Metody eksploracji danych
Laboratorium 1

Weka + Python + regresja

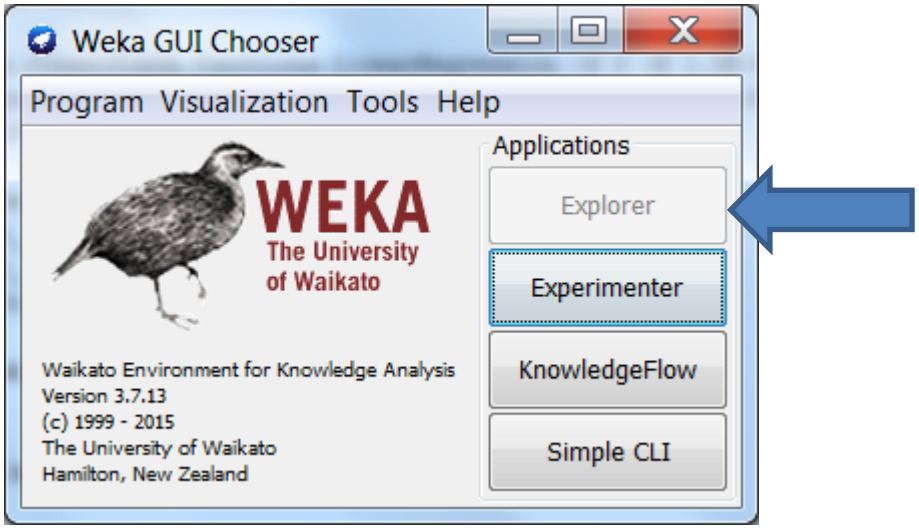
Metody eksploracji danych

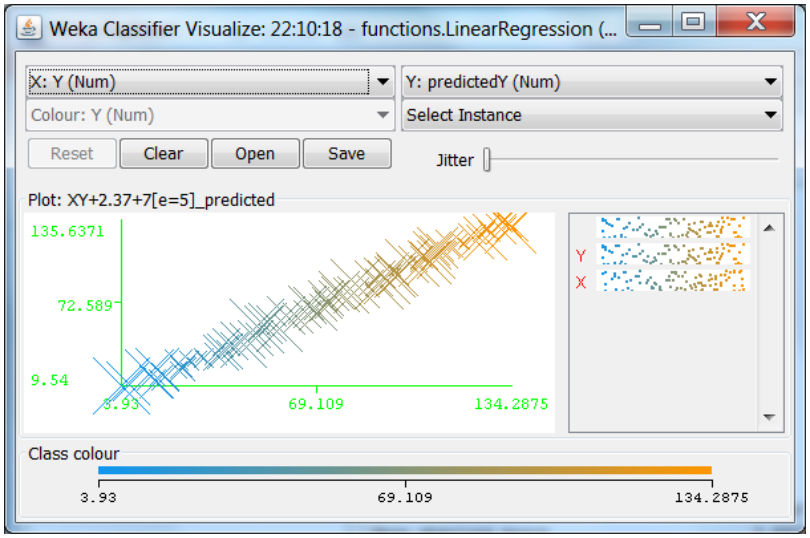
- Zasoby
  - Weka (gdzieś na dysku)
  - Środowisko dla języka Python (Spyder, Jupyter, gdzieś na dysku)
  - Zbiory danych (dostępne na stronie http://home.agh.edu.pl/~pszwed/wiki/doku.php?id=metody_eksploracji_danych)
- Cel
  - Poznanie możliwości Weki
  - Wykonanie klika przykładów regresji
  - Alternatywne środowisko i biblioteki języka Python
  - Typowy workflow podczas analizy danych

Część 1 – Nawigacja po GUI Weka + nieco bibliotek Pythona

1.1 W oknie startowym Weka uruchom opcję Explorer

- Załaduj plik **xy-001.arff** w zakładce **Preprocess**
- Możesz wyświetlić dane w tabeli **Edit**
- Wyświetl zawartość danych w zakładce **Visualize**
- W zakładce **Classify** wybierz **Choose**: functions -> LinearRegression
- Naciśnij Start. Powinno się pojawić równanie prostej
- Niestety za pomocą Weka nie można wyświetlić prostej, co najwyżej zakres błędów klasyfikacji.
- Kliknij prawym klawiszem na result list i wybierz jedną z opcji wizualizacji.





1.2. Wyświetlamy dane i krzywą

Korzystamy z bibliotek w języku Python

1. Używamy IDE Spyder lub Jupyter. Można też przygotować plik tekstowy i uruchomić z poziomu konsoli (mając nadzieję, że biblioteki są zainstalowane). Czasem też działa https://try.jupyter.org/
2. Kopiujemy fragment kodu ze strony: http://home.agh.edu.pl/~pszwed/wiki/doku.php?id=metody_eksploracji_danych
3. Wpisujemy dane w osobnych liniach. """ to specjalny zapis dla tekstu wprowadzanego w wielu wierszach.
4. x, y = np.loadtxt(inp, delimiter=',', usecols=(0, 1), unpack=**True**,skiprows=6) – ładuje tekst z kolumn 0 i 1 do tablic x,y
5. plt.scatter(x,y,s=80, marker='+') – rysuje punkty
6. fx=np.linspace(-10,60,100) 100 równomiernie rozłożonych punktów w zakresie [-10,60]
7. fy=2.3702*fx+6.1973 - wartości funkcji regresji (to są operacje na wektorach)
8. ftrue=2.37*fx+7 – „prawdziwa” funkcja użyta do generacji danych
9. plt.plot(fx,fy,linewidth=2,color='r') – wykres liniowy
10. plt.xlim(-10,60) – zakres wyświetlanych zmiennych

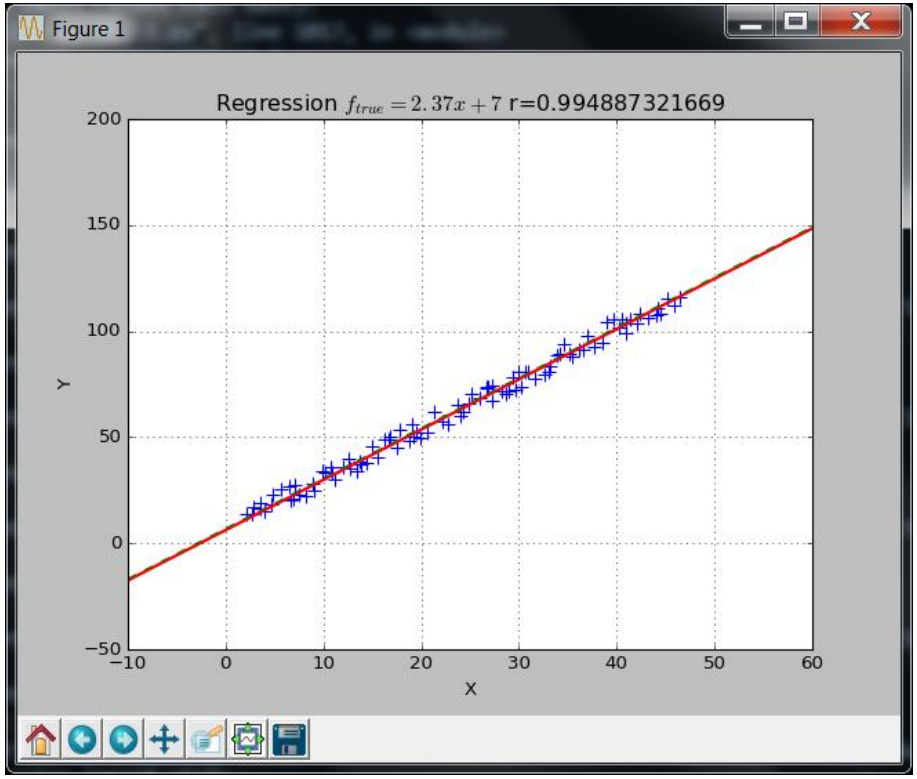
Oczekiwany wynik

- Wykres może być osadzony na stronie HTML w Jupyter: usuń komentarze z pierwszej linii: *%matplotlib notebook*
- Wykres można zapisać w jednym z formatów (bitmapowych/wektorowych)

Wartość r która jest wyświetlana dodatkowo?
Jest to współczynnik korelacji.

Przeczytaj tekst i obejrzyj przykłady:
https://en.wikipedia.org/wiki/Correlation_and_dependence

Jeśli r ≈ 0, zazwyczaj równanie regresji będzie miało postać:
$y = 0x + c$



Inne pliki

Analogicznie załaduj pliki,wyznacz równanie regresji, oceń błędy i wyświetl przebieg krzywej regresji dla:

1.3 Pliku xy-002.arff

1.4 Pliku xy-003.arff

1.5 Pliku xy-005.arff

1.6 Pliku xy-006.arff

1.7 Pliku xy-007.arff

1.8 Pliku xy-008.arff [Skomentuj przebieg krzywej regresji]

1.8 Pliku xy-009.arff

1.9 Pliku xy-010.arff

1.10 Realizacja za pomocą bibliotek Python

Przeprowadź regresję dla pliku xy-002.arff

Najprostsza forma ładowania pliku arff

inp = "xy-002.arff„

x, y = np.loadtxt(inp, delimiter=',', usecols=(0, 1), unpack=True,skiprows=6)

Liczba wierszy do przeskoczenia zależy od pliku

Tworzenie macierzy cech

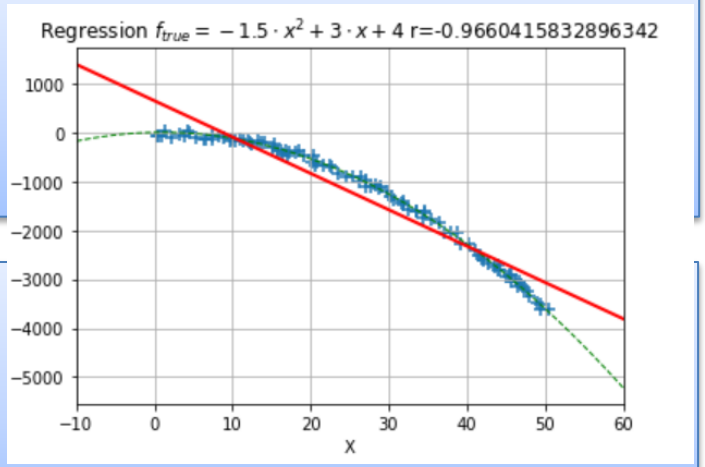
features=x.reshape(x.size,1)

Musi być to macierz dwuwymiarowa $m \times 1$ (w ogólnym przypadku $m \times n$)

Dopasowanie krzywej

regr = linear_model.LinearRegression()

regr.fit(features, y)



Parametry modelu (regr.coef_ jest tablicą)

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

…

fy= regr.coef_[0]* fx + regr.intercept_

Metody regresji

Metoda najmniejszych kwadratów

```
regr = linear_model.LinearRegression()
regr.fit(features, y)
```

Regresja grzbietowa: metoda gradientu prostego z regularyzacją współczynników, norma L^2

```
regr = linear_model.Ridge (alpha = .5)
regr.fit(features, y)
```

Lasso: metoda gradientu prostego z regularyzacją współczynników, norma L^1

```
regr  = linear_model.Lasso(alpha=0.1)
regr.fit(features, y)
```

1.11 Uruchom poniższy kod dla danych w xy-002.arff

```
import statsmodels.api as sm
X_plus_one = np.stack( (np.ones(x.size),x), axis=-1)
X_plus_one
ols = sm.OLS(y, X_plus_one)
ols_result = ols.fit()
ols_result.summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.994
Model:	OLS	**Adj. R-squared:**	0.994
Method:	Least Squares	**F-statistic:**	1.576e+04
Date:	Wed, 02 Mar 2022	**Prob (F-statistic):**	4.64e-110
Time:	23:59:19	**Log-Likelihood:**	-249.30
No. Observations:	100	**AIC:**	502.6
Df Residuals:	98	**BIC:**	507.8
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	7.8060	0.613	12.738	0.000	6.590	9.022
x1	2.3409	0.019	125.533	0.000	2.304	2.378

Omnibus:	69.651	**Durbin-Watson:**	1.747
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	7.581
Skew:	-0.059	**Prob(JB):**	0.0226
Kurtosis:	1.656	**Cond. No.**	68.2

Zinterpretuj te informacje.
Uwaga nie dotyczą xy-002.arff

- Jaki jest błąd standardowy oszacowania współczynników?
- W jakim zakresie mieszczą się ich wartości z 95% wiarygodnością?

1.12

1. Powtórz poprzednią operację dla danych w xy-004.arff
2. Jakie są standardowe błędy oszacowania współczynników?
3. W jakim zakresie mieszczą się współczynniki przy x1 oraz przesunięcie (intercept) – z wiarygodnością 95%
4. Narysuj możliwe skrajne przebiegi