Podstawy modelu wielokrotnej 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

Wielokrotna regresja liniowa jest rozwinięciem prostego modelu regresji liniowej, który pozwala na analizę związku między jedną zmienną zależną a dwoma lub więcej zmiennymi niezależnymi. To narzędzie jest niezwykle użyteczne w różnych dziedzinach - od nauk społecznych, przez ekonomię, aż po inżynierię - ponieważ pozwala na zrozumienie jak wiele różnych czynników wpływa na interesujące nas zjawisko.

Zacznijmy od wyjaśnienia kilku podstawowych pojęć:

- Zmienna zależna *(y):* jest to parametr, którego zmianę chcemy badać lub przewidzieć. Może to być na przykład cena mieszkania.
- Zmienne niezależne *(X1, X2, ...):* są to czynniki, które mogą mieć wpływ na zmienną zależną. W kontekście ceny mieszkań mogą to być na przykład: wielkość mieszkania, liczba pokoi, odległość od centrum miasta itd.

Model wielokrotnej regresji liniowej jest reprezentowany przez równanie:

$$y = b_0 + b_1X_1 + b_2X_2 + ... + b_nX_n + \varepsilon$$

gdzie:
- y to zmienna zależna,
- $X_1, X_2, ..., X_n$ to zmienne niezależne,
- b_0 to wyraz wolny (intercept) mówiący nam, jaka będzie wartość zmiennej zależnej, gdy wszystkie zmienne niezależne mają wartość 0,
- $b_1, b_2, ..., b_n$ to współczynniki kierunkowe dla odpowiednich zmiennych niezależnych, pokazujące jak zmiana danej zmiennej niezależnej wpływa na zmienną zależną,
- ε to błąd modelu, uwzględniający wszystkie inne czynniki, które nie zostały uwzględnione

w modelu.

Budowa modelu wielokrotnej regresji liniowej polega na doborze takich wartości współczynników *b_0, b_1, ..., b*_n, aby możliwie najdokładniej oddać rzeczywisty związek między zmiennymi. Dokonuje się tego poprzez minimalizację błędu ε, który mówi nam, jaka jest różnica między wartościami przewidywanymi przez model a rzeczywistymi danymi.

Wielokrotna regresja liniowa pozwala na uwzględnienie wpływu wielu różnych czynników na analizowaną zmienną zależną, co czyni analizę bardziej kompleksową i precyzyjną w porównaniu do prostej regresji liniowej. Daje nam również możliwość badania, jak poszczególne zmienne niezależne wpływają na zmienną zależną, biorąc pod uwagę obecność innych zmiennych niezależnych w modelu.

Jednak warto pamiętać, że choć wielokrotna regresja liniowa jest potężnym narzędziem, wymaga ona również ostrożności. Należy na przykład sprawdzić, czy istnieją między zmiennymi jakieś ukryte zależności (korelacje), które mogłyby wpłynąć na wyniki analizy.

W praktycznym użyciu, po ustanowieniu modelu, można go wykorzystać do przewidywania wartości zmiennej zależnej na podstawie nowych danych odnośnie zmiennych niezależnych. Daje to możliwość podejmowania bardziej informowanych decyzji w oparciu o analizę.

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

```
dane = pd.read_csv('mieszkania3.csv')
dane = dane[(dane['rok'] >= 2000) & (dane['rok'] <= 2023)]
dane = dane.sample(n=30)

dane
```

`dane = pd.read_csv('mieszkania3.csv')`

Ta linia kodu wczytuje zestaw danych z pliku CSV o nazwie *mieszkania3.csv* do zmiennej DataFrame o nazwie dane. Biblioteka pandas jest tutaj używana do łatwego wczytywania i manipulacji danymi tabelarycznymi.

`dane = dane[(dane['rok'] >= 2000) & (dane['rok'] <= 2023)]`

Tutaj dokonuje się selekcji rekordów, które mieściły się między rokiem 2000 a 2023, co pozwala na skoncentrowanie się na danych z tego konkretnego okresu. Jest to użyteczne, gdy chcemy odfiltrować nieaktualne dane lub te z przyszłości, które mogą nie być jeszcze pełne.

`dane = dane.sample(n=30)`

Ta linia kodu losowo wybiera 30 rekordów z przefiltrowanych danych. Losowe próbkowanie jest przydatne, gdy chcemy zmniejszyć rozmiar danych do bardziej zarządzalnego zakresu, co może być szczególnie przydatne w przypadku dużych zestawów danych.

Przygotowanie danych

```
X = dane.iloc[:, :-1].values
y = dane.iloc[:, -1].values
```

Ten fragment kodu jest częścią procesu przygotowania danych do analizy lub modelowania. Tutaj wykorzystujemy bibliotekę pandas (wskazuje na to użycie .iloc), bardzo popularną w analizie danych w Pythonie.

dane.iloc[:, :-1] pozwala na wybranie wszystkich wierszy (:) oraz wszystkich kolumn o jedną mniej niż całkowita liczba kolumn (:-1). Oznacza to, że z DataFrame dane bierzemy wszystkie kolumny poza ostatnią. Ta operacja jest często stosowana, gdy chcemy oddzielić "cechy" (ang. features) przewidywane przez model od "etykiety" (ang. label) lub "zmiennej docelowej" (ang. target variable).

Dodanie .values na końcu pozwala na konwersję wyniku z DataFrame do tablicy numpy, co jest przydatne w wielu algorytmach uczenia maszynowego, które wymagają danych wejściowych w tym formacie.

dane.iloc[:, -1] pozwala na wybranie wszystkich wierszy (:) oraz ostatniej kolumny (-1) z DataFrame dane. To właśnie tutaj znajduje się zmienna docelowa, którą chcemy przewidywać (na przykład cena mieszkania).

Podział danych na zestaw treningowy i testowy

```
X_treningowe, X_testowe, y_treningowe, y_testowe = train_test_split(X, y, test_size=0.2, random_state=0)
```

train_test_split(X, y, ...) to funkcja podziału danych na zestawy treningowe i testowe. Jako argumenty przyjmuje dwie tablice: X (cechy) i y (zmienna docelowa), które zostały przygotowane wcześniej.

test_size=0.2 to parametr, który określa, jaką część danych chcemy przeznaczyć na zestaw testowy. W tym przypadku 0.2 oznacza 20%, co znaczy, że 80% danych zostanie użyte do treningu modelu, a 20% do jego oceny.

random_state=0 jest parametrem, który umożliwia reprodukowalność wyników przez kontrolę losowości podziału. Dzięki temu przy każdym uruchomieniu tego kodu podział danych będzie taki sam, jeśli *random_state* ma tę samą wartość.

Wynik działania funkcji to cztery zestawy danych: *X_treningowe, X_testowe, y_treningowe, y_testowe. X_treningowe i y_treningowe* stanowią cechy i zmienną docelową dla zestawu treningowego, a *X_testowe i y_testowe* - dla zestawu testowego. Te dane zostaną użyte do trenowania modelu (za pomocą zbioru treningowego) i jego oceny (za pomocą zbioru testowego).

Trening modelu

```
model = LinearRegression()
model.fit(X_treningowe, y_treningowe)
```

Pierwsza linia kodu inicjalizuje model regresji liniowej. *LinearRegression()* to konstruktor klasy *LinearRegression* z biblioteki *scikit-learn*, która implementuje model regresji liniowej. Tworzenie instancji tego modelu nie wymaga dodatkowych argumentów, co czyni go łatwym w użyciu dla podstawowych zadań predykcyjnych.

Metoda *.fit()* używana jest do "uczenia" modelu na podstawie dostarczonych danych. Jako argumenty przyjmuje zestaw danych treningowych: *X_treningowe* (cechy) i *y_treningowe* (zmienna docelowa). W przypadku regresji liniowej, proces "uczenia" polega na znalezieniu takich wartości wag (parametrów modelu), aby zminimalizować różnicę między wartościami przewidywanymi przez model a rzeczywistymi wartościami *y_treningowe*.

```
model = LinearRegression()
model.fit(X_treningowe, y_treningowe)
```

Metoda .predict() jest używana na modelu model, który został już wytrenowany wcześniej przy użyciu metody .fit(). Do metody .predict() przekazujemy jako argument X_testowe, czyli zestaw danych testowych zawierających cechy, dla których chcemy przewidzieć wartości zmiennej docelowej y.

Wynikiem działania metody .predict() jest tablica y_pred, zawierająca przewidziane przez model wartości zmiennej docelowej dla przekazanych danych testowych. Każdy element tej tablicy odpowiada prognozie dla odpowiadającego mu wiersza w X_testowe.

Przewidywanie wartości

```
porownanie = pd.DataFrame(X_testowe, columns=['m2', 'pokoje', 'rok'])
porownanie['Cena rzeczywista'] = y_testowe
porownanie['Cena przewidywana'] = np.round(y_pred,0)
porownanie['Błąd procentowy (%)'] = np.round(((porownanie['Cena rzeczywista'] - porownanie['Cena przewidywana']) / porownanie['Cena rzeczywista']) * 100, 2)
```

Ten fragment kodu tworzy *DataFrame*, który pozwala na porównanie rzeczywistych cen mieszkań z cenami przewidzianymi przez model. Oto krok po kroku, co się dzieje:

```
porownanie = pd.DataFrame(X_testowe, columns=['m2', 'pokoje', 'rok'])
```

Tworzy *DataFrame porownanie* z danych testowych *X_testowe*, które zawierają cechy dotyczące mieszkań (metraż, liczba pokoi, rok). Kolumny są nazwane odpowiednio: *['m2', 'pokoje', 'rok']*.

```
porownanie['Cena rzeczywista'] = y_testowe
```

Do *DataFrame* porownanie dodaje nową kolumnę *Cena* rzeczywista, która zawiera rzeczywiste ceny mieszkań z zestawu danych testowych (*y_testowe*).

```
porownanie['Cena przewidywana'] = np.round(y_pred,0)
```

Dodaje kolumnę *Cena* przewidywana, w której znajdują się ceny przewidziane przez model, zaokrąglone do najbliższej całkowitej liczby. Wykorzystuje się tu funkcję *numpy np.round(y_pred,0)* do zaokrąglenia przewidywań.

```
porownanie['Błąd procentowy (%)'] = np.round(((porownanie['Cena rzeczywista'] - porownanie['Cena przewidywana']) / porownanie['Cena rzeczywista']) * 100, 2)
```

Tworzy kolumnę *Błąd procentowy (%)*, która oblicza procentowy błąd przewidywania cen przez model. Błąd procentowy jest obliczany jako różnica między ceną rzeczywistą a ceną przewidzianą, podzielona przez cenę rzeczywistą, wszystko pomnożone przez 100, aby uzyskać procent. Wynik jest zaokrąglony do dwóch miejsc po przecinku.

Wizualizacja wyników

Wizualizacja wyników modelu wielokrotnej regresji liniowej, w którym mamy do czynienia z wieloma zmiennymi niezależnymi, jest znacznie bardziej skomplikowana niż w przypadku prostej regresji liniowej z jedną zmienną niezależną. Oto kilka kluczowych powodów, dla których proces ten nie jest prosty:

Wielowymiarowość danych: W przypadku wielokrotnej regresji liniowej mamy do czynienia z wieloma zmiennymi niezależnymi (czynnikami), które wpływają na zmienną zależną. Kiedy mamy więcej niż dwie czy trzy zmienne niezależne, staje się to problemem, ponieważ nie można już łatwo przedstawić wszystkich wymiarów na dwuwymiarowym wykresie kartezjańskim. Standardowe wykresy, takie jak wykresy punktowe, nie są w stanie efektywnie reprezentować złożoności modelu z wieloma wymiarami.

Interakcje między zmiennymi: Kolejnym wyzwaniem jest przedstawienie interakcji między wieloma zmiennymi niezależnymi i ich łączny wpływ na zmienną zależną. W modelach wielokrotnych zmiennych, różne kombinacje zmiennych mogą mieć różny wpływ na wyniki, co czyni jednoznaczną wizualizację trudną.

Przestrzeń wielowymiarowa: Model wielokrotnej regresji liniowej istnieje teoretycznie w przestrzeni wielowymiarowej, z jednym wymiarem na każdą zmienną niezależną plus dodatkowym wymiarem dla zmiennej zależnej. Dla ludzkiego oka i standardowych narzędzi wizualizacyjnych trudno jest zobrazować więcej niż trzy wymiary.

Wizualizacja błędów i dopasowań modelu: Nawet jeśli uda się przekształcić dane w sposób, który umożliwia przedstawienie wielu wymiarów (na przykład za pomocą technik redukcji wymiarowości), wizualizacja poziomu dopasowania modelu (na przykład poprzez przedstawienie linii regresji czy powierzchni decyzyjnej) oraz wizualizacja błędów (reszt modelu) może być nieintuicyjna i trudna do zinterpretowania.

Chociaż istnieją techniki, które pozwalają na nieco uproszczenie tego procesu (np. projekcja na wybrane płaszczyzny, wizualizacja parzysta cech, zastosowanie technik redukcji wymiarów takich jak PCA), żadne z tych rozwiązań nie zapewniają pełnej i jednoznacznej prezentacji charakterystyki modelu wielokrotnej regresji liniowej w sposób tak prosty i intuicyjny, jak jest to możliwe dla modeli jednowymiarowych. Z tego powodu analiza modeli wielokrotnych regresji liniowej często opiera się na statystykach numerycznych i metrykach jakości dopasowania, niż na bezpośredniej wizualizacji.

Obliczenie wartości dla nowych danych

```
nowe_dane = [[70, 3, 2020]]
przewidywana_cena = model.predict(nowe_dane)
print(f'Przewidywana cena (zaokrąglona): {np.round(przewidywana_cena,0)[0]} zł')
```

```
nowe_dane = [[70, 3, 2020]]
```

Definiuje nowy zestaw danych *nowe_dane*, zawierający informacje o jednym mieszkaniu. W tym przypadku, dane opisują mieszkanie o powierzchni 70 m^2, z 3 pokojami, które zostało zbudowane lub remontowane w 2020 roku. Te dane są strukturyzowane w formie listy list, tak aby pasowały do formatu, który jest oczekiwany przez metodę *.predict()* modelu, ponieważ model oczekuje danych w formie macierzy lub DataFrame.

```
przewidywana_cena = model.predict(nowe_dane)
```

Wykorzystuje metodę *.predict()* na wytrenowanym modelu model, przekazując do niej *nowe_dane.* Model wykorzystuje swoje nauczone wagę (parametry) do obliczenia przewidywanej ceny mieszkania na podstawie podanych cech.

```
print(f'Przewidywana cena (zaokrąglona): {np.round(przewidywana_cena,0)[0]} zł')
```

Wartość *przewidywana_cena*, zwrócona przez metodę *.predict(),* to tablica *numpy* zawierająca przewidywane ceny dla przekazanych próbek. Ponieważ przekazaliśmy tylko jeden zestaw danych dotyczących mieszkania, tablica *przewidywana_cena* zawiera tylko jeden element.

Wykorzystując *np.round(przewidywana_cena,0)[0]*, zaokrąglamy tę przewidywaną cenę do najbliższej całkowitej liczby i wybieramy pierwszy (i jedyny) element tablicy do wyświetlenia. Następnie zaokrąglona przewidywana cena jest wstawiana do łańcucha znaków, który jest wyświetlany, informując użytkownika o przewidywanej cenie mieszkania.

Podsumowanie

W dzisiejszym materiale skoncentrowaliśmy się na zaawansowanej analizie wielokrotnej regresji liniowej, która umożliwia zrozumienie wpływu wielu różnorodnych czynników na określoną zmienną zależną. Zaprojektowana, by wykraczać poza ograniczenia modelu z jedną zmienną niezależną, wielokrotna regresja liniowa oferuje znacznie szerszy wgląd w złożone zależności między danymi. Szczególny nacisk położyliśmy na wyzwania, jakie niesie ze sobą wizualizacja takich wielowymiarowych modeli. Ze względu na ograniczenia ludzkiego postrzegania i tradycyjnych metod graficznych, wizualizacja modeli obejmujących więcej niż dwie czy trzy zmienne staje się nieintuicyjna, a nawet niemożliwa do jednoznacznej reprezentacji. To podkreśla ważność stosowania bardziej zaawansowanych technik analizy oraz numerycznych metod oceny modelu. Mimo że wizualizacja wielowymiarowa jest wyzwaniem, zrozumienie i efektywne stosowanie wielokrotnej regresji liniowej otwiera nowe możliwości analizy danych, pozwalając na głębsze i bardziej wielowarstwowe wnioski, które mogą mieć kluczowe znaczenie w różnorodnych zastosowaniach praktycznych.