



Uczenie liniowej hipotezy dla problemu przewidywania cen nieruchomości

Mateusz ZAJĄC

18.06.2021

Streszczenie

W pracy opisano rozwiązanie problemu przewidywania cen nieruchomości na podstawie danych testowych pochodzących z rynku nieruchomości w Bostonie.

Spis treści

1	Wprowadzenie	1
1.1	Opis problemu	1
2	Opis metody	2
2.1	Wprowadzenie teoretyczne	2
2.2	Opis danych wejściowych	2
2.3	Badania symulacyjne	3
3	Podsumowanie	7
A	Kod programu	8

Rozdział 1

Wprowadzenie

1.1 Opis problemu

Wycena nieruchomości to analiza wielu czynników takich jak stan prawny nieruchomości, stan techniczny, otoczenie, lokalizacja oraz przeznaczenie nieruchomości. Przy tak dużej liczbie danych bardzo pomocna jest sztuczna inteligencja. Może przyspieszyć, ułatwić i udoskonalić cały proces. Tematem tego projektu jest przygotowanie modelu ułatwiającego ten proces. Projekt został przygotowany w języku Python wraz z biblioteką scikit-learn. Na potrzeby trenowania modelu użyłem zestawu danych zawartych w tej bibliotece - Boston Housing. Dane te były pierwotnie częścią repozytorium uczenia maszynowego UCI. W zbiorze znajduje się 506 próbek, z której każda jest opisana trzynastoma cechami.

Rozdział 2

Opis metody

2.1 Wprowadzenie teoretyczne

Regresja liniowa służy do oszacowania wartości Y gdy dysponujemy wartościami X , wtedy Y nazywa się zmienną objaśnianą, a X zmienną objaśniającą. O ile współczynnik korelacji liniowej mówi nam jak bardzo dane są od siebie zależne o tyle regresja liniowa mówi nam jak bardzo zmieni się Y gdy zmienimy X . Za pomocą narzędzi zawartych w bibliotece scikit możemy przygotować taki model w bardzo prosty sposób. Bazujemy zawsze na jakimś zestawie danych, który dzielimy na zestawy treningowe oraz zestawy testujące. Następnie używając wbudowanej funkcji `fit` trenujemy model i za pomocą funkcji `predict` możemy przewidywać wartości wyjściowe dla nowych danych wejściowych.

2.2 Opis danych wejściowych

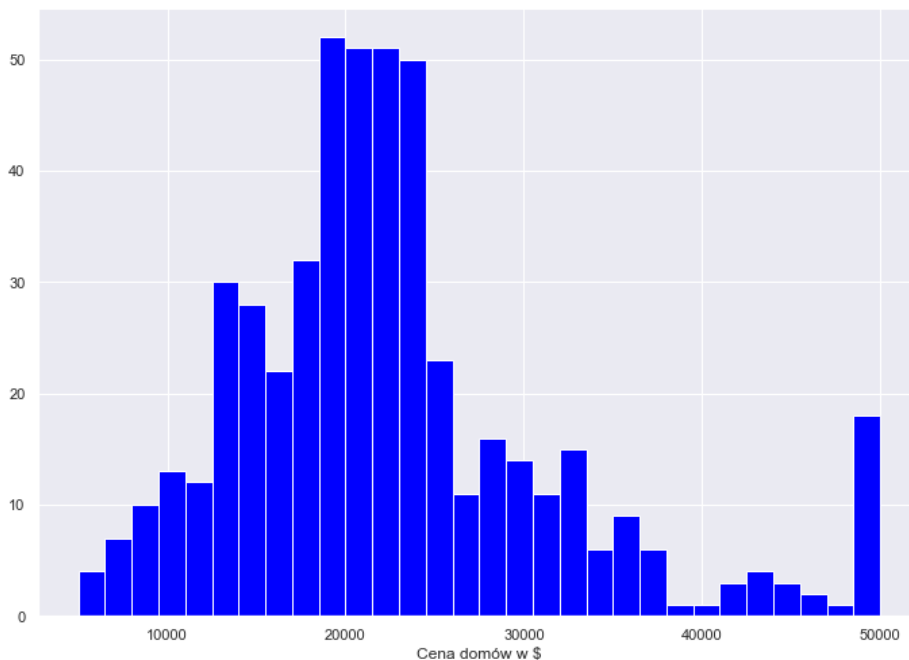
Danymi wejściowymi są cechy zawarte w zestawie "Boston house prices dataset"

Informacje o cechach:

CRIM	per capita crime rate by town
ZN	proportion of residential land zoned for lots over 25,000 sq.ft.
INDUS	proportion of non-retail business acres per town
CHAS	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
NOX	nitric oxides concentration (parts per 10 million)
RM	average number of rooms per dwelling
AGE	proportion of owner-occupied units built prior to 1940
DIS	weighted distances to five Boston employment centres
RAD	index of accessibility to radial highways
TAX	full-value property-tax rate per \$10,000
PTRATIO	pupil-teacher ratio by town
B	$1000(B_k - 0.63)^2$ where B_k is the proportion of blacks by town
LSTAT	% lower status of the population
MEDV	Median value of owner-occupied homes in \$1000's

2.3 Badania symulacyjne

Przed rozpoczęciem uczenia modelu bardzo ważną jest analiza danych. Dlatego przedstawiam na wykresie rozkład cen wartości nieruchomości z pomocą wykresu histogramu z biblioteki matplotlib.



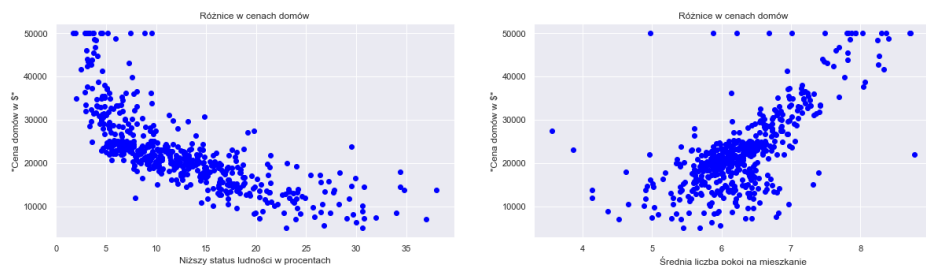
Jak widać na powyższym wykresie dane rozkładają się normalnie z kilkoma wartościami odstającymi. Natępnie przedstawie macierz korelacji, która

reprezentuje liniowe korelacje między zmiennymi. Do tego celu użyje funkcji `corr()` z biblioteki `pandas`.



Współczynnik korelacji jest liczbą z przedziału $-1, 1$. Gdy współczynnik jest bliski 1, oznacza to że istnieje mocna korelacja dodatnia między dwoma zmiennymi, a gdy współczynnik jest bliski -1 istnieje mocna korelacja ujemna. Analizując wykres można zauważyć, że RM (średnia liczba pokoi na mieszkanie) ma silną dodatnią korelację z ceną nieruchomości, natomiast LSTAT (niższy status ludności w procentach) ma silną ujemną korelację z ceną.

Z macierzy korelacji wynika, że cech RAD oraz TAX mają korelację 0.91, cechy DIS i AGE mają korelację -0.75 oraz DIS i NOX mają korelację -0.77. Te pary cech są ze sobą silnie skorelowane i mogą one wpłynąć na model.



Ceny nieruchomości rosną prawie liniowo wraz z liniowym wzrostem liczby pokoi na mieszkanie. Jest kilka wartości oddających. Natomiast wraz

ze wzrostem liczby ludności o niższym statusie spadają ceny nieruchomości. Ten spadek na wykresie nie wydaje się już liniowy.

Na początek wytrenujemy model za pomocą cechy RM. Aby wytrenować model dzielimy nasz zestaw danych na zestaw do nauki oraz do testów. Zestaw treningowy będzie się składał z 80% danych a zestaw testujący z 20%.

Przy wykożystaniu tylko tej zmiennej do trenowania modelu uzyskano następujące wyniki:

Dla zestawu treningowego:

Błąd średniokwadratowy 6.972277149440585

Współczynnik determinacji 0.43

Dla zestawu testowego:

Błąd średniokwadratowy 4.895963186952216

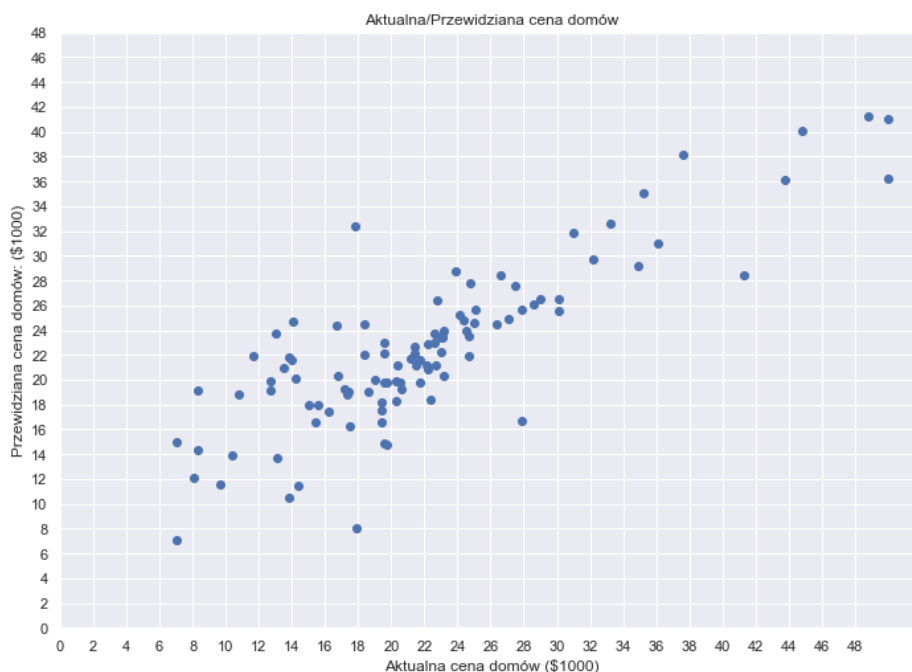
Współczynnik determinacji 0.69

Wynik przewidywania ceny dla elementu numer 19 w zbiorze danych:

Przewidziana cena dla elementu "19" 17.69280457

Realna cena dla elementu "19" 18.2

Poniższy wykres przedstawia stosunek ceny przewidzianej do realnej ceny nieruchomości.



Następnym krokiem jest wytrenowanie modelu dla wszystkich cech.

Dla zestawu treningowego:

Błąd średniokwadratowy 4.6520331848801675

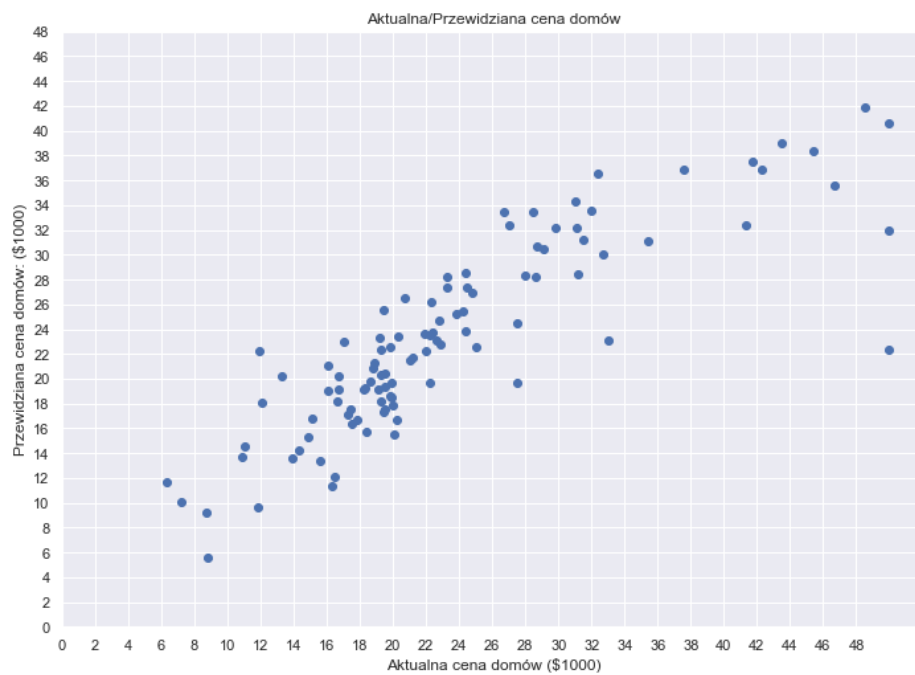
Współczynnik determinacji 0.75

Dla zestawu testowego:

Błąd średniokwadratowy 4.928602182665355

Współczynnik determinacji 0.67

Poniższy wykres przedstawia stosunek ceny przewidzianej do realnej ceny nieruchomości.



Rozdział 3

Podsumowanie

Za pomocą biblioteki scikit możemy w prosty sposób zaimplementować regresję liniową i przewidywać wartości nieruchomości w oparciu odpowiednie dane wejściowe.

Dodatek A

Kod programu

```
import numpy as np
import matplotlib.pyplot as plt

import pandas as pd
import seaborn as sb

%matplotlib inline

from sklearn.datasets import load_boston
boston_market_data = load_boston()
print(boston_market_data['DESCR'])

boston = pd.DataFrame(boston_market_data.data,
                      columns=boston_market_data.feature_names)
boston.head()

boston['MEDV'] = boston_market_data.target
boston.head()

sb.set(rc={'figure.figsize':(11.7,8.27)})
plt.hist(boston['MEDV'] * 1000, color="blue", bins=30)
plt.xlabel("Cena domów w $")
plt.show()

correlation_matrix = boston.corr().round(2)
sb.heatmap(data=correlation_matrix, annot=True)

plt.figure(figsize=(20, 5))
target = boston['MEDV'] * 1000
```

```
plt.subplot(1, 2 , 1)
x = boston['LSTAT']
y = target
plt.scatter(x, y,color='blue', marker='o')
plt.title("Różnice w cenach domów")
plt.xlabel('Niższy status ludności w procentach')
plt.ylabel('"Cena domów w $"')

plt.subplot(1, 2 , 2)
x = boston['RM']
y = target
plt.scatter(x, y,color='blue', marker='o')
plt.title("Różnice w cenach domów")
plt.xlabel('Średnia liczba pokoi na mieszkanie')
plt.ylabel('"Cena domów w $"')

X_rooms = boston.RM
y_price = boston.MEDV

X_rooms = np.array(X_rooms).reshape(-1,1)
y_price = np.array(y_price).reshape(-1,1)

X_train_1, X_test_1, Y_train_1, Y_test_1=
train_test_split(X_rooms, y_price, test_size=0.2, random_state=5)

reg_1=LinearRegression()
reg_1.fit(X_train_1, Y_train_1)

y_train_predict_1=reg_1.predict(X_train_1)
rmse= (np.sqrt(mean_squared_error(Y_train_1, y_train_predict_1)))
r2=round(reg_1.score(X_train_1, Y_train_1),2)

print('Błąd średniokwadratowy {}'.format(rmse))
print('Współczynnik determinacji {}'.format(r2))
print("\n")

y_pred_1 = reg_1.predict(X_test_1)
rmse = (np.sqrt(mean_squared_error(Y_test_1, y_pred_1)))
r2 = round(reg_1.score(X_test_1, Y_test_1),2)

print("Błąd średniokwadratowy: {}".format(rmse))
print("Współczynnik determinacji: {}".format(r2))
```

```
id = 19
y_pred_id_5 = reg_1.predict(np.array(X_rooms[id]).reshape(-1,1))

print("Przewidziana cena dla elementu \"{0}\" to {1}".format(id,
y_pred_id_5[0]))
print("Realna cena dla elementu \"{0}\" to {1}".format(id, y_price[id]))

plt.scatter(Y_test_1, y_pred_1)
plt.xlabel("Aktualna cena domów ($1000)")
plt.ylabel("Przewidziana cena domów: ($1000)")
plt.xticks(range(0, int(max(Y_test_1)),2))
plt.yticks(range(0, int(max(Y_test_1)),2))
plt.title("Aktualna/Przewidziana cena domów")

X = boston.drop('MEDV', axis = 1)
y = boston['MEDV']

X_train, X_test, y_train, y_test = train_test_split(X,y,
test_size=0.2, random_state=4)

reg_all = LinearRegression()
reg_all.fit(X_train, y_train)

# model evaluation for training set

y_train_predict = reg_all.predict(X_train)
rmse = (np.sqrt(mean_squared_error(y_train, y_train_predict)))
r2 = round(reg_all.score(X_train, y_train),2)

print('Błąd średniokwadratowy: {}'.format(rmse))
print('Współczynnik determinacji: {}'.format(r2))

y_pred = reg_all.predict(X_test)
rmse = (np.sqrt(mean_squared_error(y_test, y_pred)))
r2 = round(reg_all.score(X_test, y_test),2)

print("Błąd średniokwadratowy: {}".format(rmse))
print("Współczynnik determinacji: {}".format(r2))

plt.scatter(y_test, y_pred)
plt.xlabel("Aktualna cena domów ($1000)")
plt.ylabel("Przewidziana cena domów: ($1000)")
plt.xticks(range(0, int(max(y_test)),2))
```

```
plt.yticks(range(0, int(max(y_test)),2))  
plt.title("Aktualna/Przewidziana cena domów")
```