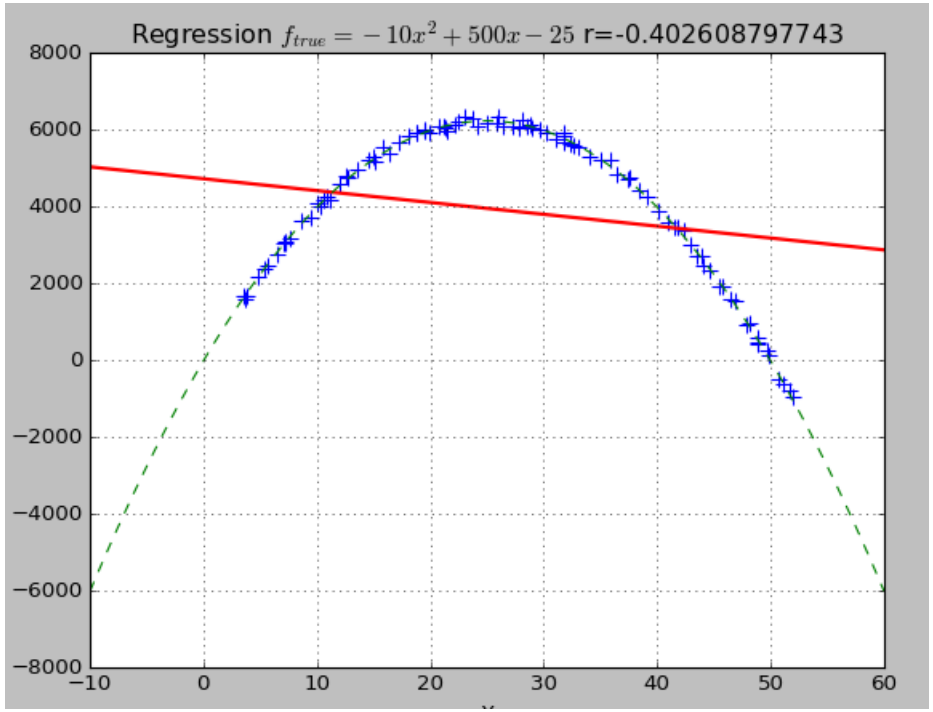
- CV (cross validation) polega na podziale zbioru na k (k=10) podzbiorów
- Uczenie i testowanie zachodzi w k iteracjach
  - k-1 części (folds) użyte do uczenia
  - 1 część użyta do testowania
- W sumie: do uczenia i walidacji użyte są wszystkie instancje
- Obliczane są średnie wyniki

Text

```
=== 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                  0.975
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 Rozszerzanie zbioru cech

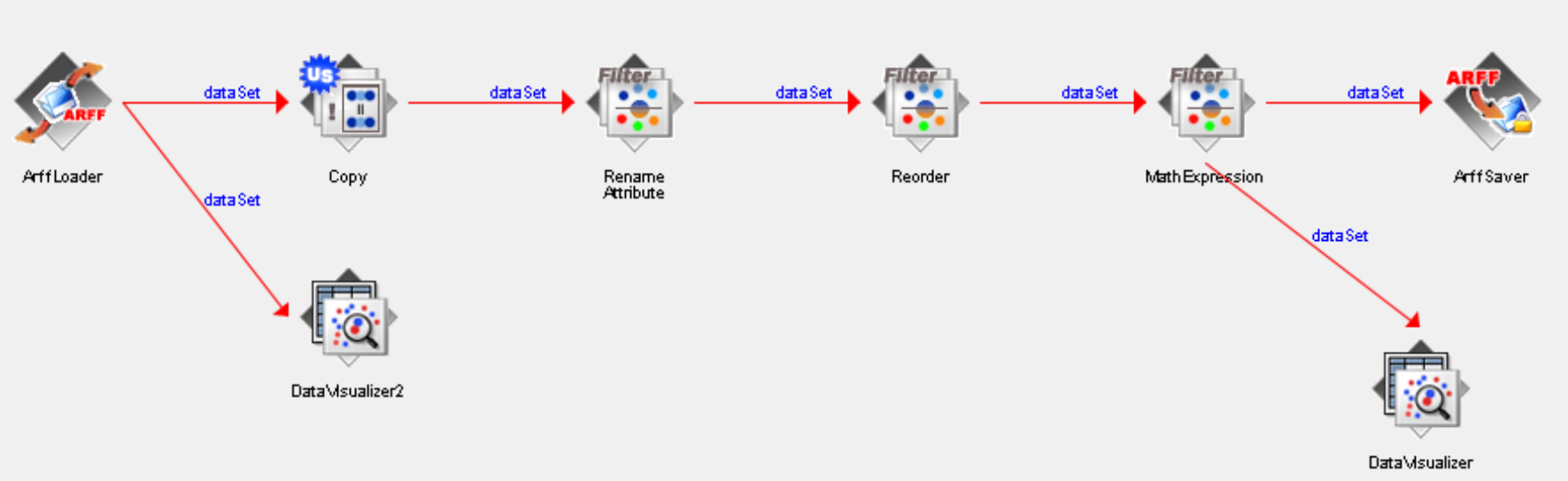
- Przetwarzany plik: **xy-004.arff**
- Uruchom zadanie regresji i odczytaj równanie krzywej
- Wyświetl dane + przebieg funkcji za pomocą odpowiednio zmodyfikowanego skryptu w języku Python
- Oczekiwany rezultat:



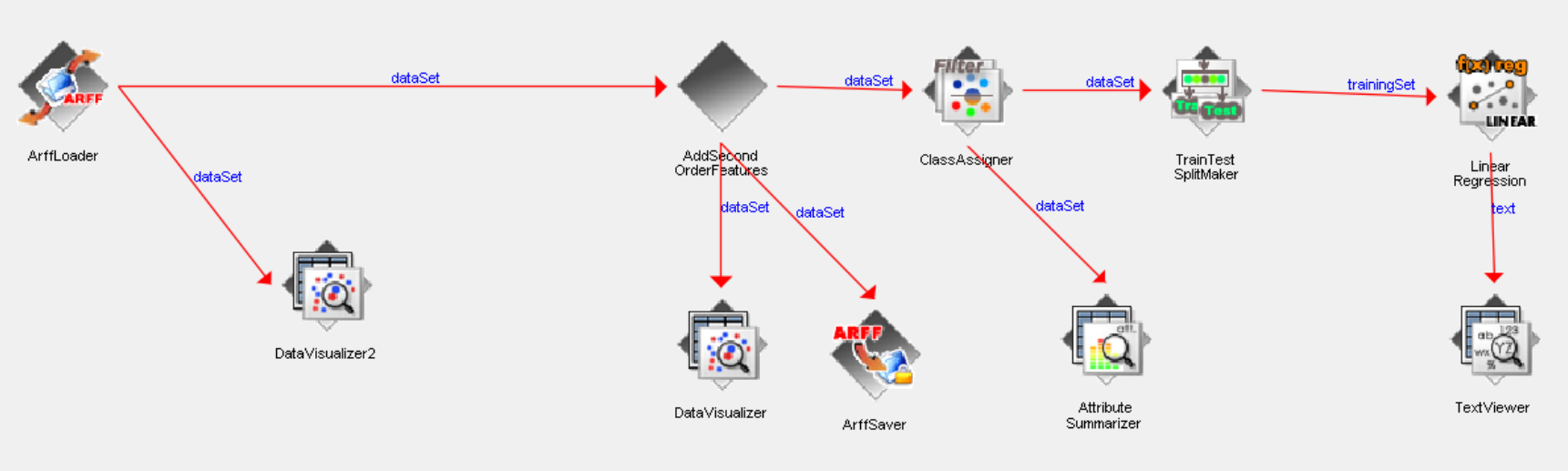
Rozszerzanie zbioru cech

Zimplementuj poniższy proces konfigurując odpowiednio kolejne operacje:

- Skopiowanie atrybutu X
- Zmiana nazwy na X2
- Zmiana porządku tak, aby Y był ostatni
- Podniesienie X2 do kwadartu
- Zapis do pliku (aby sprawdzić poprawność).
  Wybierz np. nazwę **xy-004-01.arff**



Dodaj komponenty odpowiedzialne za przeprowadzenie regresji



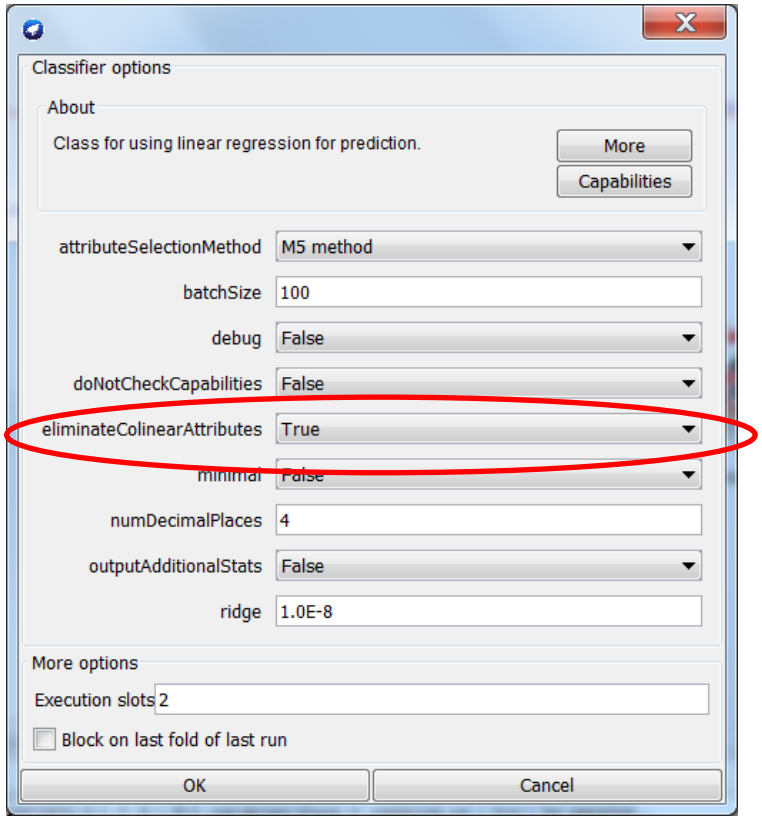
Uwaga romb to był zgrupowany proces. **W tej wersji Weka nie działa – proszę pominąć**

Uruchom i sprawdź wynik.

Jeśli ma on postać X2 = a*X → Źle

Jeśli ma postać Y = stała → Źle, ale to kwestia konfiguracji algorytmu

Konfiguracja algorytmu regresji



- Zmień parametr na false
- Spójrz na wykres X/X2 W tym zakresie atrybuty są bliskie liniowym.

✓ Równanie regresji powinno mieć współczynniki bliskie rzeczywistej funkcji

✓ Sprawdź błędy walidacji. Błąd względny powinien być kilkuprocentowy

✓ Użyj kodu Pythona do narysowania danych oraz zależności x->y

2.6 Wprowadź cechy 3 stopnia

- Zbuduj KnowledgeFlow dla pliku **XY-005.arff**
- Wprowadź cechę 3 stopnia (x^3)
- Znajdź współczynniki regresji
- Wyświetl dane i wyznaczoną krzywą posługując się skryptem w języku Python

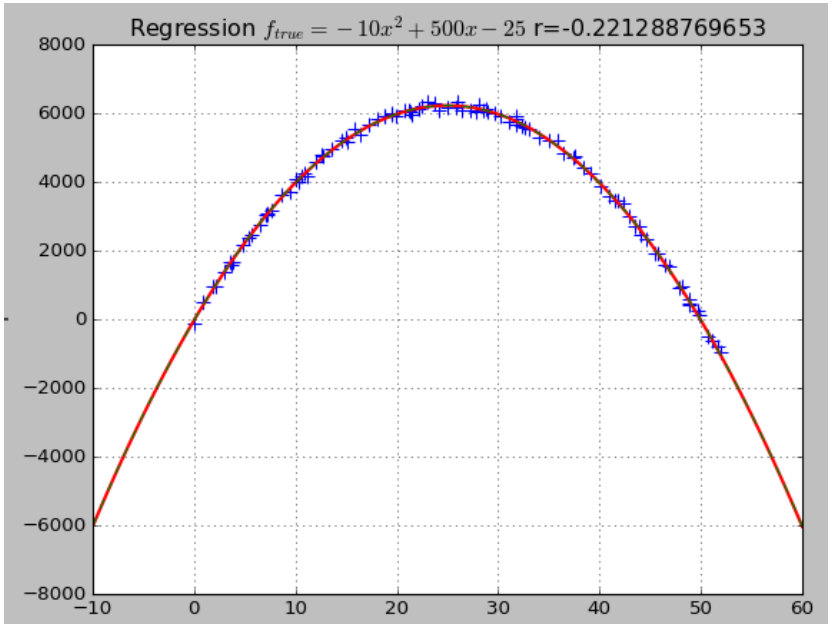
2.7 Rozwiązanie w języku Python

- Przetwarzamy **xy-004.arff:** zapisz w KnowledgeFlow plik z dodatkowym atrybutem X2, np. jako xy-004a.arff
- Skopiuj zawartość pliku do skryptu Pythona lub wczytaj z pliku

  x, x2, y = np.loadtxt(inp, delimiter=',', usecols=(0, 1, 2), unpack=True,skiprows=?)
- Utwórz dwuwymiarową macierz cech:

  features = np.stack((x,x2),axis=-1)
- Dopasuj krzywą

  regr = linear_model.LinearRegression()
  regr.fit(features, y)
- Wyświetl wynik i równanie regeresji



2.8 Przetwarzanie cech w języku Python

- 2.8.1 Przetwarzamy **xy-004.arff:** wczytaj dane z oryginalnego pliku

  x, y = np.loadtxt(inp, delimiter=',', usecols=(0, 1, 2), unpack=True,skiprows=?)
- Utwórz dwuwymiarową macierz cech:
  features = np.stack((x,x*x),axis=-1)
- Dopasuj krzywą
  regr = linear_model.LinearRegression()
  regr.fit(features, y)
- Wyświetl wynik i podaj równanie regresji
- 2.8.2 Analogicznie wyznacz model dla **xy-005.arff**

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

Parametry modelu są umieszczone w tablicy coef_ i zmiennej intercept_:

fy= regr3.coef_[0]* fx + regr3.coef_[1]* fx*fx + + regr3.coef_[2]* fx*fx*fx + regr3.intercept_

2.9 Sieć neuronowa?

- Identyczny model można zaimplementować jako sieć neuronową
- Uruchom poniższy kod dla xy-005.arff
- Dobierz eksperymentalnie parametr learning rate dla RMSProp oraz epochs dla fit.
- Skomentuj architekturę sieci i liczbę parametrów

```
x, y = np.loadtxt(inp, delimiter=',', usecols=(0, 1), unpack=True,skiprows=0)
X = np.stack((x,x**2,x**3),axis=-1)
tf.random.set_seed(1)
model = models.Sequential()
model.add(layers.InputLayer(input_shape=(X.shape[1],)))
model.add(layers.Dense(1))
model.summary()
model.compile(optimizer=tf.keras.optimizers.RMSprop(???), loss='mse', metrics=['mse','mae'])
hist = model.fit(X,y,epochs=???,verbose=1)

y_pred = model.predict(X)

plt.scatter(x,y)
plt.plot(x,(x+4)*(x+1)*(x-3),c='g')
plt.plot(x,y_pred,c='r')
```