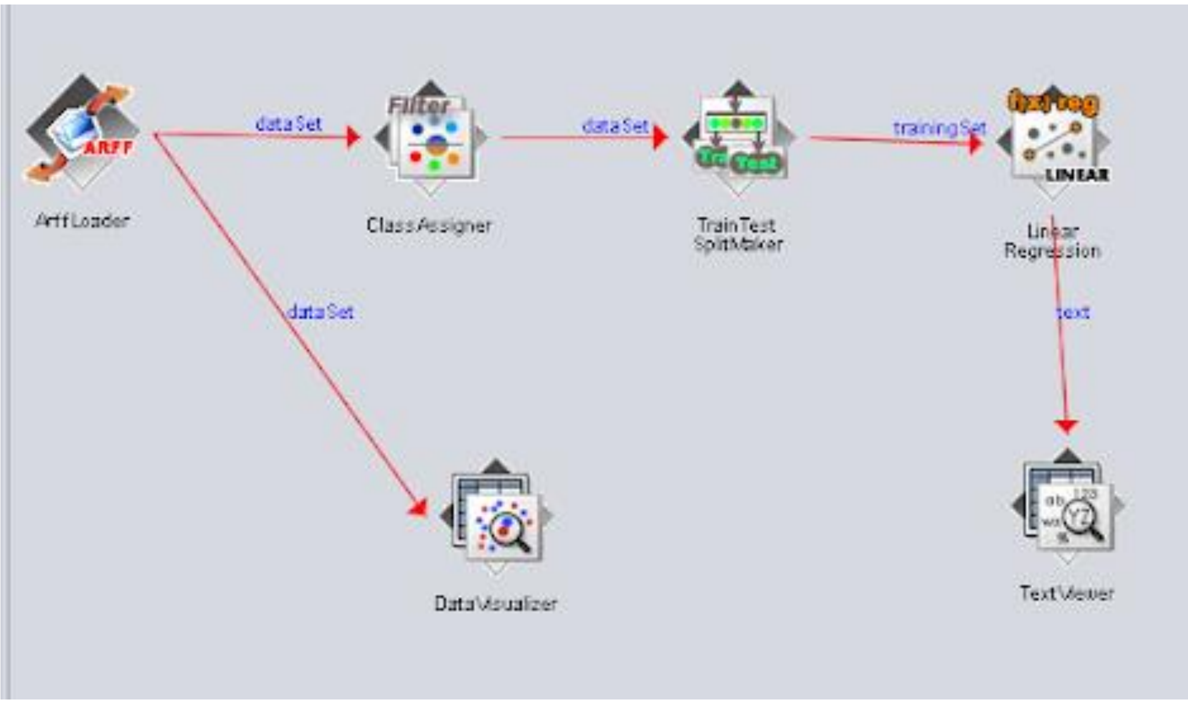
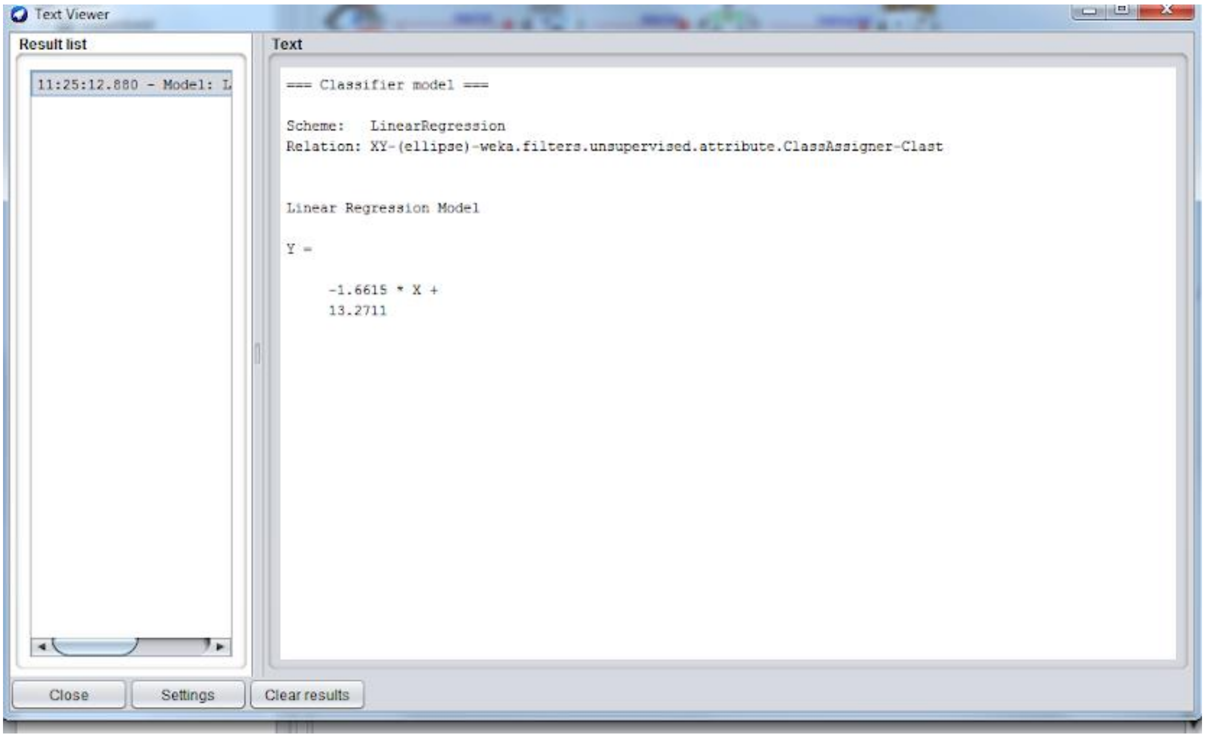
SPRAWOZDANIE – LABORATORIUM 2

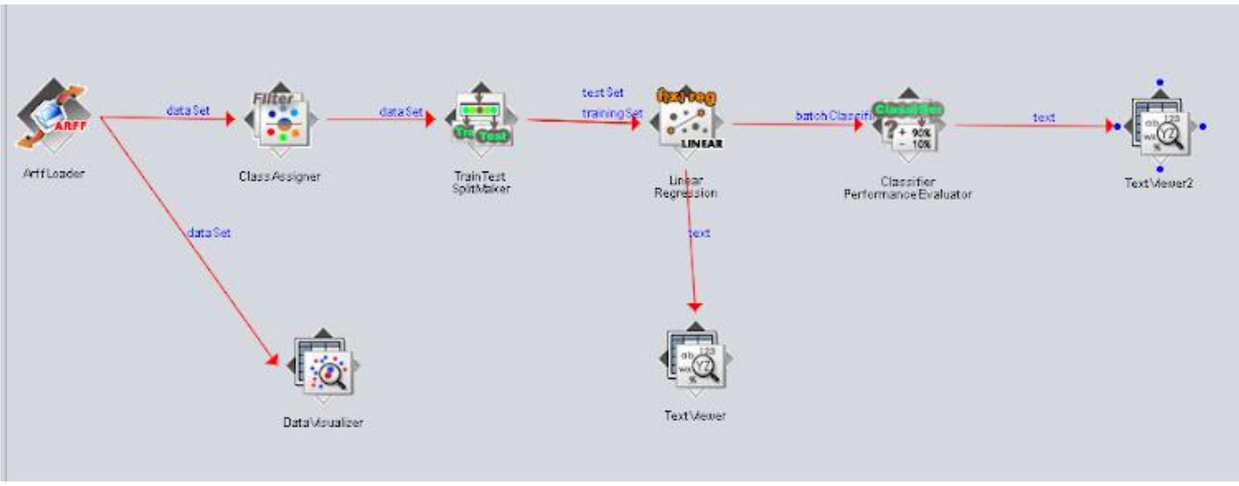
Karolina Kotłowska, 13 marca 2023

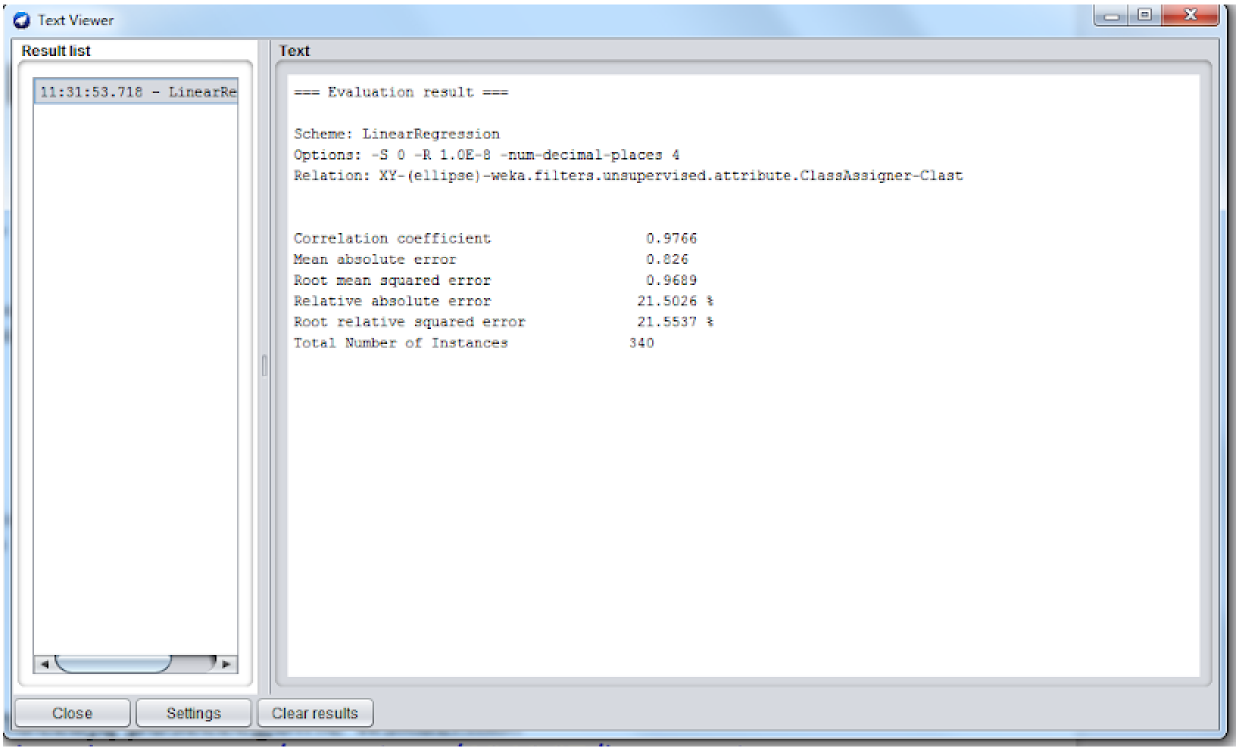
2.1





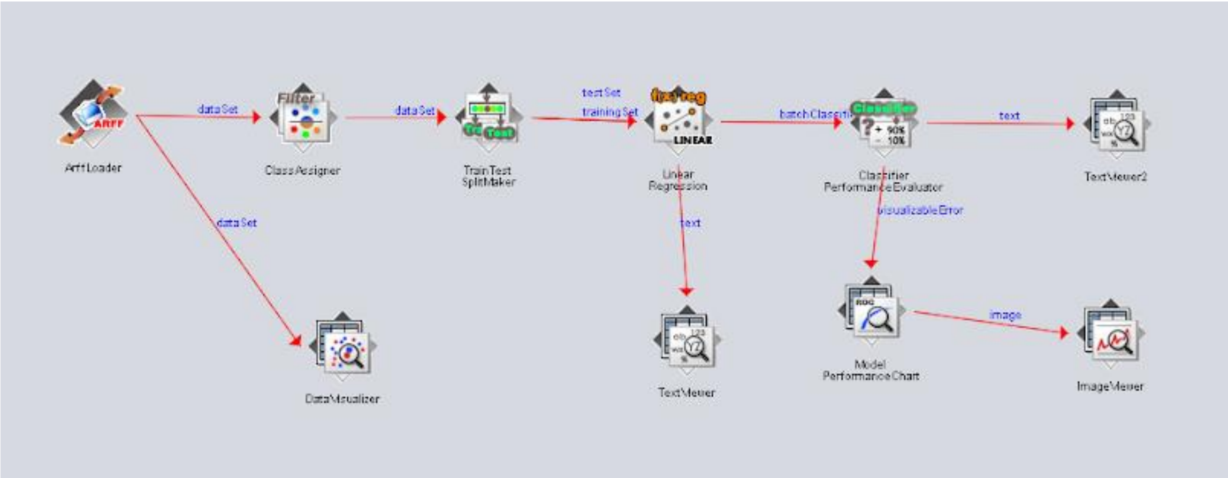
2.2

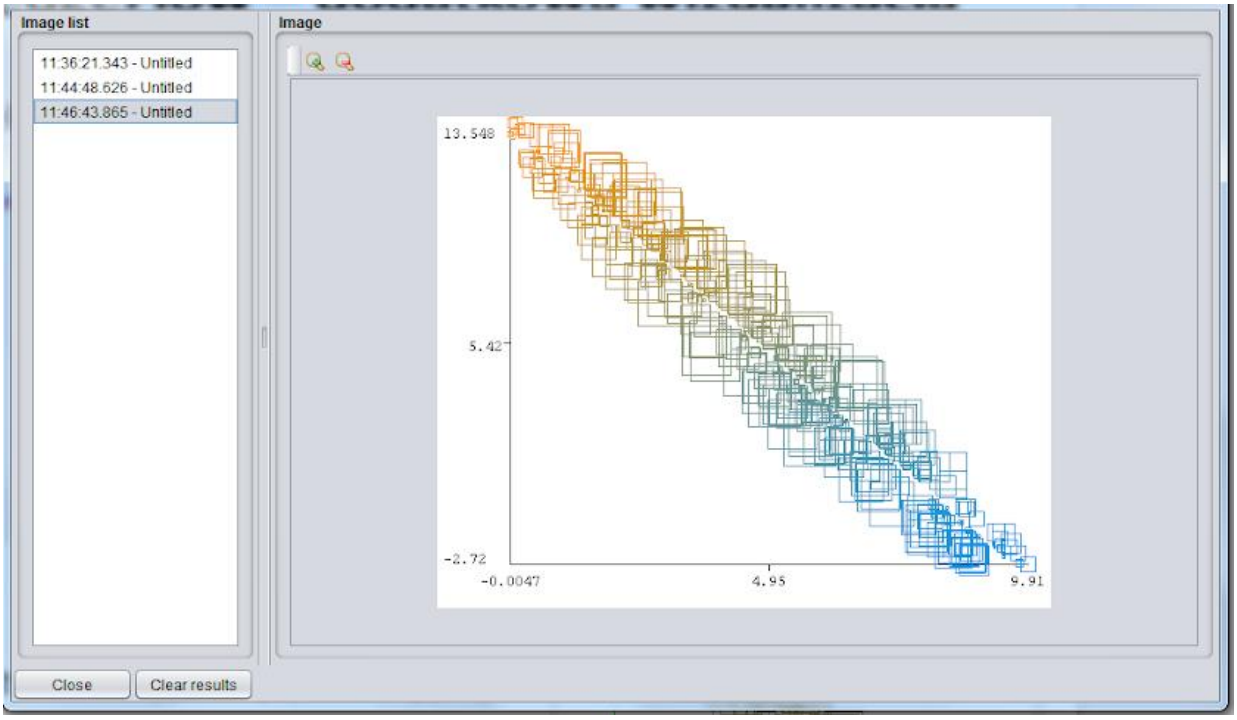




Total number of instances wynosi 340, ponieważ taki jest rozmiar zbioru testowego.

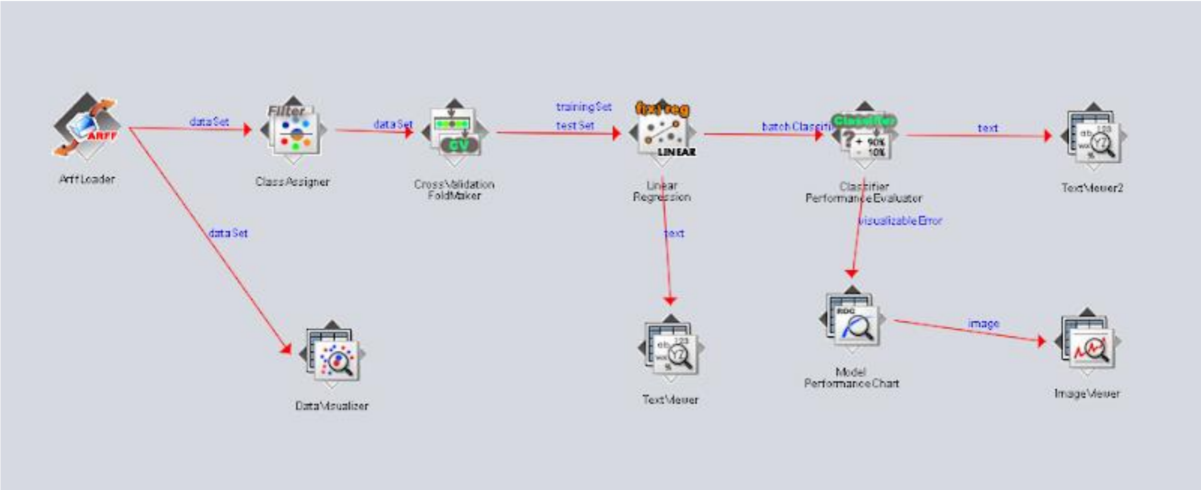
2.3

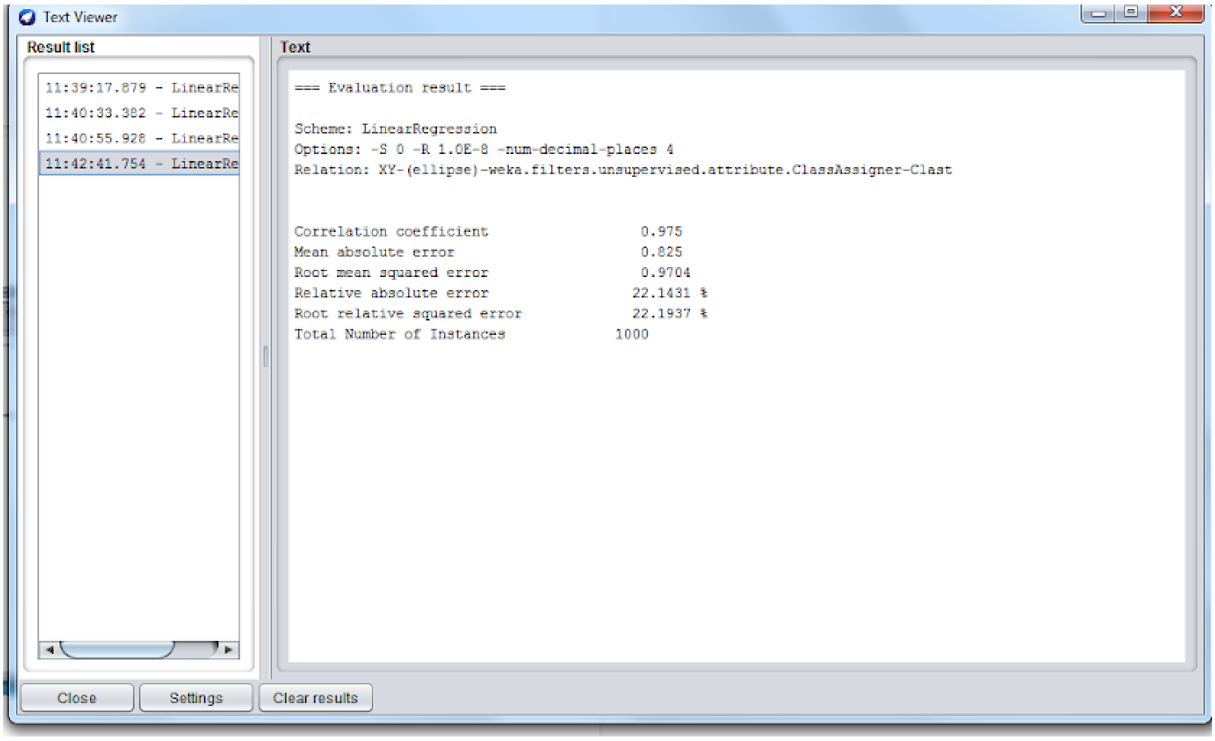




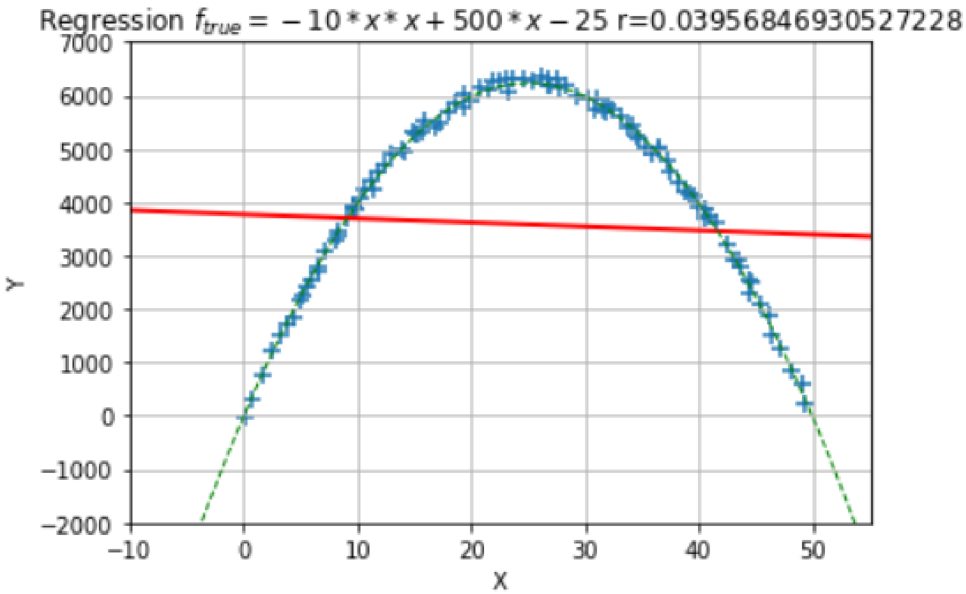
Kwadraty reprezentują błędy. Bliższa odległość do środka elipsy oznacza mniejszy błąd.

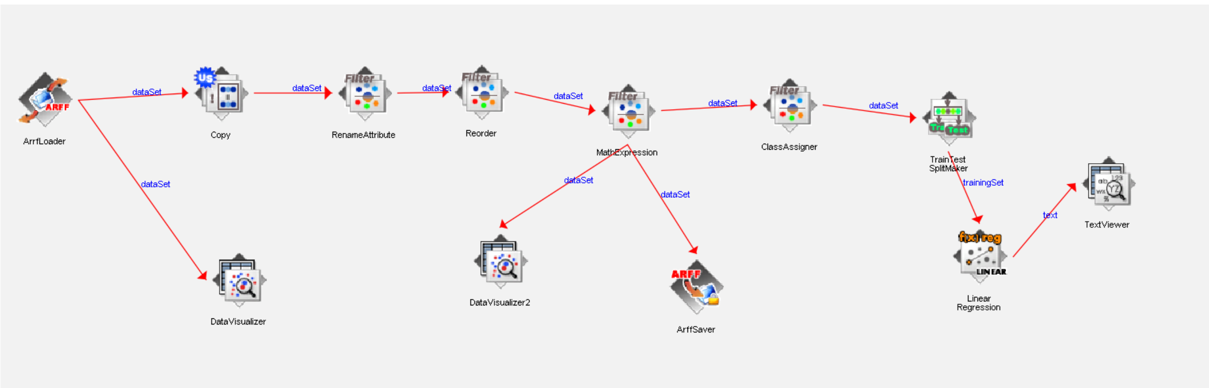
2.4


ArffLoader
dataSet
ClassAssigner
dataSet
CrossValidationFoldMaker
trainingSet
testSet
LINEAR
Linear Regression
batchClassifier
Classifier PerformanceEvaluator
text
TextViewer2
dataSet
text
visualizableError
DataVisualizer
TextViewer
Model PerformanceChart
image
ImageViewer


Text Viewer
Result list
11:39:17.879 - LinearRe
11:40:33.382 - LinearRe
11:40:55.928 - LinearRe
11:42:41.754 - LinearRe
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
Close
Settings
Clear results

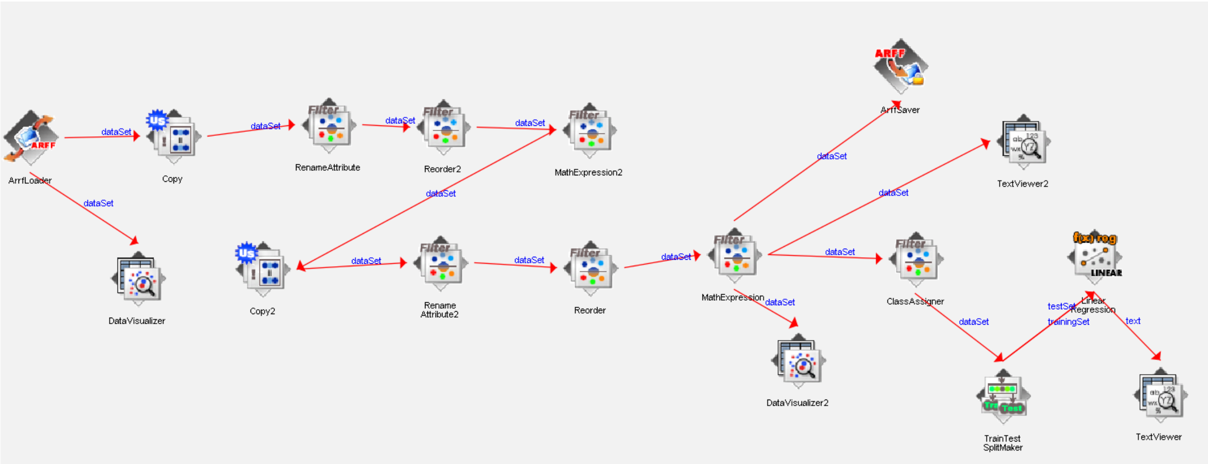
2.5



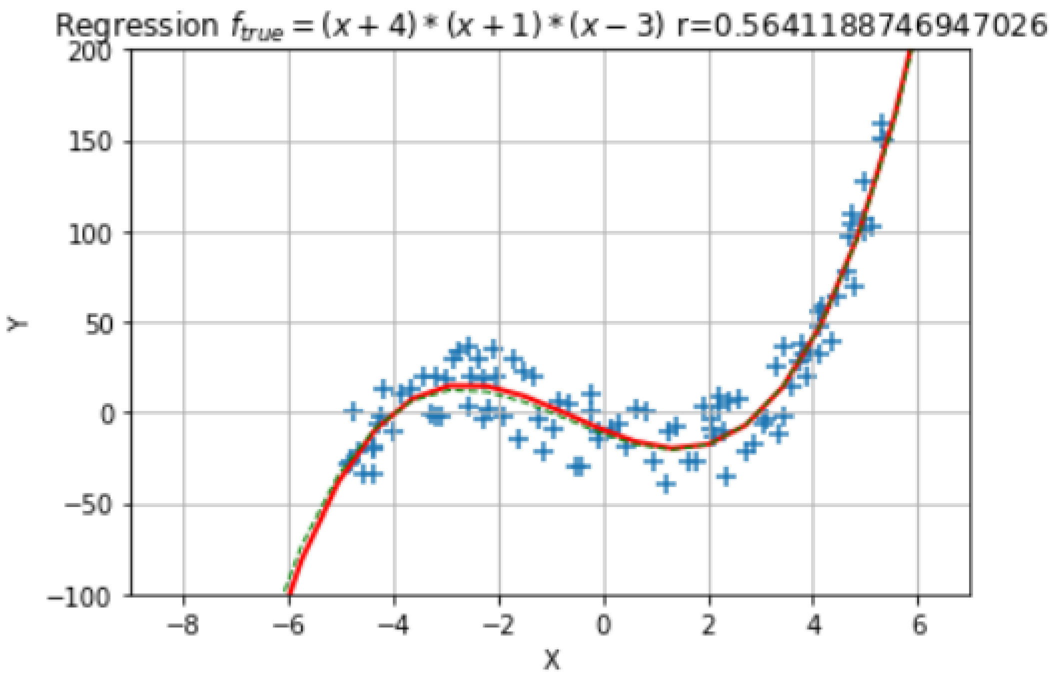



```
=== Classifier model ===

Scheme:    LinearRegression
Relation: XY-10X2+500X-25+[e=150]-weka.filters.unsupervised.attribute.Copy-R1-weka.filters.unsupervised.at


Linear Regression Model

Y =

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

2.6




```
=== Classifier model ===

Scheme:   LinearRegression
Relation: XY-(x+4)(x+1)(x-3)+[e=25]-weka.filters.unsupervised.attribute.Copy-R1-weka.filters.unsupervised.

Linear Regression Model

Y =

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



2.7

```
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats
from io import StringIO
from sklearn import linear_model
```

[12] ✓ 0.1s

```
inp = "dane/xy-004a.arff"
x,x2, y = np.loadtxt(inp, delimiter=',', usecols=(0, 1, 2), unpack=True,skiprows=7)
```

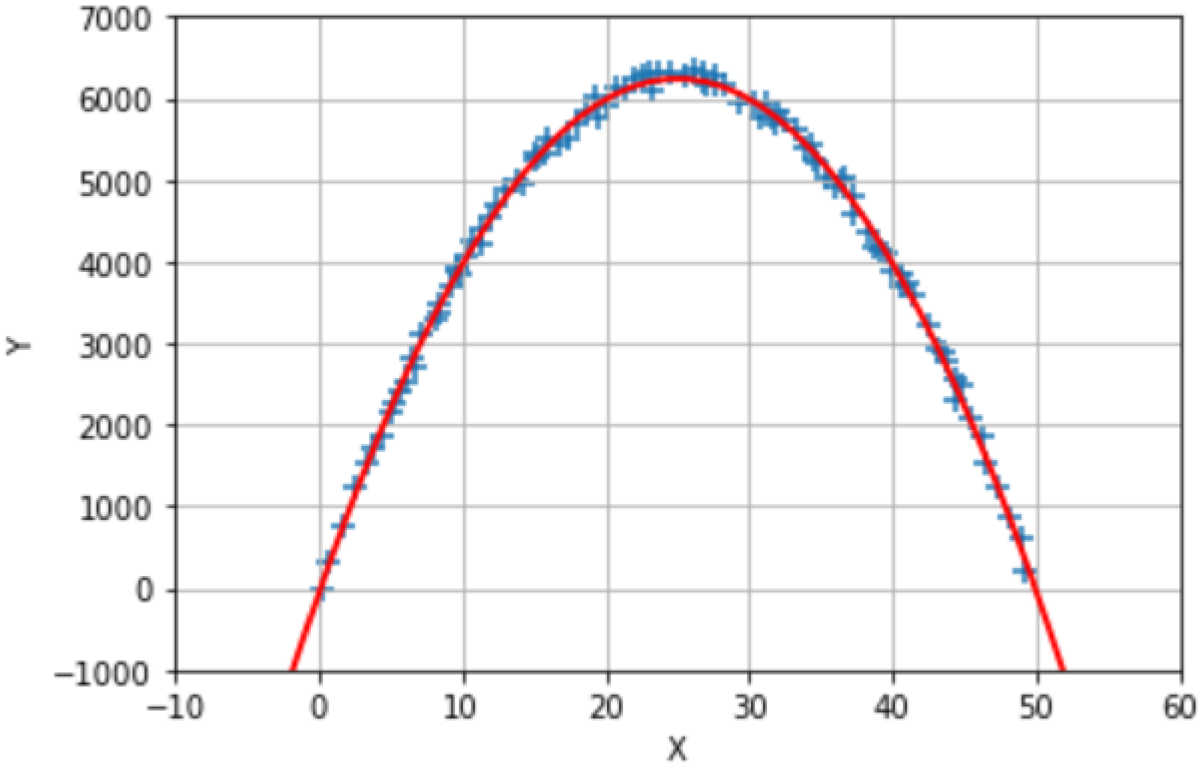
[17] ✓ 0.0s

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

[18] ✓ 0.0s

```
regr=linear_model.LinearRegression()

regr.fit(features, y)
fx=np.linspace(-100,150,200);
fy=regr.coef_[1]* fx*fx+regr.coef_[0]*fx+ regr.intercept_

plt.scatter(x,y,s=80, marker='+')
plt.plot(fx,fy,linewidth=2,color='r')
plt.xlim(-10,60)
plt.ylim(-1000,7000)
plt.grid(True)
plt.xlabel('X')
plt.ylabel('Y')
r = stats.pearsonr(x, y)[0]
plt.show()
```

[23] ✓ 0.1s

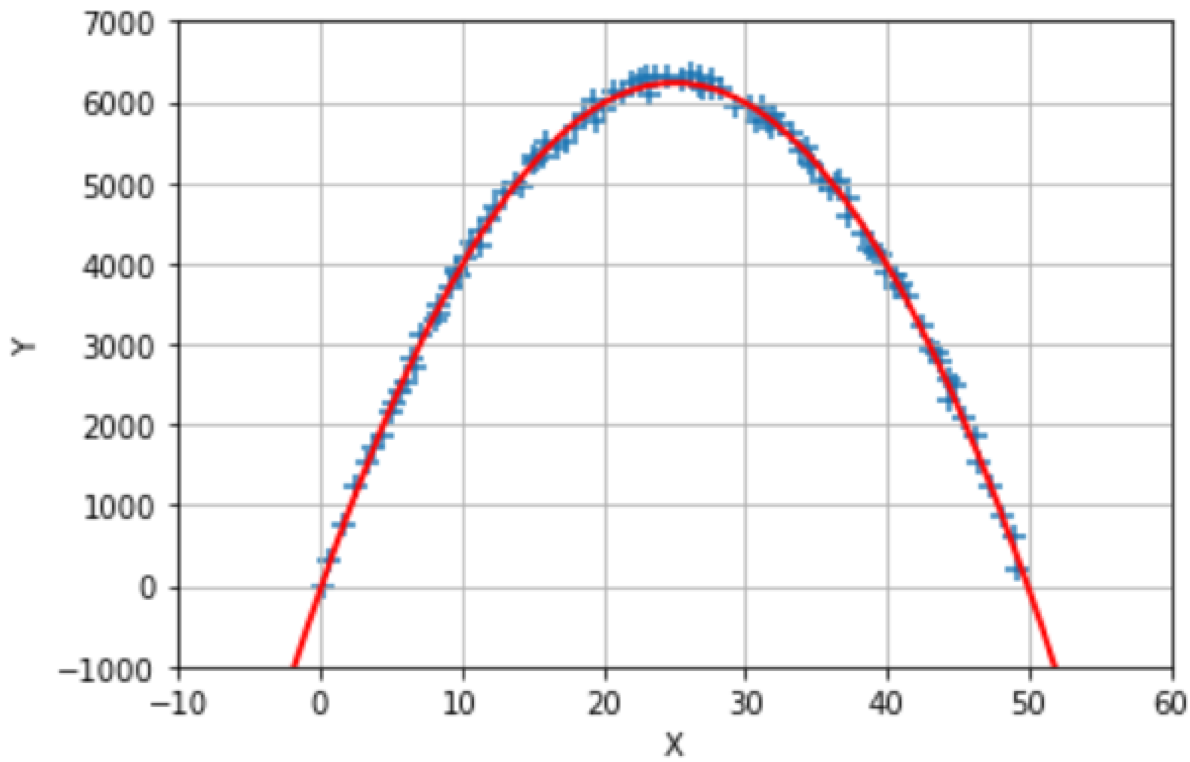


Y = -10.078577976286564*x*x 504.34622442974745*x -69.71970014858198

2.8.1

```
#enter data as a string
inp =  StringIO(data)
x, y = np.loadtxt(inp, delimiter=',', usecols=(0, 1), unpack=True,skiprows=0)
features = np.stack((x,x*x),axis=-1)
regr = linear_model.LinearRegression()
regr.fit(features, y)
fx=np.linspace(-100,150,200);
fy=regr.coef_[1]* fx*fx+regr.coef_[0]*fx+ regr.intercept_

plt.scatter(x,y,s=80, marker='+')
plt.plot(fx,fy,linewidth=2,color='r')
plt.xlim(-10,60)
plt.ylim(-1000,7000)
plt.grid(True)
plt.xlabel('X')
plt.ylabel('Y')
r = stats.pearsonr(x, y)[0]
plt.show()
```

✓ 0.3s

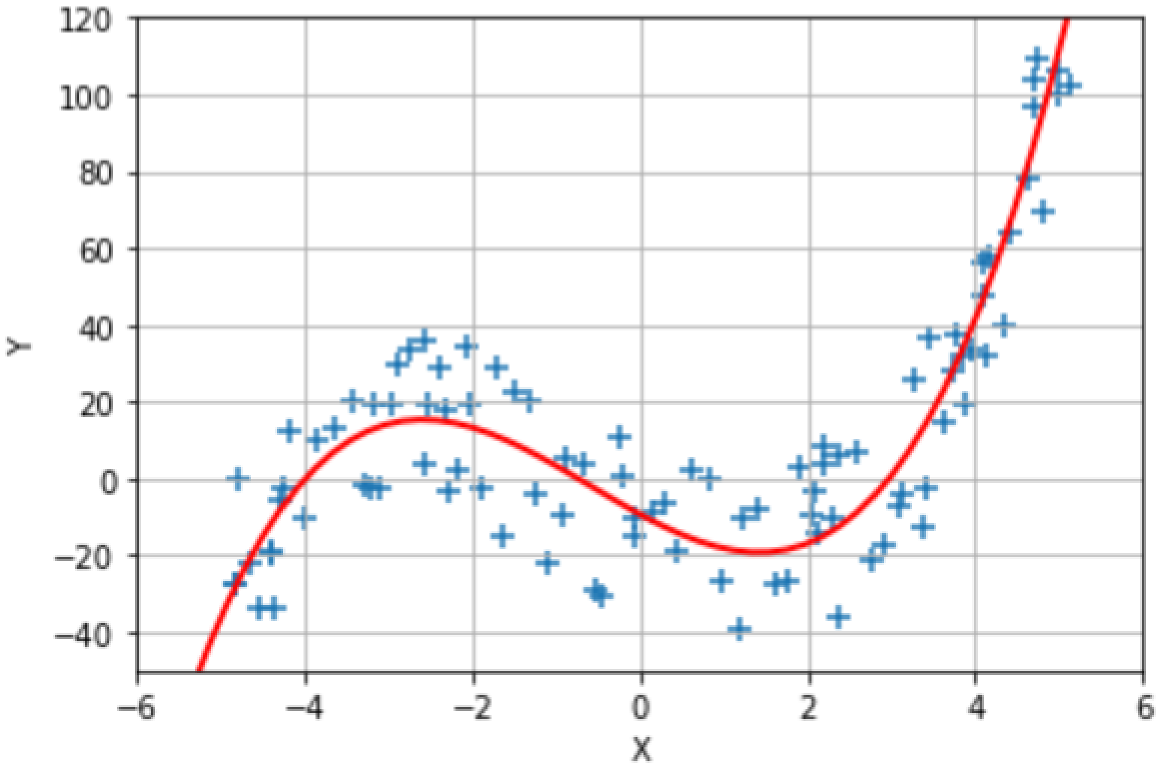



```
Y = -10.078577973912273*x*x + 504.3462243122894*x -69.719699643284
```

2.8.2

```python
#enter data as a string
inp =  StringIO(data)
x, y = np.loadtxt(inp, delimiter=',', usecols=(0, 1), unpack=True,skiprows=0)
features = np.stack((x,x**2,x**3),axis=-1)
regr = linear_model.LinearRegression()
regr.fit(features, y)
fx=np.linspace(-100,150,1200);
fy=regr.coef_[2]*fx*fx*fx + regr.coef_[1]* fx*fx+regr.coef_[0]*fx+ regr.intercept_

plt.scatter(x,y,s=80, marker='+')
plt.plot(fx,fy,linewidth=2,color='r')
plt.xlim(-6,6)
plt.ylim(-50,120)
plt.grid(True)
plt.xlabel('X')
plt.ylabel('Y')
r = stats.pearsonr(x, y)[0]
plt.show()

print(regr.coef_[2], regr.coef_[1], regr.coef_[0], regr.intercept_)
```

✓ 0.4s



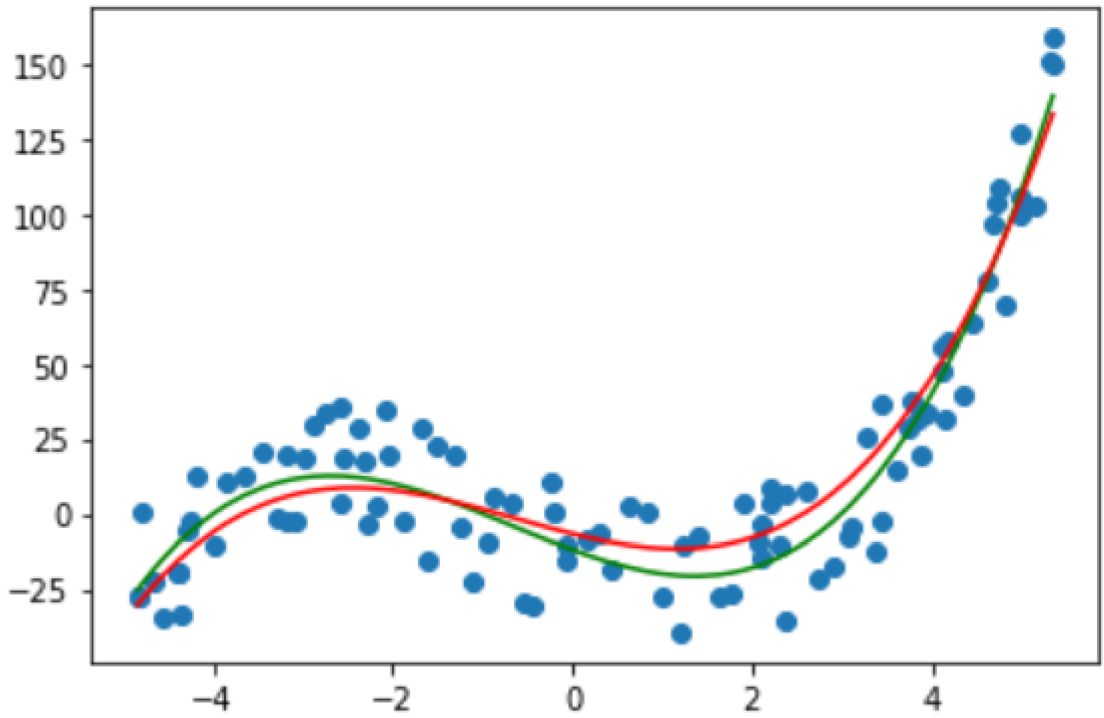
Y = 1.062124750327371*x*x*x + 1.8638922404371554*x*x + -11.81863318096572 - 9.271207608853327

2.9

```
"""
#enter data as a string
inp =  StringIO(data)
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(learning_rate=0.1), loss='mse', metrics=['mse','mae'])
hist = model.fit(X,y,epochs=50,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')
```
✓ 3.8s

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 1)                 4

=================================================================
Total params: 4
Trainable params: 4
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
4/4 [==============================] - 1s 5ms/step - loss: 3427.3066 - mse: 3427.3066 - mae: 36.6189
Epoch 2/50
4/4 [==============================] - 0s 5ms/step - loss: 1228.9156 - mse: 1228.9156 - mae: 23.9830
Epoch 3/50
4/4 [==============================] - 0s 4ms/step - loss: 719.6624 - mse: 719.6624 - mae: 20.6104
Epoch 4/50
4/4 [==============================] - 0s 5ms/step - loss: 580.2526 - mse: 580.2526 - mae: 19.6430
Epoch 5/50
4/4 [==============================] - 0s 6ms/step - loss: 551.9044 - mse: 551.9044 - mae: 18.8799
Epoch 6/50
4/4 [==============================] - 0s 6ms/step - loss: 505.1044 - mse: 505.1044 - mae: 18.0829
Epoch 7/50
4/4 [==============================] - 0s 4ms/step - loss: 524.8467 - mse: 524.8467 - mae: 18.4810
...
4/4 [==============================] - 0s 5ms/step - loss: 280.5280 - mse: 280.5280 - mae: 14.0256
Epoch 50/50
4/4 [==============================] - 0s 6ms/step - loss: 237.7508 - mse: 237.7508 - mae: 13.0921
4/4 [==============================] - 0s 4ms/step
```

Uzyskano najlepsze wyniki na liczby epok równej 50. Parametr RMSprop został uzupełniony o learning_rate = 0.1. Sieć zawiera warstwę gęstą, w której każdy neuron połączony jest z każdym neuronem z poprzedniej warstwy.



150
125
100
75
50
25
0
−25
−4
−2
0
2
4