Inżynieria uczenia maszynowego

Ewa Roszczyk, Karolina Romanowska

14 stycznia 2022

1 Wstęp

W ramach projektu otrzymałyśmy poniższy problem zgłoszony przez klienta:

Mamy problemy z odpowiednim zapełnianiem półek magazynowych. Nigdy nie wiadomo, co tak naprawdę będzie potrzebne w najbliższym tygodniu, co powinniśmy zamówić. Może da się coś z tym zrobić?

2 Implementacja mikroserwisu

Jednym z celów projektu była implementacja mikroserwisu, z którego można uzyskać predykcje ilości sprzedanych w danym tygodniu produktów. Podczas startu serwisu następuje uczenie się modeli predykcji na podstawie danych znajdujących się w bazie danych.

2.1 Dostępne endpointy

W ramach mikroserwisu dostępne są endpointy

Endpoint	**Opis**	**Parametry**
/	Przekierowuje na stronę dokumentacji mikroserwisu	-
/{username}/prediction	Podaje listę predykcji ilości zakupionych produktów w tygodniu	username: string, products: query, "all", date: format $\%d-\%m-\%y$

Tabela 1: Endpointy typu GET

Endpoint	**Opis**	**Request body**
/upload-sessions	Pozwala na dodanie kolejnych logów sesji	sessions: plik binarny
/new-user	Pozwala na dodanie użytkownika	username: string

Tabela 2: Endpointy typu POST

2.2 Baza danych

W projekcie znajduje się plikowa baza danych, zawierająca pliki:

- predictions.csv - przechowująca predykcje zwrócone przez model
- products.jsonl - przechowująca informacje o produktach sprzedawanych przez sklep
- sessions.jsonl - przechowująca sesje użytkowników
- users.csv - przechowująca listę użytkowników pobierających predykcję wraz z przypisanymi do nich grupami

2.3 Test A/B

W ramach serwisu zaimplementowane zostały testy A/B. Podczas dodawania nowego użytkownika przez endpoint /new-user zostaje mu automatycznie przypisana grupa. Następnie, podczas korzystania z endpointu /{username}/prediction użytkownik podaje swoją nazwę użytkownika. Po sprawdzeniu do której grupy należy użytkownik, zwracana jest odpowiednia predykcja. Logi z informacjami o rodzaju grupy oraz predykcji znajdują się w database/predictions.csv

2.4 Testowanie

W pliku test_endpoints.py znajdują się testy poszczególnych endpointów. Dodatkowo w projekcie umieszczono plik examples.http z przykładowymi zapytaniami do serwisu. Testy manulane przeprowadzono zarówno przy pomocy wtyczki do Visual Studio Code "REST Client" jak i aplikacji Postman.

2.5 Działanie

2.5.1 Działanie endpointu /{username}/prediction

```
Zapytanie zawierające listę id produktów
----------------------------------------
Request:
GET http://localhost:8000/Tadeusz/prediction?products=1116,1117 HTTP/1.1
content-type: application/json

Response:
HTTP/1.1 200 OK
date: Mon, 10 Jan 2022 23:58:56 GMT
server: uvicorn
content-length: 58
content-type: application/json
Connection: close

{
  "1116": 0.047911754548962994,
  "1117": 0.0009184690382460479
}

Zapytanie zawierające listę produktów, gdzie jeden z nich jest nieprawidłowy
----------------------------------------------------------------------------
Request:
```

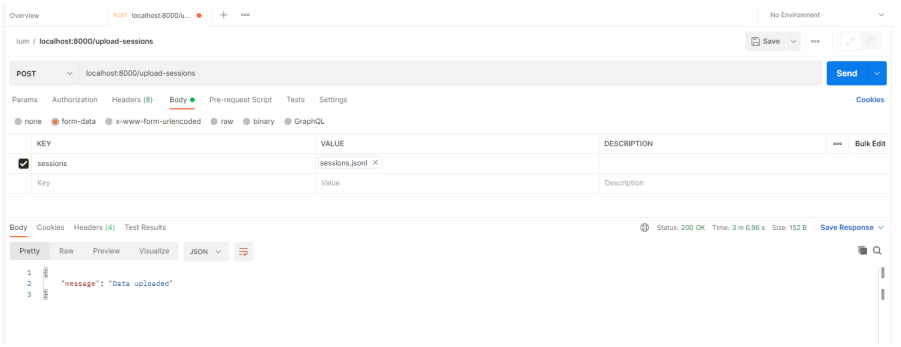
```
GET http://localhost:8000/Tadeusz/prediction?products=1116,115 HTTP/1.1
content-type: application/json

Response:
HTTP/1.1 200 OK
date: Tue, 11 Jan 2022 00:01:42 GMT
server: uvicorn
content-length: 29
content-type: application/json
Connection: close

{
  "1116": 0.047911754548962994
}

Zapytanie zawierające parametr products="all"
---------------------------------------------
Request:
GET http://localhost:8000/Karolina/prediction?products=all HTTP/1.1
content-type: application/json

Response:
HTTP/1.1 200 OK
date: Tue, 11 Jan 2022 00:03:02 GMT
server: uvicorn
content-length: 6590
content-type: application/json
Connection: close

{
  "1001": 0.0,
  "1002": 1.1742376081825339,
  "1003": 4.483184893784422,
  "1004": 1.1299134539732494,
  "1005": 0.8400314712824548,
  "1006": 2.2003241542092837,
  "1007": 2.632355625491739,
...
  "1316": 0.0,
  "1317": 0.0,
  "1318": 0.0,
  "1319": 0.0
}
```

2.5.2 Działanie endpointu /new-user

```
Request:
POST http://localhost:8000/new-user HTTP/1.1
content-type: application/json

{
```

```
    "username": "Test"
}

Response:
HTTP/1.1 200 OK
date: Tue, 11 Jan 2022 00:12:03 GMT
server: uvicorn
content-length: 50
content-type: application/json
Connection: close

{
  "message": "Username Test added. Your group is B"
}

Response po kolejnym Request:
HTTP/1.1 200 OK
date: Tue, 11 Jan 2022 00:12:58 GMT
server: uvicorn
content-length: 52
content-type: application/json
Connection: close

{
  "message": "Sorry, user with that username exists."
}
```

2.5.3 Działanie endpointu /upload-sessions



Rysunek 1: Wykonanie zapytania

3 Implementacja modeli

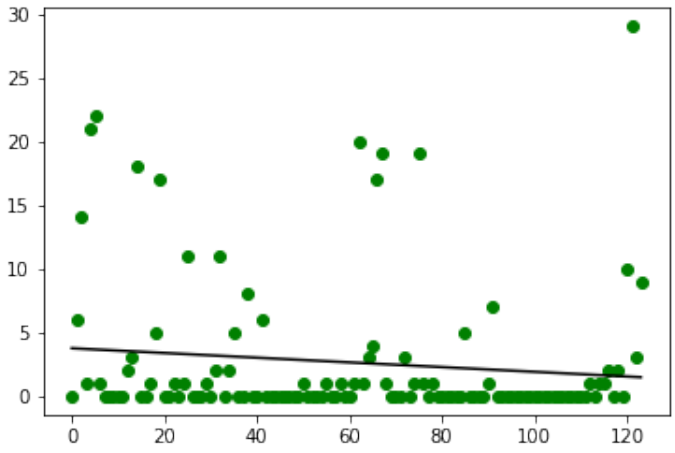
W ramach projektu zostały zaimplementowane dwa modele, przewidujące jakie produkty będą potrzebne w danym tygodniu, dzięki czemu klient, zgłaszający problem, będzie mógł odpowiednio zapełniać półki magazynowe. Zgodnie z wymaganiami zaimplementowane zostały dwa modele - model bazowy oraz model bardziej zaawansowany. Jako model bazowy zaimplementowałyśmy metodę regresji liniowej a jako model bardziej zaawansowany - metodę ARIMA.

3.1 Model regresji liniowej

Modelem bazowym w naszej implementacji jest model regresji liniowej. Kombinacje zmiennych i parametrów dopasowują model do danych. Dzięki temu dopasowana krzywa regresji, wyznacza przyszłe wartości sprzedaży dla danego produktu.

Podczas implementaji tego modelu posłużyłyśmy się biblioteką *sklearn*. Za pomocą metody najmniejszych kwadratów, model regresji liniowej jest dopasowywany do danych wejściowych.

Poniżej znajduje się wykres dopasowanego modelu do danych zawierających liczbę sprzedaży w danym tygodniu. Wykres został stworzony dla produktu o id *1291*, czyli dla produktu *Philips SDV6224*. Zgodnie z naszą analizą danych otrzymanych od klienta, to właśnie ten produkt jest najchętniej kupowany przez klientów sklepu.



Rysunek 2: Model regresji liniowej

3.2 ARIMA

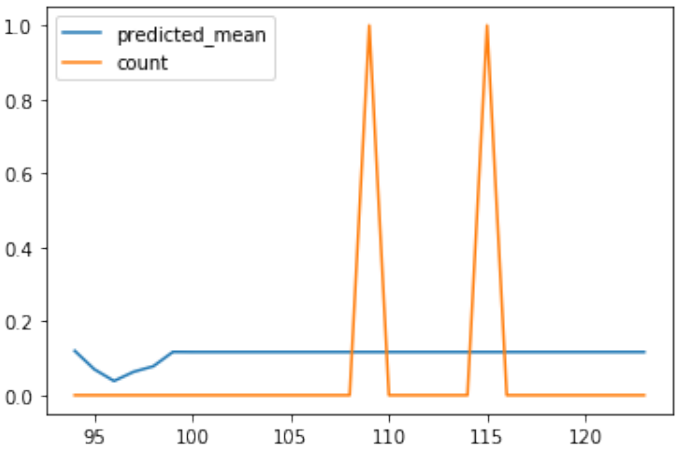
ARIMA w naszej implementacji jest modelem bardziej zaawansowanym. ARIMA to skrót od angielskiej nazwy modelu *autoregressive integrated moving average*, czyli *autoregresyjnej zintegrowanej średniej ruchomej*. Metoda ta pozwala nam na dokladne modelowanie szeregów czasowych.

Podczas implementacji tej metody posłużyłymy się biblioteką *statsmodels*, z której zaimportowałyśmy ARIMA. Metoda ta najpierw identyfikuje 3 parametry:

- p - parametr autoregresyjny,
- d - rząd różnicowania,
- q - parametr średniej ruchomej.

Arima sprawdza, czy nasze dane są stacjonarne. Po tej analizie wyświetli wyniki dla naszych danych. Wyniki interpretujemy tak samo jak dla regresji liniowej.

Poniżej znajduje się przykładowy wykres dla produktu o id *1114*. Przedstawia on prawdopodobieństwo zakupu produktu, przewidziane przez wyuczony model ARIMA w porównaniu do faktycznych danych dla zbioru testującego.



Rysunek 3: ARIMA