Projekt IUM - 2021Z - Zad.3

Aleksander Garlikowski
Cezary Moczulski

Opisy modeli i procesy ich powstawania:

Model bazowy polega na wyliczeniu średniej wartości czasu trwania dostawy dla każdej kombinacji: miasta docelowego, firmy transportowej zajmującej się dostawą i dnia tygodnia zakupu i późniejszym zwracaniu przewidywań będących parami wartości, wyznaczającymi symetryczne okna czasowe o określonej szerokości wokół odpowiednich średnich. Dla okna o szerokości 32h, precyzja przewidywań znajduje się na poziomie ok. 90%.

Model docelowy jest dosyć prostą siecią neuronową, zaimplementowaną z użyciem modułu Pytorch, przyjmuje te same dane co model bazowy w postaci kodów 1 z n. Każda z wartości jest przetwarzana przez oddzielną warstwę odpowiadającą za jej zanurzenie, wyniki są później łączone w pojedynczy wektor i przetwarzane przez resztę sieci. Podobnie jak w przypadku modelu bazowego, przewidywania zwracane przez model docelowy mają postać par wartości, będących krawędziami okna czasowego, w którym ma odbyć się dostarczenie produktu.

Proces powstawania modelu docelowego skupiał się nie tyle na modyfikacjach samej sieci neuronowej, co na dostosowywaniu funkcji straty wykorzystywanej w procesie uczenia. Ze względu na to, że model miał za zadanie zwrócić zakres wartości, do którego należy czas trwania dostawy, wykorzystanie funkcji dostępnych w module Pytorch okazało się problematyczne. Ze względu na to, powstało kilka autorskich funkcji straty.

Pierwsza funkcja straty miała następującą postać:

$baseLoss * \left(2 - inBetween - \frac{1}{high - low}\right)$, gdzie *baseLoss* to parameter podany funkcji straty z zewnątrz, *low* i *high* to odpowiednio początek i koniec zakresu wyznaczonego przez model, a *inBetween* przyjmuje następujące wartości:

$inBetween = \begin{cases} 1, & low \leq deliveryDuration \leq high \\ 0, & w.p.p \end{cases}$, *deliveryDuration* jest faktycznym czasem trwania dostawy odczytanym z danych.

Niestety ta funkcja okazała się bezużyteczna dla nauki modelu.

Następna funkcja, która już mogła być użyta miała postać:

$w_1 * \left(\frac{high+low}{2} - deliveryDuration\right)^2 + w_2 * (high - low)^2 + w_3 * (-high + low)$,

w_1, w_2 i w_3 to parametry podane z zewnątrz, znaczenie pozostałych zmiennych jest takie samo jak w poprzednim wzorze. Na tym etapie można było już trenować model z różnym naciskiem na zmniejszanie odległości pomiędzy środkiem wyznaczanych przedziałów, a wartościami czasów trwania dostawy z danych treningowych, zmniejszanie szerokości wyznaczanego okna czasowego, a zachowywaniem logicznej poprawności wartości *high* i *low.* Niestety ze względu na to, że każdy z tych parametrów wpływa na szerokość przewidywanego zakresu czasu, przewidywanie go okazało się niemożliwe.

Na tym etapie odbyły się eksperymenty ze strukturą samej sieci neuronowej:

- Sprawdzanie użyteczności poszczególnych elementów poprzez ich wyłączanie skutkowało obniżeniem skuteczności modelu.
- Dodawanie dodatkowych warstw sieci w celu poszukiwania bardziej złożonych zależności okazało się nie mieć wpływu na jakość przewidywań, więc dodatkowe elementy zostały usunięte w celu redukcji ilości obliczeń potrzebnych do wytrenowania modelu.
- Dodanie skwantowanego czasu zakupu do danych wykorzystywanych do przewidywań, okazało się nieznacznie poprawić precyzję modelu dla wąskich przedziałów czasu, ale ze względu na to, że pozostała nieakceptowalnie niska (ok. 38% dla okien o szerokości 8-9h), zmiana została cofnięta.

Ze względu na to, że druga funkcja celu bezwzględnie promowała przedziały czasu symetryczne wokół wartości z próbek treningowych i uniemożliwiała przewidywalne manipulowanie długością przedziałów wyznaczanych przez model, powstała trzecia - ostatnia funkcja straty:

$[low - \max(deliveryDuration - lowShift, 0)]^2 + [high - (deliveryDuration + highShift)]^2$, *lowShift* i *highShift* to parametry, które powinny przyjmować wartości dodatnie i pozwalają na wytrenowanie modelu przewidującego przedziały o dowolnym kształcie, a przewidzenie przybliżonej wartości średniej długości przedziałów czasu wygenerowanych przez model, sprowadza się do zsumowania ich.

Ostatecznie model docelowy osiągnął precyzję na poziomie 90% dla przedziału czasu symetrycznego wokół wartości próbek treningowych, o średniej długości 31.5h, co jest nieznacznie lepszym wynikiem od modelu bazowego, jednakże w przeciwieństwie do niego, pozwala na łatwe dodanie nowych rodzajów danych do zbioru wykorzystywanego do przewidywań, co pozwala na znaczenie szybsze dostosowanie się do zmian sytuacji, w porównaniu do modelu bazowego.

Materiały pokazujące, że implementacja działa:

Obróbka danych:

Surowe dane (jeden z 4 plików):


```
{"purchase_id": 20001, "purchase_timestamp": "2021-04-06T05:37:52", "delivery_timestamp": "2021-04-08T08:44:56.801023", "delivery_company": 620}
{"purchase_id": 20002, "purchase_timestamp": "2021-01-22T11:09:33", "delivery_timestamp": "2021-01-25T14:59:58.824197", "delivery_company": 360}
{"purchase_id": 20003, "purchase_timestamp": "2021-08-26T17:23:07", "delivery_timestamp": "2021-08-28T08:22:45.371701", "delivery_company": 620}
{"purchase_id": 20004, "purchase_timestamp": "2021-02-23T17:49:06", "delivery_timestamp": "2021-02-25T12:00:33.323784", "delivery_company": 360}
{"purchase_id": 20005, "purchase_timestamp": "2021-10-17T04:59:27", "delivery_timestamp": "2021-10-18T08:20:47.604831", "delivery_company": 516}
```


Skrypt obrabiający dane:

```
PS C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time> python ./delivery_time/data/process_data.py data/raw data/processed
2022-01-14 11:46:13,540 - __main__ - INFO - making final data set from raw data
2022-01-14 11:46:21,913 - __main__ - INFO - loaded data, example: ['Gaja Sadlak', 'Gdynia', 'plac Stawowa 21', 'Plantronics Savi W710', 'Sprzęt RTV;Audio;Słuchawki', 553.0, '2021-04-06T05:37:52', 15, '2021-04-06T05:37:52', '2021-04-08T08:44:56.801023', 620]
2022-01-14 11:46:22,116 - __main__ - INFO - splitting data into sets; all: 7119, train: 5695, test: 1424
2022-01-14 11:46:22,138 - __main__ - INFO - train data saved successfully, dir: C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time\data/processed\train_data.csv
2022-01-14 11:46:22,144 - __main__ - INFO - test data saved successfully, dir: C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time\data/processed\test_data.csv
PS C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time>
```

Przetworzone dane (dzień tygodnia, miasta (7 pozycji), firma kurierska (3 pozycje), czas dostawy):

```
weekday,Gdynia,Kraków,Poznań,Radom,Szczec
4,0,0,1,0,0,0,0,1,0,0,79.44333333333333
4,0,0,0,0,0,1,0,0,0,1,87.52555555555556
4,0,0,0,0,0,1,0,0,1,0,78.05833333333334
4,0,0,0,0,0,0,1,1,0,0,77.58305555555556
```

Model naiwny:

Trening:

```
PS C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time> python ./delivery
_time/models/train_model_naive.py data/processed/train_data.csv models/naive/naive_model.csv
2022-01-14 11:50:41,560 - __main__ - INFO - training naive model
2022-01-14 11:50:41,575 - __main__ - INFO - loaded data, example:
    weekday  Gdynia  Kraków  Poznań  Radom  Szczecin  Warszawa  Wrocław  360  516  620  deliv
ery_time
0        4       0       0       1      0         0         0        0    1    0    0      79
.443333
2022-01-14 11:50:41,930 - __main__ - INFO - trained model
2022-01-14 11:50:41,933 - __main__ - INFO - model saved successfully, dir: C:\Users\alekg\Des
ktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time\models/naive/naive_model.csv
```

Przewidywanie dla konkretnej próbki za pomocą skryptu:

```
PS C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time> python ./delivery
_time/models/predict_model_naive.py models/naive/naive_model.csv "4,0,0,1,0,0,0,0,1,0,0"
(53.01198, 85.31198)
PS C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time>
```

Przewidywanie dla tej samej próbki za pośrednictwem serwera aplikacji:

```
PS C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time\data> curl 127.0.0.
1:8000/naive/ -d "sample=4,0,0,1,0,0,0,0,1,0,0"
"{\"pred_time_from\": 53.01198, \"pred_time_to\": 85.31198}"
PS C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time\data>
```

Regresor:

Trening:

```
PS C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time> python ./delivery
_time/models/train_model_regressor.py data/processed/train_data.csv models/regressor/regresso
r_model.pt
2022-01-14 11:52:52,307 - __main__ - INFO - training regressor model
2022-01-14 11:52:52,323 - __main__ - INFO - loaded data, example:
   weekday  Gdynia  Kraków  Poznań  Radom  ...  Wrocław  360  516  620  delivery_time
0        4       0       0       1      0  ...        0    1    0    0      79.443333

[1 rows x 12 columns]
2022-01-14 11:53:06,200 - __main__ - INFO - trained model
2022-01-14 11:53:06,202 - __main__ - INFO - model saved successfully, dir: C:\Users\alekg\Des
ktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time\models/regressor/regressor_model.pt
PS C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time>
```

Przewidywanie dla konkretnej próbki za pomocą skryptu:

```
2022-01-14 11:53:06,202 - __main__ - INFO - model saved successfully, dir: C:\Users\alekg\Des
ktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time\models/regressor/regressor_model.pt
PS C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time> python ./delivery
_time/models/predict_model_regressor.py models/regressor/regressor_model.pt "4,0,0,1,0,0,0,0,
1,0,0"
(57.884117126464844, 89.2145004272461)
PS C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time>
```

Przewidywanie dla tej samej próbki za pośrednictwem serwera aplikacji:

```
PS C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time\data> curl 127.0.0.
1:8000/regressor/ -d "sample=4,0,0,1,0,0,0,0,1,0,0"
"{\"pred_time_from\": 57.884117126464844, \"pred_time_to\": 89.2145004272461}"
PS C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time\data>
```

Serwer:

Inicjalizacja i pojedyncze zapytanie na główny endpoint:

```
PS C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time> endpoints --prefi
x=controllers --host=localhost:8000
[property.application] Caching value in _application
[property.application] Caching value in _application
[property.application] Checking cache for _application
[property.backend] Caching value in _backend
[property.backend] Caching value in _backend
[property.backend] Caching value in _backend
[property.backend] Caching value in _backend
[property.backend] Checking cache for _backend
Listening on 127.0.0.1:8000
[property.backend] Checking cache for _backend
[property.path] Caching value in _path
[property.query] Caching value in _query
[property.encoding] Caching value in _encoding
[property.encoding] Caching value in _encoding
[property.kwargs] Caching value in _kwargs
[property.kwargs] Caching value in _kwargs
[property.args] Caching value in _args
[property.args] Caching value in _args
[property.path] Checking cache for _path
Searching for Controller using path: /
[property.path] Checking cache for _path
[property.path_args] Caching value in _path_args
[property.path_args] Caching value in _path_args
request server started
[param] __new__ arguments are not wrappable, so __call__ is the wrap call
[param] __call__ is the wrap call
[param] Calling decorate_func()
[property.query] Checking cache for _query
[property.query_kwargs] Caching value in _query_kwargs
[property.query_kwargs] Caching value in _query_kwargs
[property.query] Checking cache for _query
[property.path] Checking cache for _path
Request method: POST /
Request date: 2022-01-14T10:58:03.901093
Request header Host: 127.0.0.1:8000
Request header User-Agent: curl/7.55.1
Request header Accept: */*
Request header Content-Type: application/x-www-form-urlencoded
Request header Content-Length: 73
Request body: {'purchase_timestamp': '2021-05-31T19:39:23', 'delivery_company': '360', 'city'
: 'Warszawa'}
Request handle method: controllers.Default.handle
Request Controller method: controllers.Default.POST
<class 'int'>
main endpoint prediction: user_id: None, data: 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, prediction:(5
2.52718000000001, 84.82718)
Request response header Content-Type: application/json;charset=UTF-8
Request response 200 OK in 963.8 ms
127.0.0.1 - - [14/Jan/2022 11:58:04] "POST / HTTP/1.1" 200 69
```

Log z zapytaniami na każdy z endpointów:

```
2022-01-14 11:58:04 - server_logger - INFO - request server started
2022-01-14 11:58:04 - server_logger - INFO - main endpoint prediction: user_id: None, data: 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, prediction:(52.52718000000001, 84.82718)
2022-01-14 12:00:53 - server_logger - INFO - naive endpoint prediction: data: 4,0,0,1,0,0,0,0,1,0,0, prediction:(53.01198, 85.31198)
2022-01-14 12:01:36 - server_logger - INFO - regressor endpoint prediction: data: 4,0,0,1,0,0,0,0,1,0,0, prediction:(57.884117126464844, 89.2145004272461)
```

Testowanie celności:

Model naiwny:

```
PS C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time> python ./delivery_
time/validation/test_acc.py http://127.0.0.1:8000/naive data/processed/test_data.csv
2022-01-14 12:05:54,287 - __main__ - INFO - testing model at http://127.0.0.1:8000/naive for a
ccuracy
2022-01-14 12:05:54,308 - __main__ - INFO - loaded data, example:
   weekday  Gdynia  Kraków  Poznań  Radom  ...  Wrocław  360  516  620  delivery_time
0        5       1       0       0      0  ...        0    0    1    0      37.572778

[1 rows x 12 columns]
2022-01-14 12:06:10,372 - __main__ - INFO - model at http://127.0.0.1:8000/naive achieved accu
racy of 0.9009831460674157 with average window size 32.273152912219 56h
PS C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time>
```

Regresor:

```
PS C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time> python ./delivery_
time/validation/test_acc.py http://127.0.0.1:8000/regressor data/processed/test_data.csv
2022-01-14 12:07:27,872 - __main__ - INFO - testing model at http://127.0.0.1:8000/regressor f
or accuracy
2022-01-14 12:07:27,884 - __main__ - INFO - loaded data, example:
   weekday  Gdynia  Kraków  Poznań  Radom  ...  Wrocław  360  516  620  delivery_time
0        5       1       0       0      0  ...        0    0    1    0      37.572778

[1 rows x 12 columns]
2022-01-14 12:07:40,659 - __main__ - INFO - model at http://127.0.0.1:8000/regressor achieved
accuracy of 0.8925561797752809 with average window size 31.6702701352285 52h
PS C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time>
```

Logi serwera po testach (bardzo dużo zapytań):

```
2022-01-14 12:06:10 - server_logger - INFO - naive endpoint prediction: data: 2,0,0,1,0,0,0,0,1,0,0, prediction:(37.53296, 69.83296)
2022-01-14 12:06:10 - server_logger - INFO - naive endpoint prediction: data: 2,0,0,1,0,0,0,0,0,0,1, prediction:(30.14217, 62.44217)
2022-01-14 12:06:10 - server_logger - INFO - naive endpoint prediction: data: 2,0,0,0,1,0,0,0,1,0,0, prediction:(25.976057000000004, 58.276057)
2022-01-14 12:06:10 - server_logger - INFO - naive endpoint prediction: data: 2,0,1,0,0,0,0,0,1,0,0, prediction:(11.73159, 44.031589999999994)
2022-01-14 12:06:10 - server_logger - INFO - naive endpoint prediction: data: 2,0,1,0,0,0,0,0,0,0,1, prediction:(0.41820700000000244, 32.718207)
2022-01-14 12:06:10 - server_logger - INFO - naive endpoint prediction: data: 2,0,1,0,0,0,0,0,0,0,1, prediction:(0.41820700000000244, 32.718207)
2022-01-14 12:06:10 - server_logger - INFO - naive endpoint prediction: data: 2,0,0,0,1,0,0,0,0,1,0, prediction:(17.793620000000004, 50.09362)
2022-01-14 12:06:10 - server_logger - INFO - naive endpoint prediction: data: 2,0,1,0,0,0,0,0,0,0,1, prediction:(0.41820700000000244, 32.718207)
2022-01-14 12:06:10 - server_logger - INFO - naive endpoint prediction: data: 2,0,0,0,1,0,0,0,0,1,0, prediction:(17.793620000000004, 50.09362)
2022-01-14 12:06:10 - server_logger - INFO - naive endpoint prediction: data: 2,0,0,0,0,0,1,0,0,0,1, prediction:(56.46687000000001, 88.76687000000001)
2022-01-14 12:06:10 - server_logger - INFO - naive endpoint prediction: data: 2,0,1,0,0,0,0,0,0,0,1, prediction:(0.41820700000000244, 32.718207)
2022-01-14 12:06:10 - server_logger - INFO - naive endpoint prediction: data: 2,0,0,0,0,0,1,0,0,0,1, prediction:(56.46687000000001, 88.76687000000001)
2022-01-14 12:06:10 - server_logger - INFO - naive endpoint prediction: data: 2,0,1,0,0,0,0,0,0,1,0, prediction:(0, 29.411414999999998)
2022-01-14 12:06:10 - server_logger - INFO - naive endpoint prediction: data: 2,1,0,0,0,0,0,0,1,0,0, prediction:(28.220959999999998, 60.520959999999995)
2022-01-14 12:06:10 - server_logger - INFO - naive endpoint prediction: data: 2,0,0,0,0,0,1,0,0,0,1, prediction:(56.46687000000001, 88.76687000000001)
2022-01-14 12:07:27 - server_logger - INFO - regressor endpoint prediction: data: 5,1,0,0,0,0,0,0,0,1,0, prediction:(30.404287338256836, 62.37874221801758)
2022-01-14 12:07:27 - server_logger - INFO - regressor endpoint prediction: data: 5,0,0,0,0,0,1,0,0,0,1, prediction:(46.51287078857422, 78.75843811035156)
2022-01-14 12:07:27 - server_logger - INFO - regressor endpoint prediction: data: 5,0,1,0,0,0,0,0,1,0,0, prediction:(27.33431625366211, 59.17072677612305)
2022-01-14 12:07:27 - server_logger - INFO - regressor endpoint prediction: data: 5,0,0,0,0,1,0,0,1,0,0, prediction:(27.119625091552734, 58.946250915527344)
2022-01-14 12:07:27 - server_logger - INFO - regressor endpoint prediction: data: 5,0,0,0,0,0,1,0,0,0,1, prediction:(46.51287078857422, 78.75843811035156)
2022-01-14 12:07:27 - server_logger - INFO - regressor endpoint prediction: data: 5,0,0,0,1,0,0,0,0,1,0, prediction:(29.338138580322266, 61.26486587524414)
2022-01-14 12:07:27 - server_logger - INFO - regressor endpoint prediction: data: 5,0,0,1,0,0,0,0,0,0,1, prediction:(36.563350677490234, 68.79546356201172)
2022-01-14 12:07:27 - server_logger - INFO - regressor endpoint prediction: data: 5,0,0,0,0,1,0,0,0,0,1, prediction:(27.00346565246582, 58.82481002807617)
2022-01-14 12:07:27 - server_logger - INFO - regressor endpoint prediction: data: 6,0,1,0,0,0,0,0,1,0,0, prediction:(15.321027755737305, 47.423248291015625)
2022-01-14 12:07:27 - server_logger - INFO - regressor endpoint prediction: data: 6,0,0,0,0,1,0,0,0,0,1, prediction:(21.804122924804688, 53.89912414550781)
2022-01-14 12:07:27 - server_logger - INFO - regressor endpoint prediction: data: 6,0,1,0,0,0,0,0,1,0,0, prediction:(15.321027755737305, 47.423248291015625)
2022-01-14 12:07:28 - server_logger - INFO - regressor endpoint prediction: data: 6,0,1,0,0,0,0,0,1,0,0, prediction:(15.321027755737305, 47.423248291015625)
2022-01-14 12:07:28 - server_logger - INFO - regressor endpoint prediction: data: 6,0,0,1,0,0,0,0,0,1,0, prediction:(35.232112884521484, 67.23388671875)
2022-01-14 12:07:28 - server_logger - INFO - regressor endpoint prediction: data: 6,0,0,0,0,1,0,0,0,0,1, prediction:(21.804122924804688, 53.89912414550781)
```

Symulacja ruchu sieciowego:

Uruchomienie skryptu i rezultaty w logu serwera:

```
PS C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time> python delivery_ti
me/validation/simulate_traffic.py
2022-01-14 12:14:04,195 - __main__ - INFO - simulating traffic at http://127.0.0.1:8000/
2022-01-14 12:15:04,414 - __main__ - INFO - traffic simulation finished
PS C:\Users\alekg\Desktop\Praca_zdalna\IUM\Projekt\testy\ium_delivery_time>
```

```
2022-01-14 12:07:40 - server_logger - INFO - regressor endpoint prediction: data: 2,1,0,0,0,0,0,0,1,0,0, prediction:(27.93361473083496, 59.92992401123047)
2022-01-14 12:07:40 - server_logger - INFO - regressor endpoint prediction: data: 2,0,0,0,0,0,1,0,0,0,1, prediction:(57.33210754394531, 89.47933197021484)
2022-01-14 12:14:04 - server_logger - INFO - main endpoint prediction: user_id: 1000000009, data: 4, 1, 0, 0, 0, 0, 0, 1, 0, 0, prediction:(47.42362, 79.72362)
2022-01-14 12:14:04 - server_logger - INFO - main endpoint prediction: user_id: 1000000006, data: 4, 0, 1, 0, 0, 0, 0, 1, 0, 0, prediction:(19.706415, 52.006415)
2022-01-14 12:14:05 - server_logger - INFO - main endpoint prediction: user_id: 1000000008, data: 4, 0, 0, 1, 0, 0, 0, 0, 0, 1, prediction:(49.48094, 81.78093999999999)
2022-01-14 12:14:05 - server_logger - INFO - main endpoint prediction: user_id: 1000000001, data: 4, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, prediction:(42.73318300000004, 75.03318300000001)
2022-01-14 12:14:05 - server_logger - INFO - main endpoint prediction: user_id: 1000000005, data: 4, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, prediction:(21.50004, 53.800039999999996)
2022-01-14 12:14:06 - server_logger - INFO - main endpoint prediction: user_id: 1000000008, data: 4, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, prediction:(0.7370429999999999, 33.037043)
2022-01-14 12:14:06 - server_logger - INFO - main endpoint prediction: user_id: 1000000002, data: 4, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, prediction:(21.50004, 53.800039999999996)
2022-01-14 12:14:07 - server_logger - INFO - main endpoint prediction: user_id: 1000000000, data: 4, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, prediction:(38.77501, 71.07500999999999)
2022-01-14 12:14:07 - server_logger - INFO - main endpoint prediction: user_id: 1000000003, data: 4, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, prediction:(55.53994000000001, 87.83994000000001)
2022-01-14 12:14:08 - server_logger - INFO - main endpoint prediction: user_id: 1000000009, data: 4, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, prediction:(10.359033, 42.659032999999994)
2022-01-14 12:14:08 - server_logger - INFO - main endpoint prediction: user_id: 1000000006, data: 4, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, prediction:(42.73318300000004, 75.03318300000001)
2022-01-14 12:14:09 - server_logger - INFO - main endpoint prediction: user_id: 1000000000, data: 4, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, prediction:(10.359033, 42.659032999999994)
2022-01-14 12:14:09 - server_logger - INFO - main endpoint prediction: user_id: 1000000006, data: 4, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, prediction:(14.386802000000003, 46.686802)
2022-01-14 12:14:10 - server_logger - INFO - main endpoint prediction: user_id: 1000000000, data: 4, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, prediction:(31.418750000000003, 63.71875)
2022-01-14 12:14:10 - server_logger - INFO - main endpoint prediction: user_id: 1000000002, data: 4, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, prediction:(19.706415, 52.006415)
2022-01-14 12:14:10 - server_logger - INFO - main endpoint prediction: user_id: 1000000000, data: 4, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, prediction:(0.7370429999999999, 33.037043)
2022-01-14 12:14:11 - server_logger - INFO - main endpoint prediction: user_id: 1000000009, data: 4, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, prediction:(30.549986000000004, 62.849986)
2022-01-14 12:14:11 - server_logger - INFO - main endpoint prediction: user_id: 1000000003, data: 4, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, prediction:(53.01198, 85.31198)
2022-01-14 12:14:12 - server_logger - INFO - main endpoint prediction: user_id: 1000000004, data: 4, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, prediction:(26.571775000000002, 58.871775)
2022-01-14 12:14:13 - server_logger - INFO - main endpoint prediction: user_id: 1000000002, data: 4, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, prediction:(47.42362, 79.72362)
2022-01-14 12:14:13 - server_logger - INFO - main endpoint prediction: user_id: 1000000001, data: 4, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, prediction:(10.359033, 42.659032999999994)
2022-01-14 12:14:13 - server_logger - INFO - main endpoint prediction: user_id: 1000000004, data: 4, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, prediction:(48.454164, 80.754164)
2022-01-14 12:14:14 - server_logger - INFO - main endpoint prediction: user_id: 1000000002, data: 4, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, prediction:(30.549986000000004, 62.849986)
2022-01-14 12:14:15 - server_logger - INFO - main endpoint prediction: user_id: 1000000003, data: 4, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, prediction:(30.549986000000004, 62.849986)
2022-01-14 12:14:15 - server_logger - INFO - main endpoint prediction: user_id: 1000000007, data: 4, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, prediction:(14.386802000000003, 46.686802)
2022-01-14 12:14:16 - server_logger - INFO - main endpoint prediction: user_id: 1000000007, data: 4, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, prediction:(25.860853, 58.160852999999996)
2022-01-14 12:14:16 - server_logger - INFO - main endpoint prediction: user_id: 1000000002, data: 4, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, prediction:(10.359033, 42.659032999999994)
2022-01-14 12:14:16 - server_logger - INFO - main endpoint prediction: user_id: 1000000009, data: 4, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, prediction:(55.90022, 88.20022)
2022-01-14 12:14:17 - server_logger - INFO - main endpoint prediction: user_id: 1000000001, data: 4, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, prediction:(14.386802000000003, 46.686802)
2022-01-14 12:14:17 - server_logger - INFO - main endpoint prediction: user_id: 1000000002, data: 4, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, prediction:(50.51406000000001, 82.81406000000001)
2022-01-14 12:14:18 - server_logger - INFO - main endpoint prediction: user_id: 1000000009, data: 4, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, prediction:(30.549986000000004, 62.849986)
```

Domyślne zachowanie: wszystkie zapytania na główny endpoint kierowane są do modelu A (naiwnego)

```
2999999996), path of model used:models/naive/naive_model.csv
54), path of model used:models/naive/naive_model.csv
 63.71875), path of model used:models/naive/naive_model.csv
2999999994), path of model used:models/naive/naive_model.csv
82.81406000000001), path of model used:models/naive/naive_model.csv
82.81406000000001), path of model used:models/naive/naive_model.csv
), path of model used:models/naive/naive_model.csv
), path of model used:models/naive/naive_model.csv
), path of model used:models/naive/naive_model.csv
999999999), path of model used:models/naive/naive_model.csv
2999999994), path of model used:models/naive/naive_model.csv
```

Z włączonymi testy A/B przy podziale 0.5 i dodatkowym wypisem ścieżki do modelu, żeby je odróżnić:

```
79.02374267578125), path of model used:models/regressor/regressor_model.pt
, 53.24559783935547), path of model used:models/regressor/regressor_model.pt
995), path of model used:models/naive/naive_model.csv
, 58.99810028076172), path of model used:models/regressor/regressor_model.pt
, 71.50562286376953), path of model used:models/regressor/regressor_model.pt
, 65.1242904663086), path of model used:models/regressor/regressor_model.pt
32999999994), path of model used:models/naive/naive_model.csv
 82.81406000000001), path of model used:models/naive/naive_model.csv
 45.93312454223633), path of model used:models/regressor/regressor_model.pt
 81.69925689697266), path of model used:models/regressor/regressor_model.pt
415), path of model used:models/naive/naive_model.csv
 59.32156753540039), path of model used:models/regressor/regressor_model.pt
, 46.686802), path of model used:models/naive/naive_model.csv
164), path of model used:models/naive/naive_model.csv
.77116394042969), path of model used:models/regressor/regressor_model.pt
, 53.24559783935547), path of model used:models/regressor/regressor_model.pt
), path of model used:models/naive/naive_model.csv
, 65.1242904663086), path of model used:models/regressor/regressor_model.pt
 78.12507629394531), path of model used:models/regressor/regressor_model.pt
85.6073226928711), path of model used:models/regressor/regressor_model.pt
, 89.2145004272461), path of model used:models/regressor/regressor_model.pt
, 65.1242904663086), path of model used:models/regressor/regressor_model.pt
 87.83994000000001), path of model used:models/naive/naive_model.csv
, 63.71875), path of model used:models/naive/naive_model.csv
, 71.949522), path of model used:models/naive/naive_model.csv
, 71.949522), path of model used:models/naive/naive_model.csv
415), path of model used:models/naive/naive_model.csv
 82.81406000000001), path of model used:models/naive/naive_model.csv
 64.68891906738281), path of model used:models/regressor/regressor_model.pt
.77116394042969), path of model used:models/regressor/regressor_model.pt
.77116394042969), path of model used:models/regressor/regressor_model.pt
```

Jak wyżej, ale przy podziale TEST_SPLIT_A = 0.1 (10% użytkowników widzi model naiwny):

```
33.037043), path of model used:models/naive/naive_model.csv
31.656749725341797), path of model used:models/regressor/regressor_model.pt
45.93312454223633), path of model used:models/regressor/regressor_model.pt
5), path of model used:models/naive/naive_model.csv
53.24559783935547), path of model used:models/regressor/regressor_model.pt
5.6073226928711), path of model used:models/regressor/regressor_model.pt
80.8154296875), path of model used:models/regressor/regressor_model.pt
70.56119537353516), path of model used:models/regressor/regressor_model.pt
77116394042969), path of model used:models/regressor/regressor_model.pt
5.6073226928711), path of model used:models/regressor/regressor_model.pt
71.949522), path of model used:models/naive/naive_model.csv
45.93312454223633), path of model used:models/regressor/regressor_model.pt
31.656749725341797), path of model used:models/regressor/regressor_model.pt
65.1242904663086), path of model used:models/regressor/regressor_model.pt
59.32156753540039), path of model used:models/regressor/regressor_model.pt
59.27898025512695), path of model used:models/regressor/regressor_model.pt
59.32156753540039), path of model used:models/regressor/regressor_model.pt
65.1242904663086), path of model used:models/regressor/regressor_model.pt
65.1242904663086), path of model used:models/regressor/regressor_model.pt
71.75534057617188), path of model used:models/regressor/regressor_model.pt
59.27898025512695), path of model used:models/regressor/regressor_model.pt
5.6073226928711), path of model used:models/regressor/regressor_model.pt
89.14508819580078), path of model used:models/regressor/regressor_model.pt
59.32156753540039), path of model used:models/regressor/regressor_model.pt
78.12507629394531), path of model used:models/regressor/regressor_model.pt
59.32156753540039), path of model used:models/regressor/regressor_model.pt
9.02374267578125), path of model used:models/regressor/regressor_model.pt
```

Wnioski:

Wszystkie dane, na których bazujemy przewidywania są nominalne (poza dniem tygodnia, który jest porządkowy, ale model i tak osiągał lepszą celność przy jego reprezentacji nominalnej). Z tego powodu uzyskanie znacznie lepszych wyników niż model naiwny może okazać się niemożliwe.

Model na podstawie danych może jedynie zakwalifikować każdy przykład do jednej ze 147 (7 dni tygodnia, 7 miast, 3 firmy kurierskie) kategorii i zwrócić wynik dla tej kategorii, ponieważ nie ma dostępnych innych danych. Danych treningowych jest wystarczająco dużo dla każdej z kategorii, więc zadanie można praktycznie sprowadzić do wyznaczenia okna dla każdej kategorii, które zawiera odpowiednio wiele punktów, ale nie jest za szerokie. Przy mniej więcej stałej szerokości okna, jedyne w czym model uczenia maszynowego może być lepszy od naiwnego, jest przesunięcie okna tak, żeby zawierało więcej punktów niż model naiwny. Ponieważ, z tego co pamiętamy z naszej analizy, punkty te mają rozkład podobny do normalnego, średnia użyta w modelu naiwnym jest bardzo dobrym kryterium wyboru tego przesunięcia, a więc niewiele przewagi da się zyskać w ten sposób.

Model uczenia maszynowego może więc być trochę lepszy dzięki ustaleniu różnych rozmiarów okien dla różnych kategorii, żeby średnio rozmiar okna był mniejszy. Jak jednak widać, nie przynosi to wystarczająco dobrych rezultatów, żeby wyprzedzić model naiwny.

Kryterium biznesowe jest jednak spełnione przez oba modele, a więc można uznać któryś z nich za rozwiązanie danego problemu biznesowego. Model naiwny jest znacznie prostszy i szybszy w trenowaniu i zwracaniu predykcji, ale może przestać być wystarczający, jeśli zadanie się skomplikuje (jeśli dane nominalne przestaną wystarczać), albo nie będzie wystarczająco dużo dostępnych danych do trenowania wszystkich kategorii.