Podstawy modelu regresji liniowej

Wstęp

Model regresji liniowej jest jednym z najprostszych algorytmów uczenia maszynowego. Służy do przewidywania wartości ciągłej zmiennej zależnej na podstawie jednej lub większej liczby zmiennych niezależnych. Cel modelu regresji liniowej polega na znalezieniu najlepszej linii (w przypadku jednej zmiennej niezależnej) lub płaszczyzny hiperpłaszczyzny (w przypadku wielu zmiennych niezależnych), która pasuje do rozkładu danych.

Teoria

W prostym modelu regresji liniowej z jedną zmienną niezależną, równanie regresji liniowej ma postać:

$$y = ax + b$$

gdzie:
- y jest zmienną zależną (wartość, którą próbujemy przewidzieć),
- x jest zmienną niezależną,
- a jest współczynnikiem kierunkowym linii (pokazuje, jak bardzo y zmienia się z x),
- b jest wyrazem wolnym (pokazuje wartość y, gdy x = 0).

Celem treningu modelu regresji liniowej jest znalezienie takich wartości 'a' i 'b', które minimalizują błąd między przewidywanymi a rzeczywistymi wartościami y.

Analiza i implementacja kodu

Analiza i implementacja kodu przedstawia szczegółowe wykorzystanie biblioteki scikit-learn do budowy modelu regresji liniowej. Poniżej przedstawiono kroki realizacji wraz z kodem i omówieniem kluczowych funkcji.

Import bibliotek

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

Importujemy biblioteki `numpy` dla operacji matematycznych, `matplotlib.pyplot` do tworzenia wykresów oraz `pandas` do manipulacji danymi i ich wczytywania.

Wczytanie danych

```
zestaw_danych = pd.read_csv('mieszkania.csv')
X = zestaw_danych.iloc[:,0:1].values
y = zestaw_danych.iloc[:,-1].values
```

Dane wczytujemy za pomocą `pandas.read_csv`, a następnie selekcjonujemy kolumny: `X` dla cech (tutaj powierzchnia mieszkań) i `y` dla wartości docelowych (ceny mieszkań).

Podział danych na zestaw treningowy i testowy

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_treningowe,y_treningowe)
```

Używamy funkcji `train_test_split` z `sklearn.model_selection` do podziału danych na części treningową i testową, z proporcją 80% treningu i 20% testu. Parametr `random_state` zapewnia reprodukowalność podziału.

Trening modelu

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

Tworzymy instancję `LinearRegression` z `sklearn.linear_model` i używamy metody `.fit()` do trenowania modelu na danych treningowych.

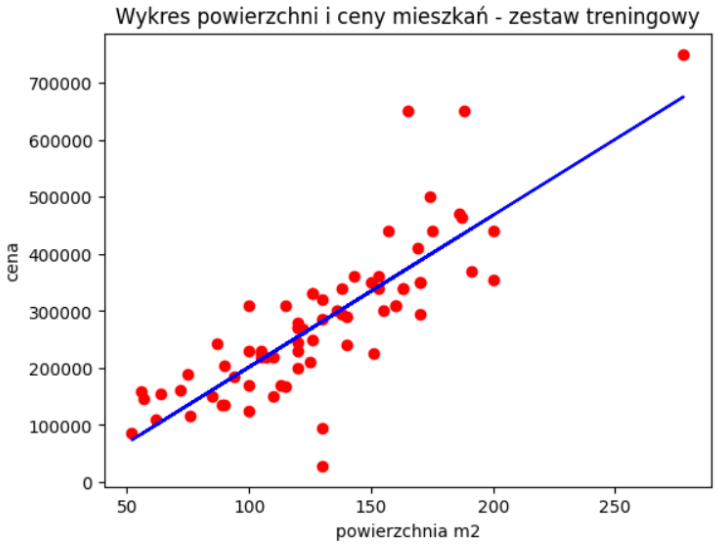
Przewidywanie wartości

```
y_przewidywania = model.predict(X_testowe)
```

Metoda `.predict()` pozwala modelowi przewidzieć wartości `y` dla danych testowych `X_testowe`.

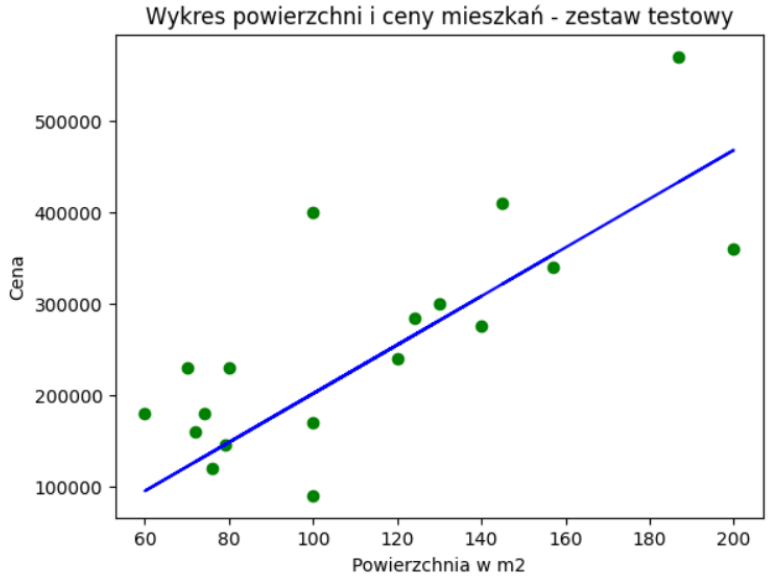
Wizualizacja wyników - dane treningowe

```
plt.scatter(X_treningowe,y_treningowe, color = 'red')
plt.title('Wykres powierzchni i ceny mieszkań - zestaw treningowy')
plt.xlabel('powierzchnia m2')
plt.ylabel('cena')
plt.plot(X_treningowe,model.predict(X_treningowe), color = 'blue')
plt.show()
```



Wizualizacja wyników - dane testowe

```
plt.scatter(X_testowe, y_testowe, color = 'green')
plt.title('Wykres powierzchni i ceny mieszkań - zestaw testowy')
plt.xlabel('Powierzchnia w m2')
plt.ylabel('Cena')
plt.plot(X_testowe, model.predict(X_testowe), color = 'blue')
plt.show()
```



Wizualizacja danych treningowych i testowych wraz z przewidywaniami modelu, co

pozwala na ocenę dopasowania modelu.

Obliczenie wartości dla nowych danych

```
powierzchnia = [[100]]
przewidziana_cena = model.predict(powierzchnia)
```

```
print(f"Dla mieszkania o powierzchni {powierzchnia[0][0]} przewidziana cena to {przewidziana_cena[0]:.0f}")
```

Demonstracja użycia modelu do przewidzenia ceny dla mieszkania o określonej powierzchni.

Podsumowanie

Model regresji liniowej, mimo swojej prostoty, stanowi potężne narzędzie w uczeniu maszynowym. Umożliwia przewidywanie wartości ciągłych jest to model od którego najłatwiej zacząć przygodę z ML.