Spis treści

Streszczenie

Niniejszy projekt ma na celu przyglądnięcie się opóźnieniom tramwajów w mieście Kraków. Składa się z dwóch rozdziałów oraz krótkiego podsumowania. Do analizy wykorzystano język programowania Python z uwagi na jego prostotę i bogatą ilość bibliotek oraz Jupyter Notebook – interaktywny notes IPython otwierany w przeglądarce internetowej.

Pierwszy rozdział obejmuje wstęp oraz cel opracowania, gdzie wyjaśnione jest co było inspiracją oraz skąd można pobrać potrzebne dane do analizy. Drugi rozdział jest nieco bardziej obszerny. Na wstępie omówiona jest struktura danych wraz z ich wyjaśnieniem. Z kolei przechodzimy do wyjaśnienia kodu źródłowego w języku Python na podstawie dokładnej analizy danych pobranych w poniedziałek 23-07-2018. Obejmuje to:

- Informację o wykorzystanych bibliotekach;
- Wyjaśnienie sposobu pobierania danych;
- Sprawdzenie średnich opóźnień dla analizowanych linii tramwajowych;
- W jaki sposób opóźnienia rozkładają się w ciągu dnia;
- Które przystanki i linie tramwajowe są najbardziej i najmniej podatne na opóźnienia;
- Prosty model prezentujący ogólny sens stosowania uczenia maszynowego, na podstawie którego można następnie dokonywać predykcji opóźnień.

Następnie dokonano ogólnej analizy opóźnień dla innych dni tygodnia, celem porównania wyników.

Na samym końcu znajduje się krótkie podsumowanie obejmujące wnioski.

Rozdział 1 - Wprowadzenie

1. 1 Wstęp

Tematem opracowania będzie analiza opóźnień krakowskich tramwajów. Inspiracją jest Korona Wyzwań Uczenia Maszynowego Data Workshop [5] oraz artykuł [6], na który natknęłam się na jednej ze stron internetowych. Postanowiłam wykorzystać pobrane przez autora dane [7], by następnie móc je przeanalizować i wyciągnąć wnioski. Nic nie stoi też na przeszkodzie, by dane te pobrać samodzielnie, bowiem od jakiegoś już czasu w przeglądarce możliwe jest śledzenie informacji o czasie przyjazdu danego tramwaju na przystanek, uwzględniając jego opóźnienie. Są to te same dane, które znajdziemy na tablicach na przystankach tramwajowych. Informacje te możemy znaleźć na stronie TTSS (Traffic Tram Supervision System): ttss.krakow.pl. Na podstronie http://www.ttss.krakow.pl/internetservice/ możemy także prześledzić znacznie ciekawsze informacje, gdzie możemy sprawdzić aktualne położenie tramwajów na mapie i ich przewidywane opóźnienie. Jest też możliwość prześledzenia i analizy ruchu autobusowego, pobierając dane ze strony https://mpk.jacekk.net/

1.2 Cel opracowania

Jak wszyscy wiemy, tramwaje nieraz się spóźniają. Raz jest to 1min, a innym razem może to być 20min. Czasem opóźnienia mogą mieć przykre konsekwencje, gdy np. spóźnimy się na ważne spotkanie z klientem. W niniejszej pracy planuję sprawdzić, czy jest jakaś zależność pomiędzy czasem przyjazdu tramwaju, a daną linią, przystankiem, godziną, czy odległością od pętli. Gdy więc wedle rozkładu nasz tramwaj powinien przyjechać o 7:30, ale nasz model mówi, że o tej godzinie średnie opóźnienie wynosi np. 8min, to mimo, że wyszliśmy o 2min za późno z domu, jest duże prawdopodobieństwo, że zdążymy na ten tramwaj.

Rozdział 2 – Analiza danych

2.1 Dane

Wykorzystane dane są dostępne w formacie .csv na stronie internetowej [7]. Dotyczą one miasta Kraków, w tym przypadku ruchu tramwajowego. Obejmują wybrane dni lipca zeszłego roku.

	index	time_stamp	stop	stopName	number	direction	plannedTime	vehicleId	tripId	status	delay	seq_num
0	1	2018-07-23 06:00:45	378	Os.Piastów	21	Kopiec Wandy	2018-07-23 05:59:00	NaN	6351558574044883205	PLANNED	1	1.0
1	1	2018-07-23 06:00:47	612	Borsucza	22	Walcownia	2018-07-23 06:00:00	6.352185e+18	6351558574044899587	STOPPING	0	7.0
2	1	2018-07-23 06:00:48	572	Smolki	11	Czerwone Maki P+R	2018-07-23 06:00:00	6.352185e+18	6351558574044670211	STOPPING	0	10.0
3	1	2018-07-23 06:00:49	319	Jubilat	1	Wzgórza K.	2018-07-23 05:59:00	NaN	6351558574044363010	PLANNED	1	3.0
4	1	2018-07-23 06:00:49	322	Filharmonia	8	Bronowice Małe	2018-07-23 06:01:00	6.352185e+18	6351558574044592386	STOPPING	0	15.0

Tabela 1 Podgląd danych

Wyjaśnienie danych:

index – numery kolejnych rund zapytań serwera (runda obejmuje wszystkie przystanki, trwa 20 sekund)

time_stamp – czas wysłania zapytania do serwera (zaokrąglając do pełnych minut możemy o utożsamiać z rzeczywistym czasem, odjazdu)

stop – numer przystanku

stopName – nazwa przystanku

number – numer tramwaju

direction – kierunek jazdy tramwaju

plannedTime – planowany czas odjazdu

vehicleId – numer pojazdu

tripId – numer podróży

status – status obserwacji (PLANNED - nie śledzony, PREDICTED - oczekiwany, STOPPING - stoi na przystanku, czyli >>> na tablicach)

delay – wyliczone opóźnienie

seq_num – kolejność przystanków na trasie

2.2 Kod źródłowy na przykładzie danych z dnia 23-07-2018

Biblioteki

Do celów analizy wykorzystano następujące biblioteki:

- pandas – pakiet ten zapewnia struktury danych i funkcje wysokiego poziomu, które przyśpieszają pracę z ustrukturyzowanymi danymi, a także danymi w formie tabel; to dzięki niej Python stał się solidnym środowiskiem analitycznym [3];
- NumPy – pakiet ten jest podstawowym narzędziem przeznaczonym do przeprowadzania obliczeń numerycznych w Pythonie; biblioteka ta obsługuje struktury danych, algorytmy i mechanizmy spajające niezbędne w większości zastosowań naukowych związanych z przeznaczeniem danych numerycznych [3];
- Matplotlib – pakiet ten to najpopularniejsza biblioteka Pythona przeznaczona do tworzenia wykresów i innych dwuwymiarowych wizualizacji danych, biblioteka ta jest stworzona z myślą o tworzeniu wykresów nadających się do publikacji;
- Seaborn – Seaborn jest „nakładką" na matplotlib, z założenia ma umożliwić budowanie ładnych wykresów w nieco prostszy sposób [8];
- Scikit-learn – pakiet ten jest obecnie uważany przez programistów Pythona za najważniejszy zestaw narzędzi uczenia maszynowego; zawiera moduły obsługujące między innymi modele: klasyfikacja, regresja, analiza skupień, redukcja liczby wymiarów, selekcja modelu, wstępna obróbka danych [3].

Z pakietu Scikit-learn importujemy *DecisionTreeRegressor*, gdyż jest to problem regresji. W tym przypadku wykonujemy prognozowanie opóźnień, które będą podane w minutach. Oczywiście większość opóźnień to 1, 2, 3 minuty i moglibyśmy to potraktować jako klasyfikację, ale w przypadku, gdy będziemy mieć opóźnienie wynoszące 15 minut to będziemy mieć 15 klas. Jest to znaczna liczba i w tym przypadku zarządzanie za pomocą klasyfikacji nie będzie najlepszym podejściem. Reasumując, wybieramy regresję dlatego, że wartość, którą prognozujemy jest wartością ciągłą. Przyda się także *cross_val_score*, czyli walidacja krzyżowa, która zostanie omówiona później.

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

from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import cross_val_score

%matplotlib inline
```

Dane

Poniższa analiza obejmuje dane pobrane w dniu 23-07-2018 (poniedziałek).

Na początku wczytujemy nasze dane, gdzie wklejamy adres URL do naszych danych. Z kolei za pomocą polecenia *head* wczytujemy 5 pierwszych wierszy, by przyglądnąć zorientować się, jak wyglądają nasze dane w formie tabelarycznej.

```
df23=pd.read_csv('https://raw.githubusercontent.com/aczepielik
/KRKtram/master/reports/report_07-23.csv')
df.head()
```

	index	time_stamp	stop	stopName	number	direction	plannedTime	vehicleId	tripId	status	delay	seq_num
0	1	2018-07-23 06:00:45	378	Os.Piastów	21	Kopiec Wandy	2018-07-23 05:59:00	NaN	6351558574044883205	PLANNED	1	1.0
1	1	2018-07-23 06:00:47	612	Borsucza	22	Walcownia	2018-07-23 06:00:00	6.352185e+18	6351558574044899587	STOPPING	0	7.0
2	1	2018-07-23 06:00:48	572	Smolki	11	Czerwone Maki P+R	2018-07-23 06:00:00	6.352185e+18	6351558574044670211	STOPPING	0	10.0
3	1	2018-07-23 06:00:49	319	Jubilat	1	Wzgórza K.	2018-07-23 05:59:00	NaN	6351558574044363010	PLANNED	1	3.0
4	1	2018-07-23 06:00:49	322	Filharmonia	8	Bronowice Małe	2018-07-23 06:01:00	6.352185e+18	6351558574044592386	STOPPING	0	15.0

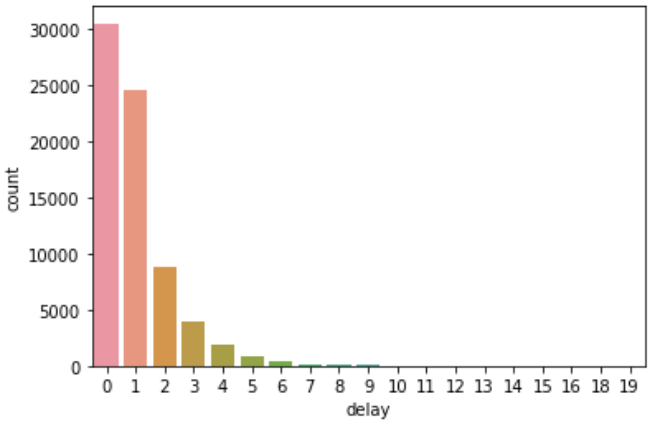
Tabela 2 Podgląd pięciu pierwszych wersów danych

Rozkład spóźnień

Możemy sprawdzić jakiego rzędu są opóźnienia oraz jakie występują najczęściej:

```
sns.countplot(x='delay', data=df23)
df23.delay.value_counts()
```

```
0      30531
1      24653
2       8833
3       4004
4       1818
5        816
6        347
7        190
8         85
9         62
10        42
14        36
12        34
11        25
13        24
15        11
16         3
19         2
18         1
```



Rys. 1 Wykres pokazujący ilość pojazdów w zależności od opóźnienia w [min]

Czasem dobrze jest zobaczyć ilość tramwajów, które nie pojawiły się na czas, ale czasem ciężko jest oszacować, czy ilość pojazdów, które przyjechały na czas, czyli 30531 to jest dużo, czy mało. Z pomocą przyjdzie nam normalizacja:

```
df23.delay.value_counts(normalize = True)
```

```
0     0.426905
1     0.344715
2     0.123509
3     0.055987
4     0.025421
5     0.011410
6     0.004852
7     0.002657
8     0.001189
9     0.000867
10    0.000587
14    0.000503
12    0.000475
11    0.000350
13    0.000336
15    0.000154
16    0.000042
19    0.000028
18    0.000014
```

```
df23.delay.describe()
```

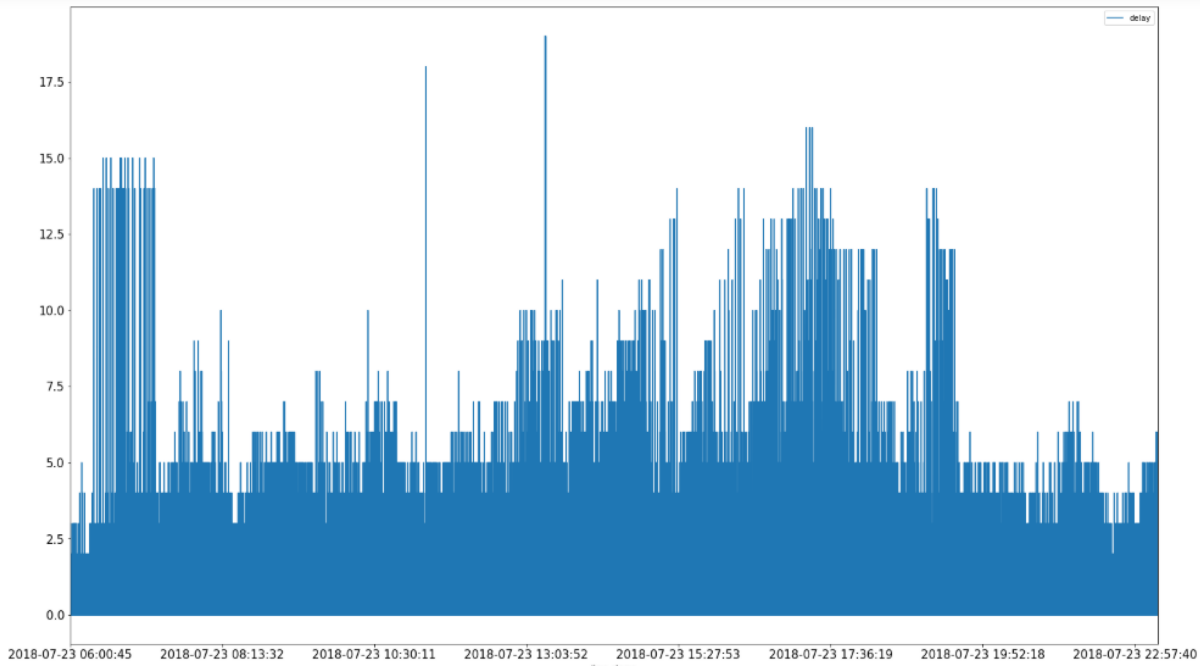
```
count    71517.000000
mean         1.014039
std          1.357324
min          0.000000
25%          0.000000
50%          1.000000
75%          1.000000
max         19.000000
```

Z punktu widzenia pesymisty można wywnioskować, że ponad połowa tramwajów przyjeżdża opóźniona. Jednak patrząc na to z drugiej strony, zdecydowana większość tramwajów spóźnia się niewiele – do 2 minut. Średnie spóźnienie w próbie wyniosło w przybliżeniu 1 minutę 1 sekundę (1.01 minuty), a odchylenie standardowe 1 minutę i 21 sekund (1.35 minuty).

Spóźnienia w czasie

Zobaczmy jak wygląda rozkład opóźnień w ciągu dnia w zależności od godziny:

```
df23.plot(x='time_stamp',        y='delay',        kind='line',
figsize=(25,15), fontsize=15)
```



Rys. 2 Wykres obrazujący opóźnienie pojazdu w [min] w zależności od pory dnia

Największe spóźnienia możemy zaobserwować w godzinach szczytu (dojazdy do pracy), najmniejsze w godzinach wczesno porannych, przedpołudniowych oraz wieczornych. Między 6:30-7:00 rano mogą one sięgać nawet do 15 minut, następnie gwałtownie spadają do 4-5min. Od godz. 13:00 do 16:30 zaczynają powoli wzrastać, co może oznaczać, że na przystanku będziemy musieć zaczekać nawet dodatkowe 10min. We wspomnianych godzinach największe opóźnienia zdarzają się o pełnej godzinie, co może się wiązać z końcem dnia pracy. Od 16:30 do 18:00 możemy zaobserwować popołudniowe godziny szczytu wiążące się opóźnieniami sięgającymi 15min oraz podobne opóźnienia w godz. 19:00-19:30 (np. wieczorne spotkania w gronie znajomych). Po godz. 19:30 opóźnienia są już niewielkie.

Ranking spóźnień na przystankach

Sprawdźmy teraz na jakich przystankach można spodziewać się największych, a na jakich najmniejszych opóźnień.

```
stopMeanDelay = df23.groupby('stopName').delay.mean().
reset_index(name='stopMeanDelay')
stopMeanDelay.sort_values(by='stopMeanDelay',ascending=False).
head(10)
stopMeanDelay.sort_values(by='stopMeanDelay',ascending=True).h
ead(10)
```

Przystanki z największym średnim opóźnieniem:

	stopName	stopMeanDelay
149	Łagiewniki ZUS	2.076087
94	Plaza	1.950276
78	Ofiar Dąbia	1.920110
131	Teatr Variété	1.779614
40	Francesco Nullo	1.760989
36	Dąbie	1.756906
38	Fabryczna	1.730769
127	TAURON Arena Kraków Al. Pokoju	1.704420
44	Hala Targowa	1.595568
116	Smolki	1.588362

Tabela 3 Średnie największe opóźnienia danego przystanku [min]

Przystanki z najmniejszym średnim opóźnieniem:

	stopName	stopMeanDelay
66	Mały Płaszów	0.096296
24	Czerwone Maki P+R	0.100000
19	Cichy Kącik	0.119565
8	Borek Fałęcki	0.148148
13	Bronowice Małe	0.158301
58	Krowodrza Górka	0.169014
22	Cmentarz Rakowicki	0.180000
144	Wzgórza Krzesławickie	0.187879
137	Walcownia	0.204545
48	Kampus UJ	0.273663

Tabela 4 Średnie najmniejsze opóźnienia danego przystanku [min]

Z powyższych zestawień możemy wywnioskować, że im bliżej pętli znajduje się przystanek, tym opóźnienie jest mniejsze, przy czym najmniejsze jest na przystankach początkowych, gdy tramwaj dopiero wyrusza z pętli. Największe opóźnienia możemy zaobserwować na przystankach w pobliżu miejsc, gdzie ruch tramwajowy krzyżuje się z ruchem samochodowym (np. w pobliżu rond, gdzie tramwaje niejednokrotnie muszą zatrzymać się na czerwonym świetle).

Ranking spóźnień dla danych linii

Sprawdźmy teraz, które z linii tramwajowych są najbardziej podatne na opóźnienia.

```
lineMeanDelay = df23.groupby(['number', 'direction']).delay.mean().reset_index(name='lineMeanDelay')
lineMeanDelay.sort_values(by='lineMeanDelay',ascending=False).head(10)
lineMeanDelay.sort_values(by='lineMeanDelay',ascending=True).head(10)
```

Linie z największym średnim opóźnieniem:

	number	direction	lineMeanDelay
40	22	Walcownia	2.109223
24	14	Bronowice Małe	1.762376
41	24	Bronowice Małe	1.649254
31	19	Borek Fałęcki	1.602434
38	22	Borek Fałęcki	1.567422
25	14	Mistrzejowice	1.515894
18	10	Kopiec Wandy	1.481264
13	6	Salwator	1.340852
39	22	Kombinat	1.316129
21	11	Mały Płaszów	1.232456

Tabela 5 Średnie największe opóźnienia danej linii [min]

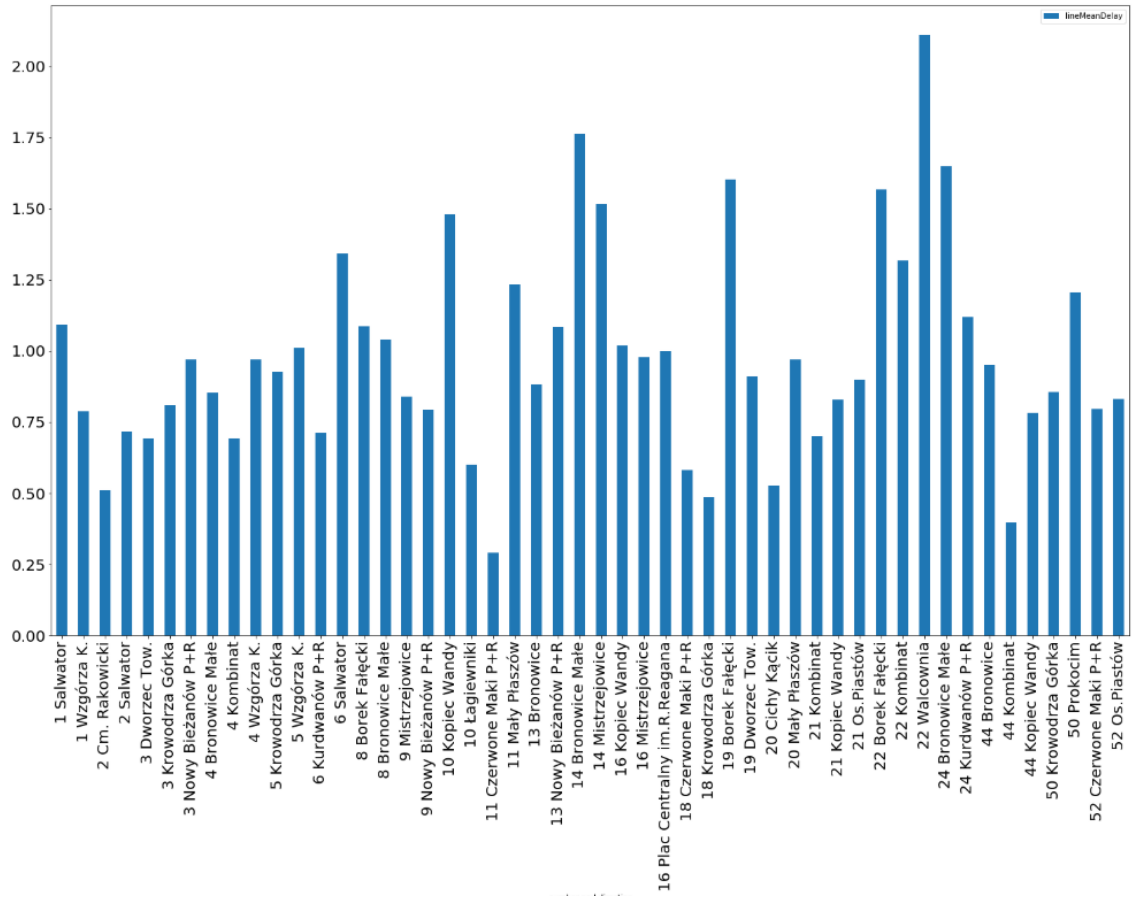
Linie z najmniejszym średnim opóźnieniem:

	number	direction	lineMeanDelay
20	11	Czerwone Maki P+R	0.289753
44	44	Kombinat	0.395349
30	18	Krowodrza Górka	0.485581
2	2	Cm. Rakowicki	0.512097
33	20	Cichy Kącik	0.526414
29	18	Czerwone Maki P+R	0.580252
19	10	Łagiewniki	0.601317
4	3	Dworzec Tow.	0.691667
8	4	Kombinat	0.692308
35	21	Kombinat	0.700000

Tabela 6 Średnie najmniejsze opóźnienia danej linii [min]

Poniżej przedstawione są powyższe dane na wykresie:

```
lineMeanDelay['number and direction'] =
lineMeanDelay.agg('{0[number]} {0[direction]}'.format, axis=1)
lineMeanDelay.plot(x='number and direction', y='lineMeanDelay',
kind='bar', figsize=(25,15), fontsize=20)
```



Rys. 3 Wykres przedstawiający średnie opóźnienie w [min] dla danej linii i kierunku jazdy

O ile najkrótsze linie mają tendencję do niewielkiego średniego opóźnienia, to nie widać ogólnej, silnej zależności.

Model uczenia maszynowego

Poniżej zbudowano prosty model prezentujący ogólny sens stosowania uczenia maszynowego, na podstawie którego można następnie dokonywać predykcji opóźnień. Na potrzeby modelu zostało stworzonych siedem kombinacji. Trenując model otrzymano wielkość błędu.

Wartość *plannedTime* jest stringiem, niezbędne staje się więc jego przekonwertowanie do formatu daty.

```
df['plannedTime'] = pd.to_datetime (df['plannedTime'])
df[['plannedTime']].info()
```

```
df['hour'] = df['plannedTime'].dt.hour.value_counts()
```

Aby uniknąć analizy naszego opóźnienia na którymś miejscu po przecinku, zamieniamy minuty na sekundy:

```
df['delay_secs'] = df['delay'].map(lambda x: x*60)
```

Kierunek jazdy tramwajów (direction), jest stringiem. Rzutujemy więc wartości tekstowe na wartości numeryczne. Robimy to przypisując dla każdej wartości (kierunku) unikalną wartość numeryczną, można to traktować jako ID

```
df['direction_cat'] = df['direction'].factorize()[0]
```

Jak możemy się domyślić, usuwanie danych to bardzo słaba strategia. Aby więc zapobiec wysypaniu się modelu z powodu niektórych pustych wartości w naszym zbiorze i jednocześnie nie pozbywając się tych danych, możemy przypisać im jakąś wartość. Dlaczego akurat -1? Ważne, żeby ta wartość była unikalna i się nie powtarzała, bo jeśli nie, możemy przypadkowo nadpisać jakąś już istniejącą wartość.

```
df['vehicleId'].fillna(-1, inplace = True)
df['seq_num'].fillna(-1, inplace = True)
```

Możemy także połączyć dwie zmienne, np. numer tramwaju i kierunek albo przystanek i kierunek, w jakim ten tramwaj jedzie. Możemy do tego celu użyć funkcji apply.

```
def gen_id_num_direction(x):
    return '{} {}'.format(x['number'], x['direction'])
df['number_direction_id']  =  df.apply(gen_id_num_direction,
axis = 1).factorize()[0]

def gen_id_stop_direction(x):
    return '{} {}'.format(x['stop'], x['direction'])
df['stop_direction_id']  =  df.apply(gen_id_stop_direction,
axis = 1).factorize()[0]
```

Jeśli chodzi o zmienną X, będą to wartości, które wpływają na opóźnienie, jest to lista. y natomiast będzie wektorem, gdyż zawiera on tylko jedną wartość, związaną z opóźnieniem. Obliczenia zostały wykonane dla różnych kombinacji zmiennych celem sprawdzenia, która z nich będzie najbardziej optymalna.

```
feats1 = [
    'number'
]
X1 = df23[ feats1 ].values
```

```
feats2 = [
    'number',
    'stop'
]
X2 = df23[ feats2 ].values

feats3 = [
    'number',
    'stop',
    'direction_cat'
]
X3 = df23[ feats3 ].values

feats4 = [
    'number',
    'stop',
    'direction_cat',
    'vehicleId'
]
X4 = df23[ feats4 ].values

feats5 = [
    'number',
    'stop',
    'direction_cat',
    'vehicleId',
    'seq_num'
]
X5 = df23[ feats5 ].values

feats6 = [
    'number',
    'stop',
    'direction_cat',
    'vehicleId',
    'seq_num',
    'number_direction_id'
]
```

```
X6 = df23[ feats6 ].values

feats7 = [
    'number',
    'stop',
    'direction_cat',
    'vehicleId',
    'seq_num',
    'number_direction_id',
    'stop_direction_id'
]
X7 = df23[ feats7 ].values

y = df23['delay_secs'].values
```

Następnie budujemy model. Wykorzystujemy w tym przypadku walidacji krzyżowej (k-fold cross validation), która umożliwia wykorzystanie całego zbioru danych zarówno do uczenia, jak i do walidacji modelu. Zbiór uczący dzielimy na k równolicznych podzbiorów, w tym przypadku 5, z których k-1 jest wykorzystywanych do uczenia modelu, natomiast 1 podzbiór służy do walidacji modelu.

```
model = DecisionTreeRegressor(max_depth=10, random_state=0)
scores1 = cross_val_score(model, X1, y, cv=5,
scoring='neg_mean_absolute_error')
scores2 = cross_val_score(model, X2, y, cv=5,
scoring='neg_mean_absolute_error')
scores3 = cross_val_score(model, X3, y, cv=5,
scoring='neg_mean_absolute_error')
scores4 = cross_val_score(model, X4, y, cv=5,
scoring='neg_mean_absolute_error')
scores5 = cross_val_score(model, X5, y, cv=5,
scoring='neg_mean_absolute_error')
scores6 = cross_val_score(model, X6, y, cv=5,
scoring='neg_mean_absolute_error')
scores7 = cross_val_score(model, X7, y, cv=5,
scoring='neg_mean_absolute_error')
```

```
Data = [(abs(np.mean(scores1))),
        (abs(np.mean(scores2))),
        (abs(np.mean(scores3))),
        (abs(np.mean(scores4))),
        (abs(np.mean(scores5))),
        (abs(np.mean(scores6))),
        (abs(np.mean(scores7)))]

df23 = pd.DataFrame(Data, index=['feats1','feats2','feats3',
'feats4','feats5','feats6','feats7'], columns=['np.mean'])
```

Zobaczmy jak przedstawiają się wyniki dla poszczególnych kombinacji. Im wynik jest bliżej zera, tym nasz model jest dokładniejszy.

	np.mean
feats1	54.362443
feats2	52.573274
feats3	50.683268
feats4	49.895306
feats5	48.111726
feats6	48.218893
feats7	48.227872

Tabela 7 Średni błąd dla poszczególnych kombinacji w [s]

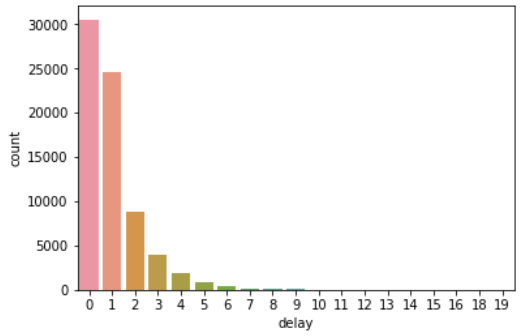
```
minVal23 = df23.min()

print('Minimum value is: ')
print(minVal23)


minValInd23 = df23.idxmin()

print("Min value is at row index position:")
print(minValInd23)

Minimum value is:
np.mean    48.111726
Min value is at row index position:
np.mean    feats5
```

Najbardziej korzystną opcją okazała się kombinacja:

- numer tramwaju
- numer przystanku
- kierunek jazdy tramwaju
- numer pojazdu
- kolejność przystanku na trasie

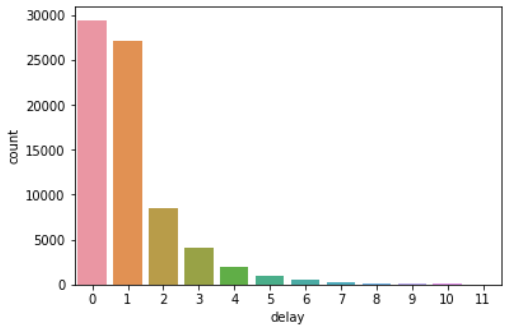
2.3 Analiza opóźnień na przykładzie wybranych dni tygodnia

Wykresy pokazujące ilości pojazdów w zależności od opóźnienia w [s] dla danego dnia:



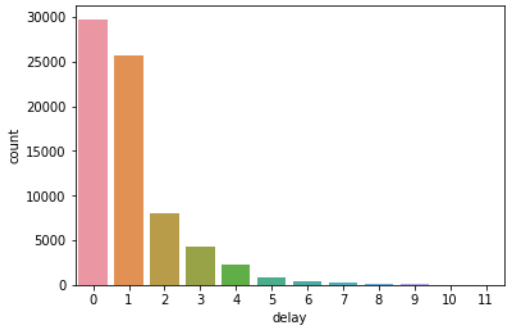
Rys. 4 Poniedziałek 23-07-2018

```
0     0.426905
1     0.344715
2     0.123509
3     0.055987
4     0.025421
5     0.011410
6     0.004852
7     0.002657
8     0.001189
9     0.000867
10    0.000587
14    0.000503
12    0.000475
11    0.000350
13    0.000336
15    0.000154
16    0.000042
19    0.000028
18    0.000014

count    71517.000000
mean         1.014039
std          1.357324
min          0.000000
25%          0.000000
50%          1.000000
75%          1.000000
max         19.000000
```



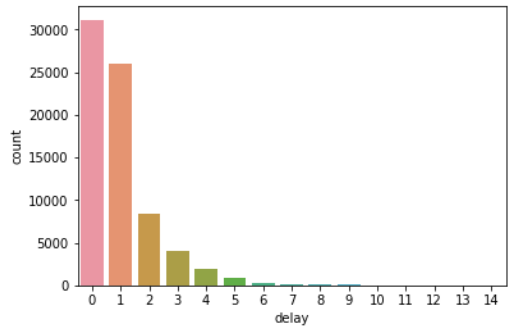
Rys. 5 Wtorek 24-07-2018

```
0     0.402407
1     0.370208
2     0.115680
3     0.056023
4     0.026611
5     0.013729
6     0.006612
7     0.004098
8     0.001762
9     0.001284
10    0.001120
11    0.000464

count    73202.000000
mean         1.055053
std          1.347920
min          0.000000
25%          0.000000
50%          1.000000
75%          1.000000
max         11.000000
```



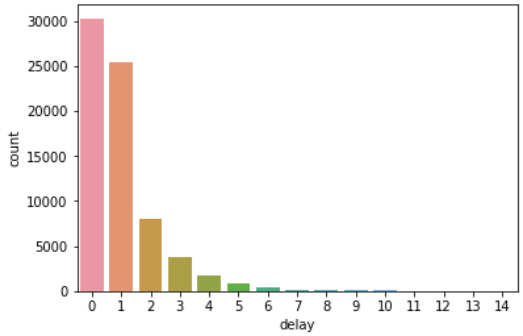
Rys. 6 Środa 25-07-2018

```
0     0.414098
1     0.358323
2     0.111814
3     0.059920
4     0.030906
5     0.011767
6     0.005925
7     0.003352
8     0.002100
9     0.001321
10    0.000389
11    0.000083

count    71896.000000
mean         1.036692
std          1.314104
min          0.000000
25%          0.000000
50%          1.000000
75%          1.000000
max         11.000000
```



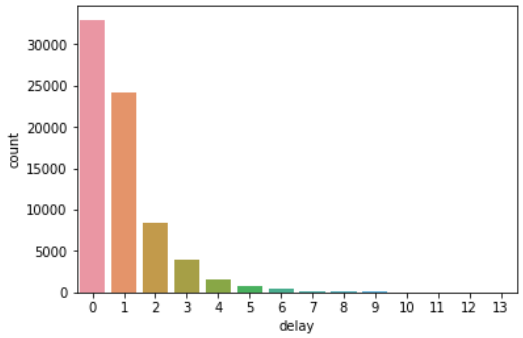
Rys. 7 Czwartek 26-07-2018

```
0     0.425986
1     0.355036
2     0.114351
3     0.055110
4     0.026810
5     0.012406
6     0.004651
7     0.002271
9     0.000875
8     0.000862
10    0.000588
11    0.000479
13    0.000328
12    0.000164
14    0.000082

count    73108.000000
mean         0.995445
std          1.293580
min          0.000000
25%          0.000000
50%          1.000000
75%          1.000000
max         14.000000
```



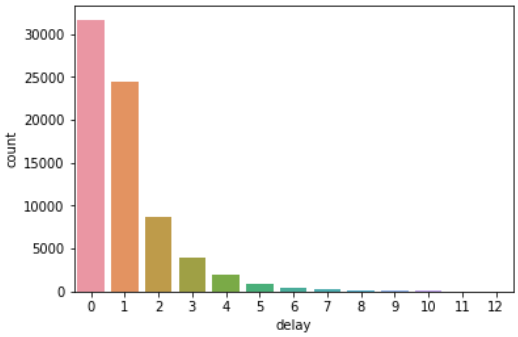
Rys. 8 Piątek 27-07-2018

```
0     0.428440
1     0.358777
2     0.112486
3     0.052350
4     0.024221
5     0.011594
6     0.005195
7     0.002619
8     0.001897
9     0.000821
10    0.000807
11    0.000382
12    0.000212
13    0.000156
14    0.000042

count    70640.000000
mean         0.985164
std          1.296983
min          0.000000
25%          0.000000
50%          1.000000
75%          1.000000
max         14.000000
```



Rys. 9 Poniedziałek 30-07-2018

```
0     0.453183
1     0.333219
2     0.115072
3     0.054980
4     0.022495
5     0.010911
6     0.005126
7     0.002528
8     0.001003
9     0.000660
10    0.000453
11    0.000151
13    0.000110
12    0.000110

count    72772.000000
mean         0.944196
std          1.246442
min          0.000000
25%          0.000000
50%          1.000000
75%          1.000000
max         13.000000
```

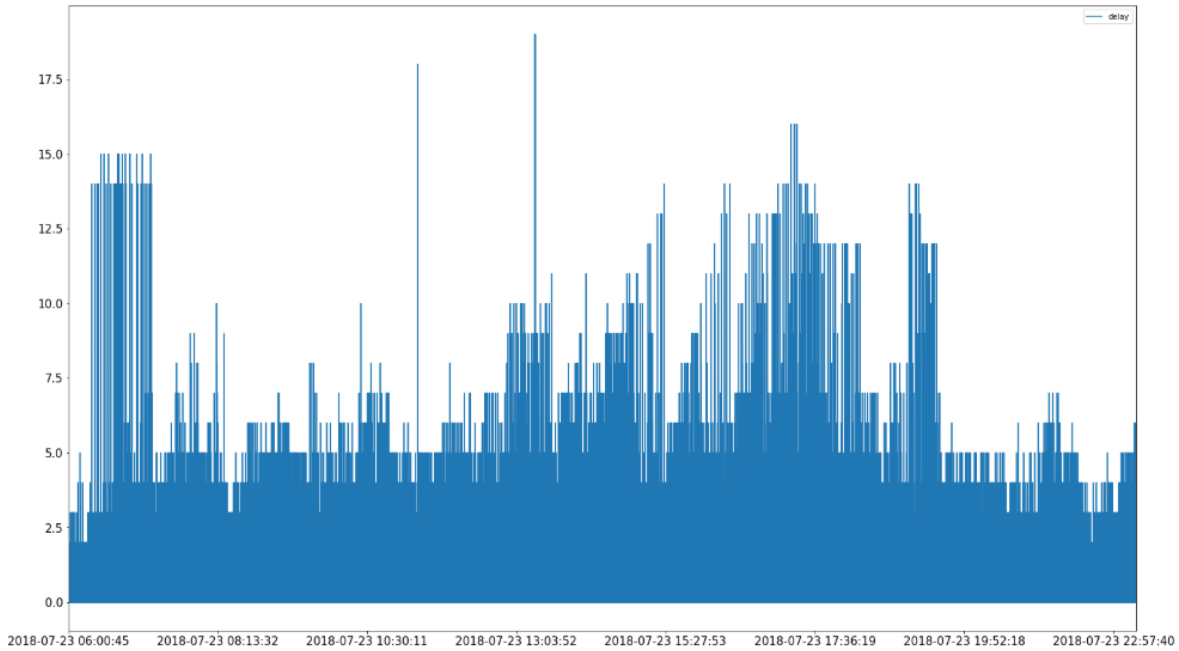


Rys. 10 Wtorek 31-07-2018

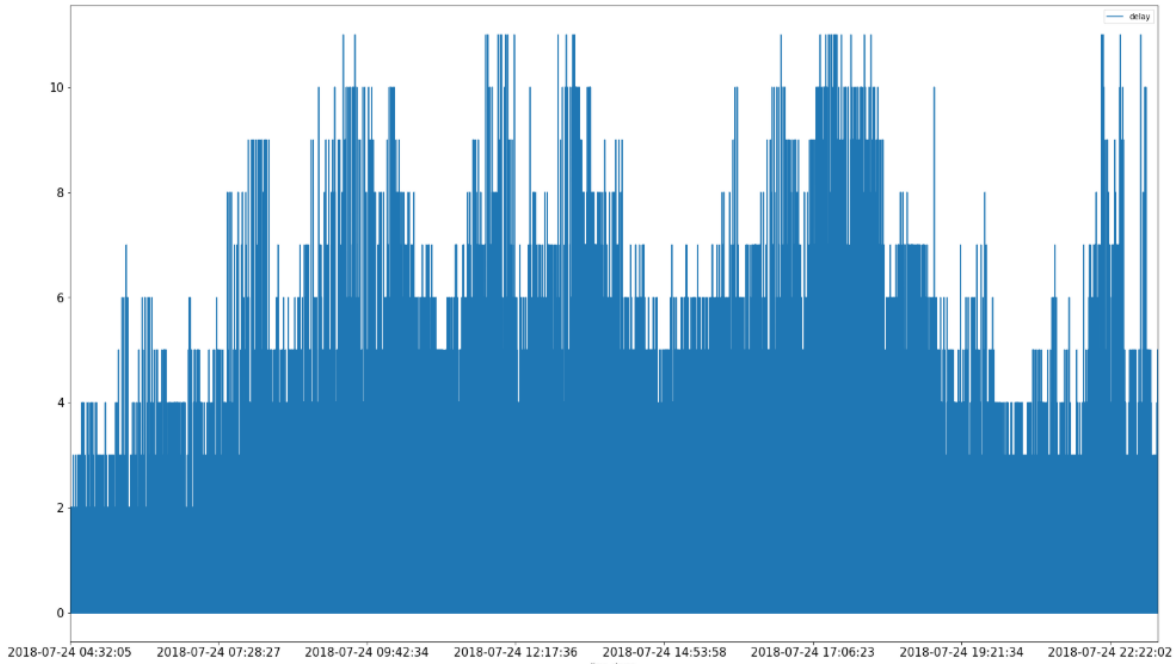
```
0     0.437701
1     0.336924
2     0.120455
3     0.053661
4     0.026099
5     0.011876
6     0.005869
7     0.003245
8     0.002071
9     0.001091
10    0.000690
11    0.000290
12    0.000028

count    72417.000000
mean         0.997335
std          1.311809
min          0.000000
25%          0.000000
50%          1.000000
75%          1.000000
max         12.000000
```

Wniosek: Rezultaty dla wszystkich przypadków są podobne. Ponad połowa tramwajów przyjeżdża opóźniona, jednak zdecydowana większość tramwajów spóźnia się niewiele – do 2 minut. Średnie spóźnienie wynosi w przybliżeniu 1 minutę.

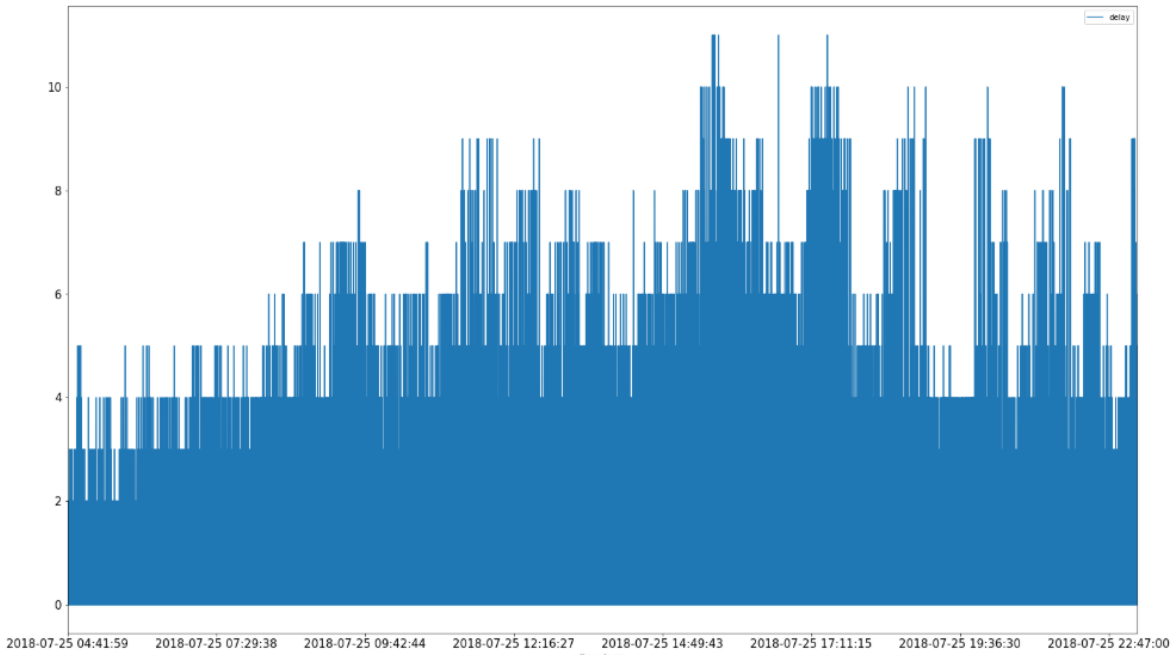
Wykresy obrazujące rozkład opóźnień pojazdu w [min] w zależności od pory dnia:



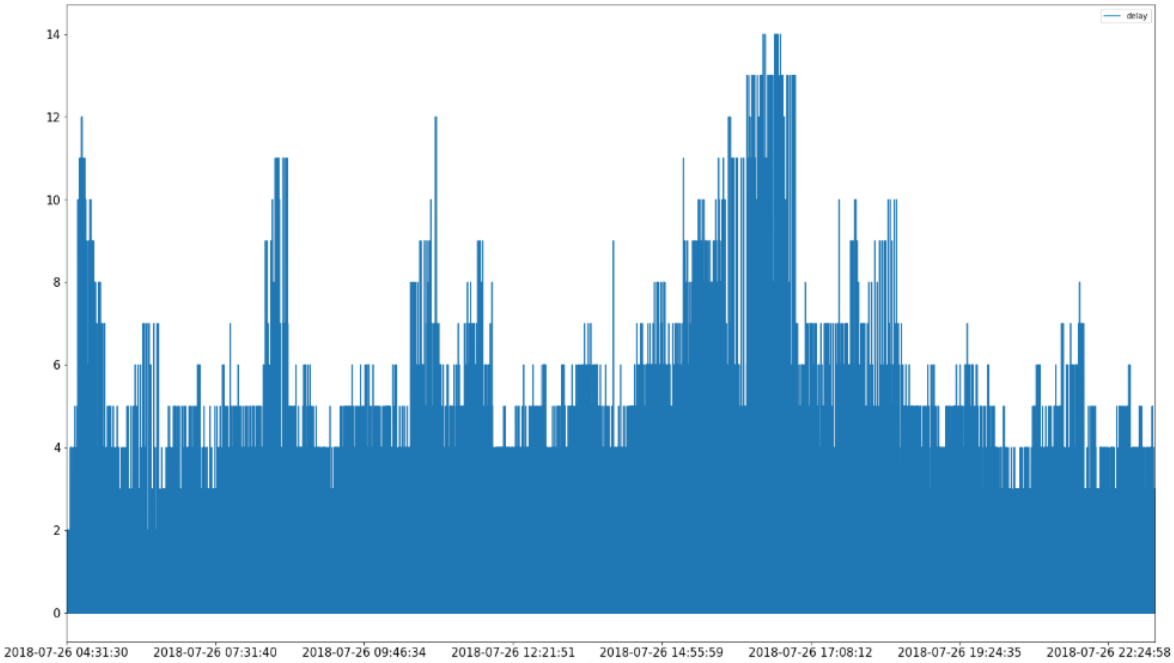
Rys. 11 Poniedziałek 23-07-2018



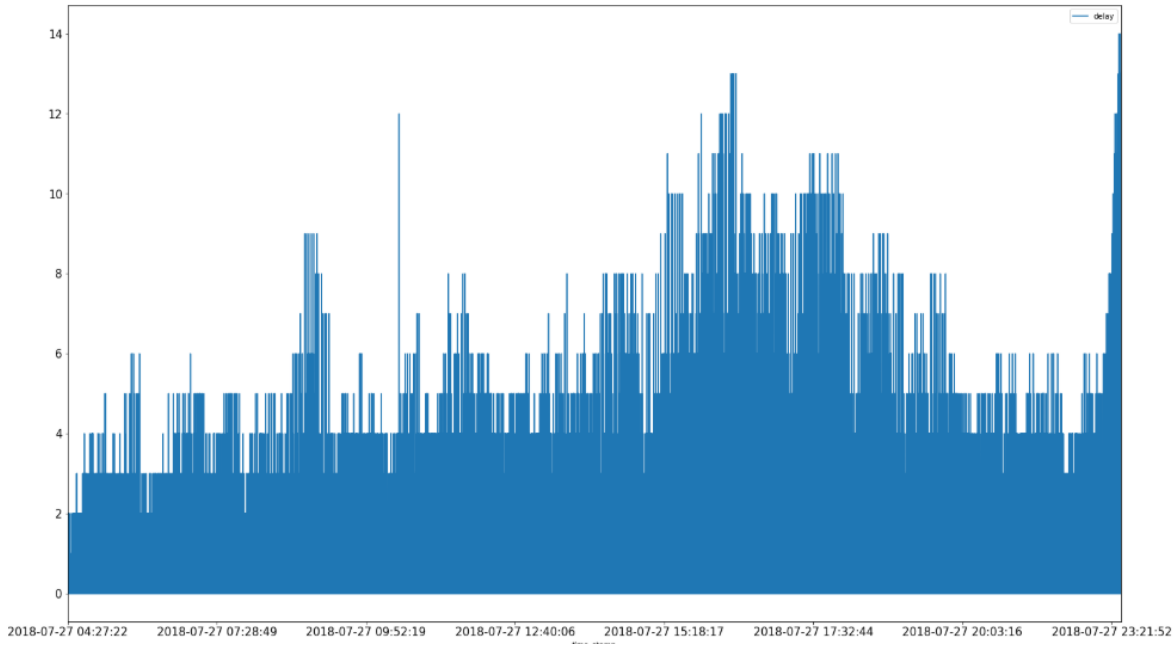
Rys. 12 Wtorek 24-07-2018



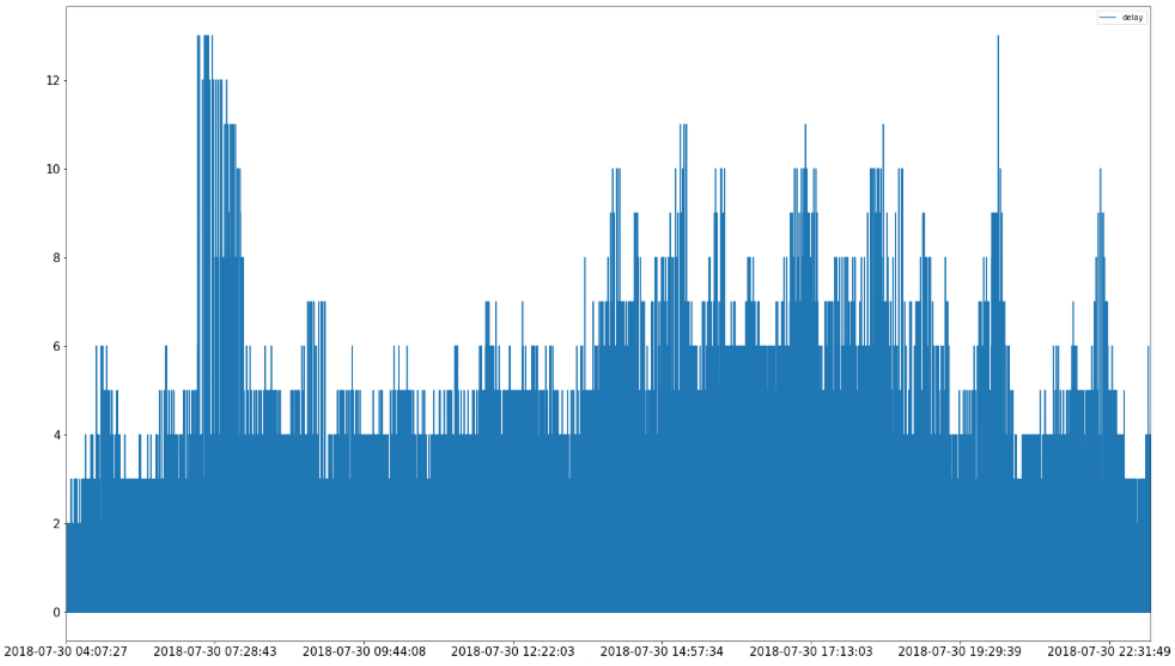
Rys. 13 Środa 25-07-2018



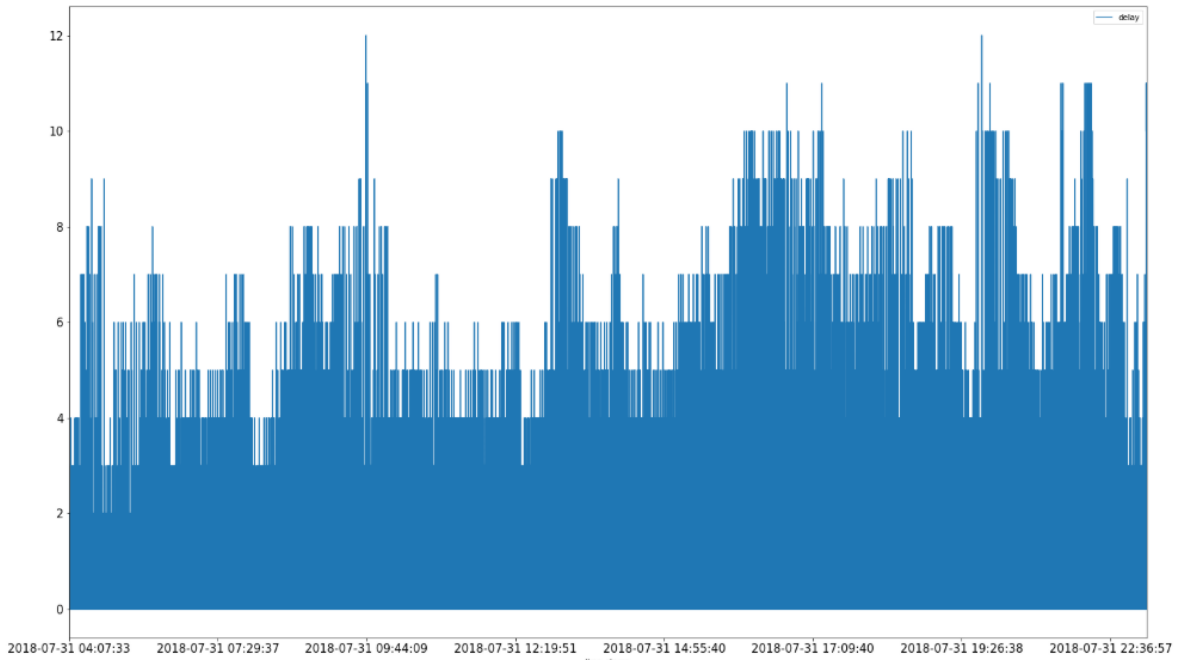
Rys. 14 Czwartek 26-07-2018



Rys. 15 Piątek 27-07-2018



Rys. 16 Poniedziałek 30-07-2018



Rys. 17 Wtorek 31-07-2018

Wniosek: Analizując rozkład opóźnień w ciągu dnia w poprzednim rozdziale (poniedziałek 23-07-2019), spodziewałam się uzyskać podobne wykresy dla pozostałych dni. Kierując się intuicją, szczyt opóźnień powinniśmy zaobserwować w godzinach porannych i popołudniowych – w przypadku poszczególnych dni możemy zaobserwować tendencję do w miarę równomiernego rozkładu opóźnień w ciągu dnia, np. wtorek 24-07, środa 25-07, czy wtorek 31-07. Warto jednak wziąć pod uwagę fakt, że pomiary dokonywane są w środku wakacji, co może wykazywać pewne zaburzenia związane z tym, że część osób ma urlop, rok akademicki się nie rozpoczął, a dzieci nie uczęszczają do szkół, bądź przedszkoli.

Tabele pokazujące średnie największe i najmniejsze opóźnienie w [min] dla danego przystanku w danym dniu:

	stopName	stopMeanDelay
149	Łagiewniki ZUS	2.076087
94	Plaza	1.950276
78	Ofiar Dąbia	1.920110
131	Teatr Variété	1.779614
40	Francesco Nullo	1.760989
36	Dąbie	1.756906
38	Fabryczna	1.730769
127	TAURON Arena Kraków Al. Pokoju	1.704420
44	Hala Targowa	1.595568
116	Smolki	1.588362

Tabela 8 Poniedziałek 23-07-2018 - średnie największe opóźnienia danego przystanku

	stopName	stopMeanDelay
66	Mały Płaszów	0.096296
24	Czerwone Maki P+R	0.100000
19	Cichy Kącik	0.119565
8	Borek Fałęcki	0.148148
13	Bronowice Małe	0.158301
58	Krowodrza Górka	0.169014
22	Cmentarz Rakowicki	0.180000
144	Wzgórza Krzesławickie	0.187879
137	Walcownia	0.204545
48	Kampus UJ	0.273663

Tabela 9 Poniedziałek 23-07-2018 - średnie najmniejsze opóźnienia danego przystanku

	stopName	stopMeanDelay
150	Łagiewniki ZUS	2.198925
75	Nowosądecka	2.091667
88	Piaski Nowe	2.008310
29	Dauna	1.964088
31	Dworcowa	1.793187
22	Cmentarz Podgórski	1.788063
48	Kabel	1.787091
152	Św.Wawrzyńca	1.771654
117	Smolki	1.753623
90	Plac Bohaterów Getta	1.696252

Tabela 10 Wtorek 24-07-2018 - średnie największe opóźnienia danego przystanku

	stopName	stopMeanDelay
23	Cmentarz Rakowicki	0.127273
20	Cichy Kącik	0.135417
145	Wzgórza Krzesławickie	0.143646
59	Krowodrza Górka	0.157480
25	Czerwone Maki P+R	0.227273
102	Rakowicka	0.245455
67	Mały Płaszów	0.261194
84	Os.Piastów	0.297872
76	Nowy Bieżanów P+R	0.311111
9	Borek Fałęcki	0.325301

Tabela 11 Wtorek 24-07-2018 - średnie najmniejsze opóźnienia danego przystanku

	stopName	stopMeanDelay
150	Łagiewniki ZUS	2.015789
75	Nowosądecka	1.877966
88	Piaski Nowe	1.831650
29	Dauna	1.744108
48	Kabel	1.726327
146	Zabłocie	1.644841
31	Dworcowa	1.637908
22	Cmentarz Podgórski	1.596330
79	Ofiar Dąbia	1.592593
95	Plaza	1.581579

Tabela 12 Środa 25-07-2018 - średnie największe opóźnienia danego przystanku

	stopName	stopMeanDelay
23	Cmentarz Rakowicki	0.018182
59	Krowodrza Górka	0.076433
20	Cichy Kącik	0.091837
145	Wzgórza Krzesławickie	0.111732
25	Czerwone Maki P+R	0.151639
67	Mały Płaszów	0.211382
102	Rakowicka	0.263636
14	Bronowice Małe	0.267658
9	Borek Fałęcki	0.273743
138	Walcownia	0.292683

Tabela 13 Środa 25-07-2018 - średnie najmniejsze opóźnienia danego przystanku

	stopName	stopMeanDelay
150	Łagiewniki ZUS	2.164021
95	Plaza	1.620690
79	Ofiar Dąbia	1.570292
75	Nowosądecka	1.564470
45	Hala Targowa	1.500000
1	Agencja Kraków Wschód	1.495050
48	Kabel	1.477799
88	Piaski Nowe	1.466851
71	Mrozowa	1.460000
72	Muzeum Lotnictwa	1.459064

Tabela 14 Czwartek 26-07-2018 - średnie największe opóźnienia danego przystanku

	stopName	stopMeanDelay
23	Cmentarz Rakowicki	0.037037
25	Czerwone Maki P+R	0.102881
14	Bronowice Małe	0.151163
20	Cichy Kącik	0.153061
67	Mały Płaszów	0.165468
145	Wzgórza Krzesławickie	0.171429
59	Krowodrza Górka	0.196429
9	Borek Fałęcki	0.228571
138	Walcownia	0.326531
76	Nowy Bieżanów P+R	0.330049

Tabela 15 Czwartek 26-07-2018 - średnie najmniejsze opóźnienia danego przystanku

	stopName	stopMeanDelay
75	Nowosądecka	1.955882
150	Łagiewniki ZUS	1.891429
88	Piaski Nowe	1.819767
29	Dauna	1.767045
48	Kabel	1.688541
31	Dworcowa	1.645963
22	Cmentarz Podgórski	1.591022
143	Witosa	1.589342
152	Św.Wawrzyńca	1.552083
95	Plaza	1.514905

Tabela 16 Piątek 27-07-2018 - średnie największe opóźnienia danego przystanku

	stopName	stopMeanDelay
23	Cmentarz Rakowicki	0.019048
20	Cichy Kącik	0.041667
67	Mały Płaszów	0.115108
14	Bronowice Małe	0.116105
25	Czerwone Maki P+R	0.126531
102	Rakowicka	0.132075
134	Uniwersytet Ekonomiczny	0.150943
59	Krowodrza Górka	0.154321
9	Borek Fałęcki	0.179641
145	Wzgórza Krzesławickie	0.204545

Tabela 17 Piątek 27-07-2018 - średnie najmniejsze opóźnienia danego przystanku

	stopName	stopMeanDelay
151	Łagiewniki ZUS	1.647059
48	Kabel	1.629738
147	Zabłocie	1.618395
75	Nowosądecka	1.600000
89	Piaski Nowe	1.545961
72	Muzeum Lotnictwa	1.510981
53	Klimeckiego	1.499022
31	Dworcowa	1.484185
42	Gromadzka	1.475728
29	Dauna	1.458564

Tabela 18 Poniedziałek 30-07-2018 - średnie największe opóźnienia danego przystanku

	stopName	stopMeanDelay
87	PH	0.000000
20	Cichy Kącik	0.010101
23	Cmentarz Rakowicki	0.046296
14	Bronowice Małe	0.108209
67	Mały Płaszów	0.110345
25	Czerwone Maki P+R	0.122951
139	Walcownia	0.139535
59	Krowodrza Górka	0.141994
146	Wzgórza Krzesławickie	0.162921
76	Nowy Bieżanów P+R	0.220096

Tabela 19 Poniedziałek 30-07-2018 - średnie najmniejsze opóźnienia danego przystanku

	stopName	stopMeanDelay
150	Łagiewniki ZUS	1.802139
48	Kabel	1.617476
72	Muzeum Lotnictwa	1.580981
75	Nowosądecka	1.574648
31	Dworcowa	1.539024
22	Cmentarz Podgórski	1.533825
146	Zabłocie	1.518447
113	Rzebika	1.511475
88	Piaski Nowe	1.498607
53	Klimeckiego	1.480545

Tabela 20 Wtorek 31-07-2018 - średnie największe opóźnienia danego przystanku

	stopName	stopMeanDelay
23	Cmentarz Rakowicki	0.019417
20	Cichy Kącik	0.020202
59	Krowodrza Górka	0.067692
138	Walcownia	0.073171
25	Czerwone Maki P+R	0.112971
67	Mały Płaszów	0.134328
145	Wzgórza Krzesławickie	0.147929
102	Rakowicka	0.233010
14	Bronowice Małe	0.257576
134	Uniwersytet Ekonomiczny	0.278846

Tabela 21 Wtorek 31-07-2018 - średnie najmniejsze opóźnienia danego przystanku

Wniosek: Możemy zauważyć tendencję, że największe średnie opóźnienia mają miejsce dla przystanków oddalonych od pętli, natomiast najmniejsze – dla przystanków należących do pętli lub znajdujących się w jej pobliżu. Największe opóźnienia możemy zaobserwować na przystankach: Łagiewniki ZUS (średnio 1.97min), Nowosądecka (1.78min), Piaski Nowe (1.70min), Kabel (1.65min), Dworcowa (1.62min). Najmniej natomiast podatne na opóźnienia przystanki to: Cmentarz Rakowicki (0.06min). Cichy Kącik (0.08min), Czerwone Maki P+R (0.13min), Krowodrza Górka (0.14min), Mały Płaszów (0.16min), Wzgórza Krzesławickie (0.16min).

Tabele pokazujące średnie największe i najmniejsze opóźnienie w [min] dla danej linii i kierunku jazdy w danym dniu:

	number	direction	lineMeanDelay
40	22	Walcownia	2.109223
24	14	Bronowice Małe	1.762376
41	24	Bronowice Małe	1.649254
31	19	Borek Fałęcki	1.602434
38	22	Borek Fałęcki	1.567422
25	14	Mistrzejowice	1.515894
18	10	Kopiec Wandy	1.481264
13	6	Salwator	1.340852
39	22	Kombinat	1.316129
21	11	Mały Płaszów	1.232456

Tabela 22 Poniedziałek 23-07-2018 - średnie największe opóźnienia danej linii

	number	direction	lineMeanDelay
20	11	Czerwone Maki P+R	0.289753
44	44	Kombinat	0.395349
30	18	Krowodrza Górka	0.485581
2	2	Cm. Rakowicki	0.512097
33	20	Cichy Kącik	0.526414
29	18	Czerwone Maki P+R	0.580252
19	10	Łagiewniki	0.601317
4	3	Dworzec Tow.	0.691667
8	4	Kombinat	0.692308
35	21	Kombinat	0.700000

Tabela 23 Poniedziałek 23-07-2018 - średnie najmniejsze opóźnienia danej linii

	number	direction	lineMeanDelay
41	24	Bronowice Małe	2.078532
42	24	Kurdwanów P+R	2.005641
31	19	Borek Fałęcki	1.872424
40	22	Walcownia	1.718242
18	10	Kopiec Wandy	1.467078
23	13	Nowy Bieżanów P+R	1.452522
47	50	Prokocim	1.447093
8	4	Kombinat	1.423077
14	8	Borek Fałęcki	1.390722
13	6	Salwator	1.378760

Tabela 24 Wtorek 24-07-2018 - średnie największe opóźnienia danej linii

	number	direction	lineMeanDelay
2	2	Cm. Rakowicki	0.364815
7	4	Bronowice Małe	0.479460
3	2	Salwator	0.480000
20	11	Czerwone Maki P+R	0.542401
10	5	Krowodrza Górka	0.565964
33	20	Cichy Kącik	0.583186
35	21	Kombinat	0.666667
39	22	Kombinat	0.695652
45	44	Kopiec Wandy	0.712329
29	18	Czerwone Maki P+R	0.716060

Tabela 25 Wtorek 24-07-2018 - średnie najmniejsze opóźnienia danej linii

	number	direction	lineMeanDelay
40	22	Walcownia	2.060193
41	24	Bronowice Małe	1.792352
42	24	Kurdwanów P+R	1.626364
13	6	Salwator	1.613215
39	22	Kombinat	1.611111
18	10	Kopiec Wandy	1.494536
38	22	Borek Fałęcki	1.486979
47	50	Prokocim	1.482509
23	13	Nowy Bieżanów P+R	1.470830
14	8	Borek Fałęcki	1.361259

Tabela 26 Środa 25-07-2018 - średnie największe opóźnienia danej linii

	number	direction	lineMeanDelay
20	11	Czerwone Maki P+R	0.240103
2	2	Cm. Rakowicki	0.359922
10	5	Krowodrza Górka	0.379393
7	4	Bronowice Małe	0.575866
44	44	Kombinat	0.583333
3	2	Salwator	0.620690
5	3	Krowodrza Górka	0.638918
43	44	Bronowice	0.639854
33	20	Cichy Kącik	0.659485
19	10	Łagiewniki	0.670946

Tabela 27 Środa 25-07-2018 - średnie najmniejsze opóźnienia danej linii

	number	direction	lineMeanDelay
40	22	Walcownia	1.947727
31	19	Borek Fałęcki	1.642157
38	22	Borek Fałęcki	1.629082
18	10	Kopiec Wandy	1.486525
13	6	Salwator	1.459392
41	24	Bronowice Małe	1.433771
23	13	Nowy Bieżanów P+R	1.419794
39	22	Kombinat	1.307692
42	24	Kurdwanów P+R	1.276281
43	44	Bronowice	1.175182

Tabela 28 Czwartek 26-07-2018 - średnie największe opóźnienia danej linii

	number	direction	lineMeanDelay
20	11	Czerwone Maki P+R	0.311545
2	2	Cm. Rakowicki	0.403377
8	4	Kombinat	0.403846
3	2	Salwator	0.561321
5	3	Krowodrza Górka	0.598140
7	4	Bronowice Małe	0.603598
10	5	Krowodrza Górka	0.607229
16	9	Mistrzejowice	0.628348
17	9	Nowy Bieżanów P+R	0.656393
29	18	Czerwone Maki P+R	0.676543

Tabela 29 Czwartek 26-07-2018 - średnie najmniejsze opóźnienia danej linii

	number	direction	lineMeanDelay
40	22	Walcownia	1.845657
41	24	Bronowice Małe	1.700237
31	19	Borek Fałęcki	1.573589
42	24	Kurdwanów P+R	1.540914
13	6	Salwator	1.533208
47	50	Prokocim	1.396602
23	13	Nowy Bieżanów P+R	1.366318
18	10	Kopiec Wandy	1.335423
24	14	Bronowice Małe	1.230521
38	22	Borek Fałęcki	1.213123

Tabela 30 Piątek 27-07-2018 - średnie największe opóźnienia danej linii

	number	direction	lineMeanDelay
2	2	Cm. Rakowicki	0.262452
3	2	Salwator	0.289431
20	11	Czerwone Maki P+R	0.411924
7	4	Bronowice Małe	0.445732
4	3	Dworzec Tow.	0.547826
33	20	Cichy Kącik	0.558182
10	5	Krowodrza Górka	0.568709
29	18	Czerwone Maki P+R	0.574262
17	9	Nowy Bieżanów P+R	0.601966
19	10	Łagiewniki	0.651976

Tabela 31 Piątek 27-07-2018 - średnie najmniejsze opóźnienia danej linii

	number	direction	lineMeanDelay
47	50	Prokocim	1.671225
18	10	Kopiec Wandy	1.644847
13	6	Salwator	1.460880
40	22	Walcownia	1.423523
31	19	Borek Fałęcki	1.410129
6	3	Nowy Bieżanów P+R	1.281336
42	24	Kurdwanów P+R	1.229674
41	24	Bronowice Małe	1.204842
37	21	Os.Piastów	1.200653
46	50	Krowodrza Górka	1.169622

Tabela 32 Poniedziałek 30-07-2018 - średnie największe opóźnienia danej linii

	number	direction	lineMeanDelay
8	4	Kombinat	0.192308
2	2	Cm. Rakowicki	0.261023
20	11	Czerwone Maki P+R	0.333913
10	5	Krowodrza Górka	0.405449
44	44	Kombinat	0.458101
30	18	Krowodrza Górka	0.459471
39	22	Kombinat	0.460526
3	2	Salwator	0.510172
29	18	Czerwone Maki P+R	0.526405
33	20	Cichy Kącik	0.560446

Tabela 33 Poniedziałek 30-07-2018 - średnie najmniejsze opóźnienia danej linii

	number	direction	lineMeanDelay
18	10	Kopiec Wandy	1.717129
40	22	Walcownia	1.659769
13	6	Salwator	1.392991
41	24	Bronowice Małe	1.387524
25	14	Mistrzejowice	1.387027
31	19	Borek Fałęcki	1.383629
42	24	Kurdwanów P+R	1.377037
23	13	Nowy Bieżanów P+R	1.344026
47	50	Prokocim	1.287571
14	8	Borek Fałęcki	1.271654

Tabela 34 Wtorek 31-07-2018 - średnie największe opóźnienia danej linii

	number	direction	lineMeanDelay
20	11	Czerwone Maki P+R	0.258844
2	2	Cm. Rakowicki	0.370642
3	2	Salwator	0.521242
8	4	Kombinat	0.596154
10	5	Krowodrza Górka	0.605833
39	22	Kombinat	0.629139
30	18	Krowodrza Górka	0.649674
43	44	Bronowice	0.653775
1	1	Wzgórza K.	0.655473
45	44	Kopiec Wandy	0.664894

Tabela 35 Wtorek 31-07-2018 - średnie najmniejsze opóźnienia danej linii

Wniosek: Można zauważyć, że tendencję do spóźniania mają tramwaje o dużej liczbie przystanków na trasie oraz których trasa przebiega przez centrum miasta, gdzie ruch tramwajowy krzyżuje się z ruchem samochodowym. Największe opóźnienia możemy zaobserwować na liniach/ kierunkach: 22/ Walcownia/Kombinat (1.70min), 24/ Bronowice Małe (1.61min), 10/ Kopiec Wandy (1.52min), 24/ Kurdwanów P+R (1.51min), 6/ Salwator (1.45min). Najmniej natomiast podatne na opóźnienia przystanki to: 11/ Czerwone Maki P+R (0.34min), 2/ Cm. Rakowicki (0.36min), 2/ Salwator (0.50min), 5/ Krowodrza Górka (0.52min).

Poniżej znajdują się średnie wielkości błędu dla danych kombinacji dla poszczególnych dni stosując algorytm uczenia maszynowego:

	np.mean
feats1	54.362443
feats2	52.573274
feats3	50.683268
feats4	49.895306
feats5	48.111726
feats6	48.218893
feats7	48.227872

Tabela 36 Poniedziałek 23-07-2018 - średni błąd dla poszczególnych kombinacji w [s]

```
feats5 = [
    'number',
    'stop',
    'direction_cat',
    'vehicleId',
    'seq_num'
]
```

	np.mean
feats1	54.710845
feats2	53.646152
feats3	51.493884
feats4	49.650312
feats5	48.743480
feats6	48.317763
feats7	48.331660

Tabela 37 Wtorek 24-07-2018 - średni błąd dla poszczególnych kombinacji w [s]

```
feats6 = [
    'number',
    'stop',
    'direction_cat',
    'vehicleId',
    'seq_num',
    'number_direction_id'
]
```

	np.mean
feats1	54.299938
feats2	52.975194
feats3	50.564410
feats4	48.850661
feats5	47.477024
feats6	47.470479
feats7	47.451824

Tabela 38 Środa 25-07-2018 - średni błąd dla poszczególnych kombinacji w [s]

```
feats7 = [
    'number',
    'stop',
    'direction_cat',
    'vehicleId',
    'seq_num',
    'number_direction_id',
    'stop_direction_id'
]
```

	np.mean
feats1	53.078118
feats2	52.002295
feats3	49.316307
feats4	47.338965
feats5	46.426719
feats6	46.514335
feats7	46.483185

Tabela 39 Czwartek 26-07-2018 - średni błąd dla poszczególnych kombinacji w [s]

```
feats5 = [
    'number',
    'stop',
    'direction_cat',
    'vehicleId',
    'seq_num'
]
```

	np.mean
feats1	53.217528
feats2	51.701163
feats3	49.288203
feats4	47.708430
feats5	46.310295
feats6	46.220786
feats7	46.123915

Tabela 40 Piątek 27-07-2018 - średni błąd dla poszczególnych kombinacji w [s]

```
feats7 = [
    'number',
    'stop',
    'direction_cat',
    'vehicleId',
    'seq_num',
    'number_direction_id',
    'stop_direction_id'
]
```

	np.mean
feats1	52.501657
feats2	50.870876
feats3	48.684770
feats4	47.611281
feats5	46.554821
feats6	46.476108
feats7	46.344128

Tabela 41 Poniedziałek 30-07-2018 - średni błąd dla poszczególnych kombinacji w [s]

```
feats7 = [
    'number',
    'stop',
    'direction_cat',
    'vehicleId',
    'seq_num',
    'number_direction_id',
    'stop_direction_id'
]
```

	np.mean
feats1	54.790189
feats2	53.142940
feats3	50.543639
feats4	49.506352
feats5	48.451810
feats6	48.207503
feats7	48.180627

Tabela 42 Wtorek 31-07-2018 - średni błąd dla poszczególnych kombinacji w [s]

```
feats7 = [
    'number',
    'stop',
    'direction_cat',
    'vehicleId',
    'seq_num',
    'number_direction_id',
    'stop_direction_id'
]
```

Wniosek: Wyniki się różnią ale tylko nieznacznie. Obserwujemy, iż w większości przypadków wpływ na obniżenie błędu mają:

- numer tramwaju
- numer przystanku
- kierunek jazdy tramwaju
- numer pojazdu
- kolejność przystanku na trasie
- numer tramwaju wraz z kierunkiem
- przystanek wraz z kierunkiem

Podsumowanie

Analizując dane z siedmiu dni roboczych miesiąca wakacyjnego, w których dokonano analizy można wyciągnąć następujące wnioski:

1. Na czas pojawia się średnio ok. 43% tramwajów.
2. Mimo, że ponad połowa tramwajów przyjeżdża opóźniona, to jest to opóźnienie niewielkie, czyli 1-2min. Średnio 89% tramwajów przyjeżdża z opóźnieniem max. 2min. Zalecane jest natomiast przeprowadzenie analizy również dla danych z okresu trwania roku akademickiego. Można przypuszczać, że średnie opóźnienia wzrosną.
3. Największe opóźnienia można odnotować na przystankach oddalonych od pętli, gdy trasa tramwaju biegnie przez centrum miasta lub krzyżuje się z ruchem samochodowym oraz gdy trasa tramwaju jest długa.
4. Najmniejsze opóźnienia można odnotować na przystankach blisko pętli, gdy trasa tramwaju omija centrum miasta, ma wydzielony pas tramwajowy oraz jest stosunkowo krótka.
5. Rozkład opóźnień tramwajów w ciągu dnia jest zmienny, jednak w przypadku niektórych analizowanych dni ciężko jest zauważyć tendencję do wyraźnego pojawiania się największych opóźnień w godzinach szczytu (rano i po południu). Może być to spowodowane faktem, że analiza jest dokonywana w miesiącach wakacyjnych.
6. Celem osiągnięcia jeszcze lepszych rezultatów warto byłoby również przeanalizować jaki wpływ na opóźnienie ma numer pojazdu (możemy przypuszczać, iż starsze składy są wolniejsze, a nowsze są w stanie szybciej rozwinąć prędkość) oraz przeanalizować zależność opóźnienia tramwaju na danym przystanku przy uwzględnieniu kierunku jazdy (można założyć, że tramwaj na przystanku znajdującym się bezpośrednio przy pętli odnotuje niewielkie opóźnienie, jeśli pojazd dopiero co wyjechał z pętli, natomiast dla tramwaju zmierzającego w kierunku pętli na tym przystanku zostanie odnotowane dużo większe opóźnienie, gdyż pojazd kończy bieg).

Bibliografia

Literatura

1. Albon, Chris. 2018. Uczenie maszynowe w Pythonie. Receptury. Gliwice: Wydawnictwo Helion.
2. Geron, Aurelien. 2018. Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow. Gliwice: Wydawnictwo Helion.
3. McKinney, Wes. 2018. Python w analizie danych. Gliwice: Wydawnictwo Helion.

Kursy internetowe

4. Brunner, Rene. Python fuer Data Science, Maschinelles Lernen & Visualization. Udemy.com
5. **Vladimir Alekseichenko, Korona Wyzwań Uczenia Maszynowego, Data Workshop, https://dataworkshop.eu/challenge**

Źródła internetowe

6. https://aczepielik.github.io/post/kraktram/#regresja-kwantylowa
7. https://github.com/aczepielik/KRKtram/tree/master/reports
8. https://mateuszgrzyb.pl/3-najlepsze-sciagawki-z-bibliotek-python/