TRAM DELAYS ANALYSIS

Capstone Project – Data Scientist Nanodegree

ANNA DUTKIEWICZ
May 14, 2020

Table of Content

Chapter 0 - Project Overview 2

Chapter 1 - Introduction 3

1. 1 Introduction 3

1.2 Purpose of the study 3

Chapter 2 – Data Analysis 4

2.1 Data 4

2.2 Source code on the example of May 23, 2018 5

Libraries 5

Data 6

Delays distribution 6

Delays over certain period of time 7

Delays for given lines 9

Machine Learning model 11

2.3 Delay analysis on the example of selected days of the week 16

Summary 32

Bibliography 33

Books 33

Online courses 33

Internet sources 33

Chapter 0 - Project Overview

This project aims to look at the delay of trams in the city of Krakow, Poland. It consists of two chapters and a short summary. The Python programming language was used for analysis due to its simplicity and rich number of libraries.

The first chapter covers the introduction and the purpose of the study, which explains what was the inspiration and where you can download the necessary data for analysis. The second chapter is a bit more extensive. At the beginning, the structure of data is discussed along with their explanation. In turn, we proceed to clarify the source code in Python based on a thorough analysis of the data collected on Monday July 23, 2018. This includes:

- Information about the libraries used;
- Explanation how to download data;
- Checking average delays for the analyzed tram lines;
- How delays spread throughout the day;
- Which tram stops and lines are most and least susceptible to delays;
- A simple model presenting the general sense of the use of machine learning, on the basis of which you can then predict delays.

An overall delay analysis was then performed on other days of the week to compare the results.

At the very end there is a short summary with the conclusions.

Chapter 1 - Introduction

1. 1 Introduction

. The subject of the study will be the analysis of Krakow tram delays. The inspiration is the Crown of Challenges of Machine Learning Data Workshop [5] and the article [6], which I came across on one of the websites. I decided to use the data downloaded by the author [7] to be able to analyze them and draw conclusions. There is also no obstacle to download these data yourself, because for some time in the browser it is possible to track information about the arrival time of a given tram at the stop, taking into account its delay. These are the same data that we will find on the boards at the tram stops. This information can be found on the TTSS (Traffic Tram Supervision System) website: ttss.krakow.pl. On the subpage http://www.ttss.krakow.pl/internetservice/ we can also track much more interesting information, where we can check the current location of trams on the map and their anticipated delay. It is also possible to track and analyze bus traffic by downloading data from https://mpk.jacekk.net/

1.2 Purpose of the study

As we all know, trams are often late. Once it is 1 minute, other times it can be 20 minutes. Sometimes delays can have unpleasant consequences when, for example, we are late for an important meeting with the client. In this work, I plan to check if there is any relationship between the time the tram arrives and the line, stop, time or distance from the loop. So when according to the schedule our tram should arrive at 7:30am, but our model says that at this time the average delay is e.g. 8 minutes, despite the fact that we left 2 minutes too late from home, there is a high probability that we will reach this tram .

Chapter 2 – Data Analysis

2.1 Data

The data used are available in the .csv format on the website [7]. They concern the city of Krakow, in this case tram traffic. They cover selected days of July.

	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

Table 1 Data overview

Data explanation:

index – numbers of subsequent rounds of server queries (the round includes all stops, it lasts 20 seconds)

time_stamp – time of sending the query to the server (rounding to the nearest minutes we can identify with the real time of departure)

stop – stop number

stopName – stop name

number – tram number

direction – tram direction

plannedTime – scheduled departure time

vehicleId – vehicle number

tripId – trip number

status – status (PLANNED – not tracked, PREDICTED - expected, STOPPING – already on tram stop)

delay – calculated delay

seq_num – stops sequence on the route

2.2 Source code on the example of May 23, 2018

Libraries

The following libraries were used for the analysis:

- pandas – this package provides high-level data structures and functions that speed up the work with structured data as well as table data; it is thanks to her that Python has become a solid analytical environment [3];
- NumPy – this package is the basic tool for performing numerical calculations in Python; this library supports data structures, algorithms and binding mechanisms necessary for most scientific applications related to the purpose of numerical data [3];
- Matplotlib – this package is the most popular Python library designed for creating charts and other two-dimensional data visualizations, this library is created for creating charts suitable for publication;
- Seaborn – Seaborn is a "overlay" for matplotlib, by design it is to enable the construction of nice charts in a slightly simpler way [8];
- Scikit-learn – this package is currently considered by Python programmers to be the most important set of machine learning tools; includes modules supporting, among others, models: classification, regression, cluster analysis, reduction of the number of dimensions, model selection, initial data processing [3].

We import *DecisionTreeRegressor* from the Scikit-learn package, as this is a regression problem. In this case, we perform delay forecasting, which will be given in minutes. Of course, most delays are 1, 2, 3 minutes and we could treat it as a classification, but if we have a delay of 15 minutes we will have 15 classes. This is a significant number and in this case classification management will not be the best approach. To sum up, we choose regression because the value we forecast is a continuous value. *cross_val_score*, i.e. cross-validation, will be useful later

```
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
```

Data

The analysis below covers data collected on May 23, 2018 (Monday).

At the beginning we load our data, where we paste the URL into our data. In turn, using the head command we load the first 5 lines to look at what our data looks like in table form.

```
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

Table 2 Preview of the first five lines of data

Delays distribution

We can check what are the delays and what are the most common:

```
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
```

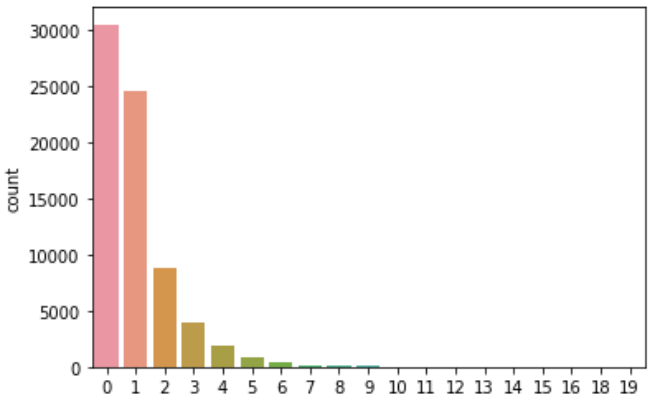

Fig. 1 Histogram showing the number of vehicles depending on the delay in [min]

Sometimes it is good to see the number of trams that did not arrive on time, but sometimes it is difficult to estimate whether the number of vehicles that arrived on time, i.e. 30,531 is a lot or not. Normalization will help us:

```
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
```

From the pessimist's point of view, it can be concluded that more than half of the trams arrive late. However, looking at it from the other side, the vast majority of trams are a little late - up to 2 minutes. The average lateness in the trial was approximately 1 minute 1 second (1.01 minutes), and the standard deviation 1 minute and 21 seconds (1.35 minutes).

Delays over certain period of time

Let's see the distribution of delays throughout the day depending on the time:

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

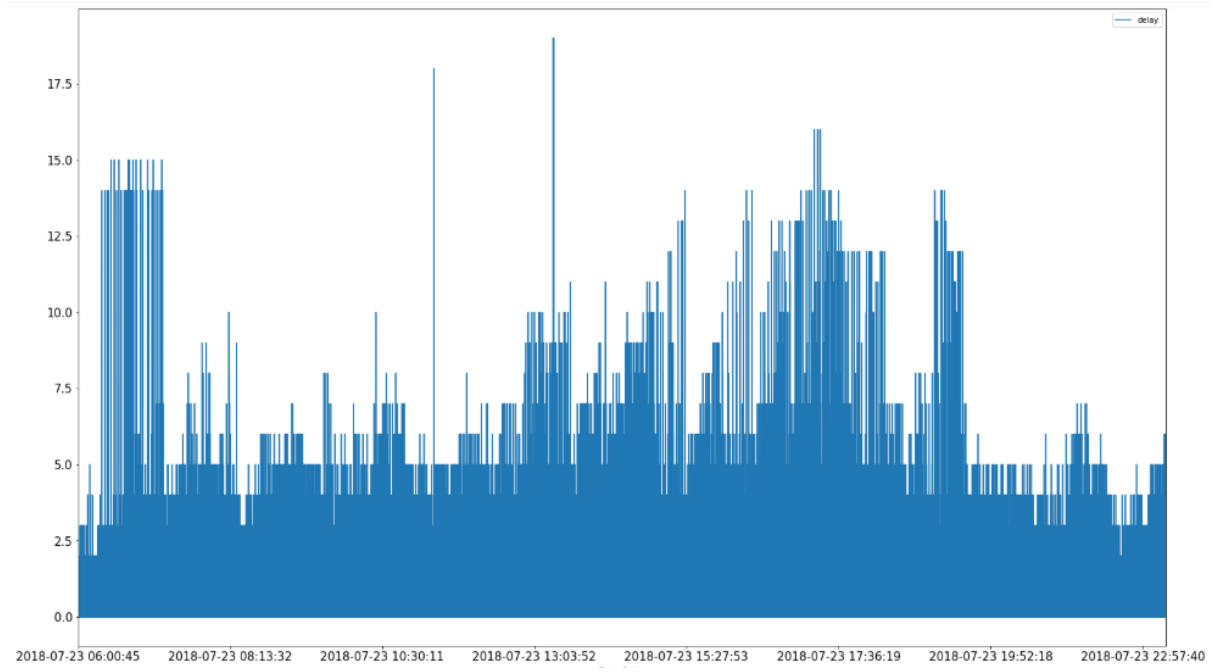

Fig. 2 Chart showing vehicle delay in [min] depending on the time of day

The greatest lateness can be observed during rush hour (commuting), the smallest in the early morning, morning and evening hours. Between 6:30am and 7:00am in the morning they can reach up to 15 minutes, then fall sharply to 4-5 minutes. From 1:00pm to 4:30pm they begin to slowly increase, which may mean that we will have to wait even an additional 10min at the stop. During these hours, the greatest delays occur at full hours, which may be due to the end of the working day. From 4:30pm to 6:00pm we can observe the afternoon rush hours involving delays of up to 15 minutes and similar delays in the hours. 7:00pm-7:30pm (e.g. evening meetings with friends). After 7:30pm delays are already small.

Delays at specific stops

Let's check at which stops the largest and the lowest delays can be expected.

```
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)
```

Stops with the highest average delay:

	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

Table 3 Average delay for a given stop [min]

Stops with the lowest average delay:

	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

Table 4 Average delay for a given stop [min]

From the above statements, we can conclude that the closer the stop is to the loop, the smaller the delay, the smallest is at the initial stops when the tram is just leaving the loop. The greatest delays can be observed at stops near places where tram traffic intersects with car traffic (e.g. near roundabouts, where trams often have to stop at a red light).

Delays for given lines

Let's check which tram lines are most susceptible to delays.

```
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).h
ead(10)
```

Lines with the largest average delay:

	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

Table 5 Average delay of a given line [min]

Lines with the lowest average delay:

	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

Table 6 Average delay for a given line [min]

The above data is presented in the chart below:

```
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)
```

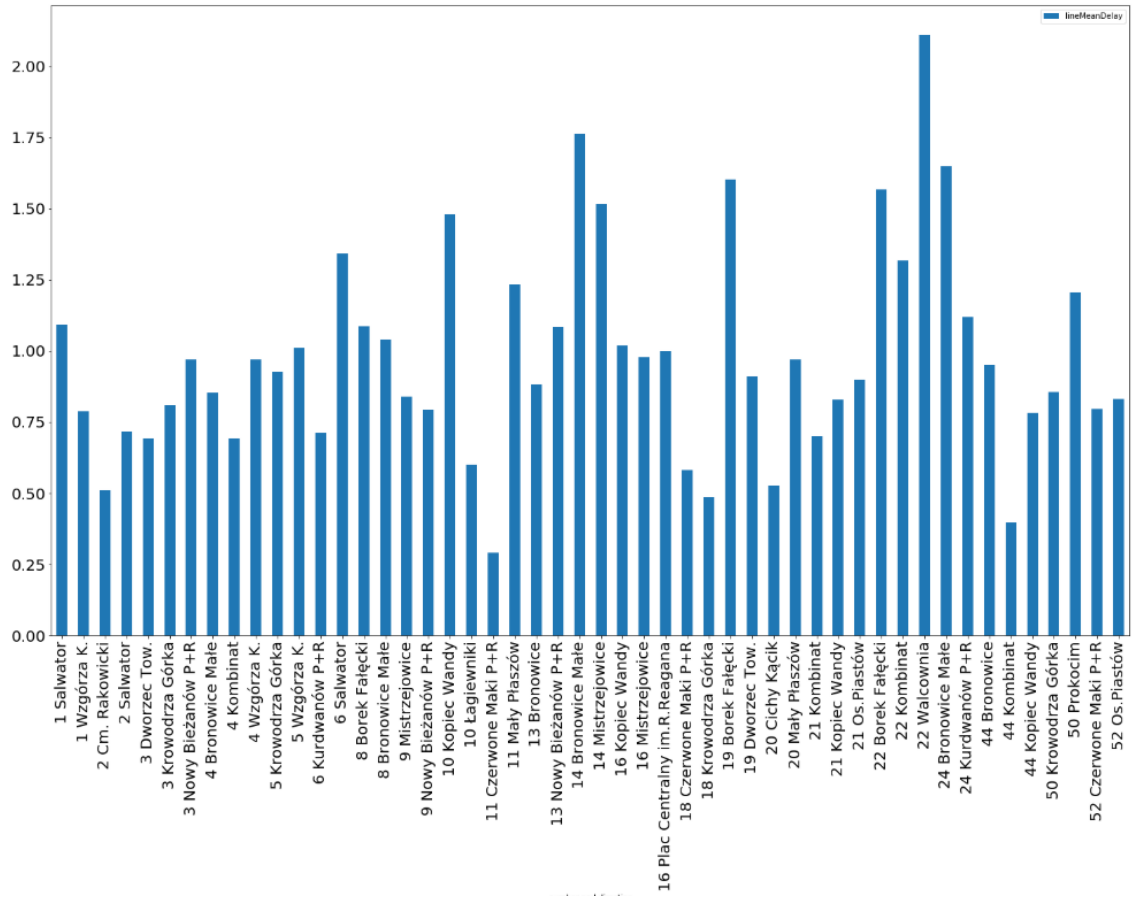

Fig. 3 Chart showing the average delay in [min] for a given line and direction of travel

While the shortest lines tend to have a small average delay, there is no general, strong relationship.

Machine Learning model

A simple model has been built below to show the general sense of using machine learning, which can then be used to predict latency. Seven combinations have been created for the needs of the model. By training the model the error size was obtained.

plannedTime is a string, so you must convert it to date format.

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

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

To avoid analyzing our delay somewhere after the decimal point, we change minutes to seconds:

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

The direction of tram travel (direction) is a string. So we project textual values into numerical values. We do this by assigning a unique numeric value for each value (direction), this can be treated as an ID

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

As we can guess, deleting data is a very weak strategy. So in order to prevent the model from spilling out due to some empty values in our set and at the same time without getting rid of this data, we can assign them some value. Why -1? It is important that this value is unique and does not repeat, because if not, we can accidentally overwrite any existing value.

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

We can also combine two variables, e.g. tram number and direction or stop and direction in which this tram is going. We can use the apply function for this.

```
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]
```

As for the variable X, these will be the values that affect the delay, this is a list. y will be a vector because it contains only one value associated with the delay. Calculations were made for various combinations of variables to check which one would be the most optimal.

```
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
```

Then we build the model. In this case, we use k-fold cross validation, which allows you to use the entire data set for both learning and model validation. The training set is divided into k equal subsets, in this case 5, of which k-1 is used to train the model, while 1 subset is used to validate the model.

```
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'])
```

Let's see how the results for each combination are presented. The closer the result is to zero, the more accurate our model.

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

Table 7 The average error for individual combinations in [sec]

```
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
```

The following combination was the most preferred option:

- tram number
- stop number
- direction
- vehicle number
- stop sequence on the route

2.3 Delay analysis on the example of selected days of the week

Histograms showing the number of vehicles depending on the delay in [sec] for a given day:

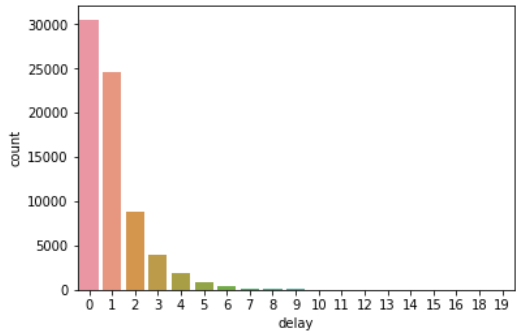

Fig. 4 Monday July 23, 2018

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

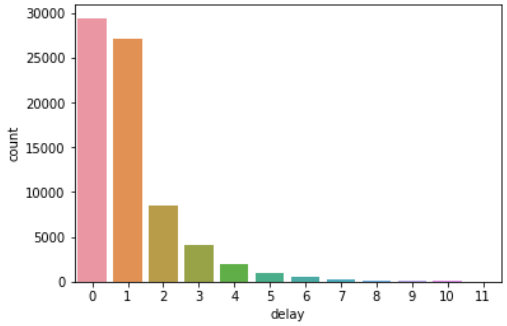

Fig. 5 Tuesday July 24, 2018

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

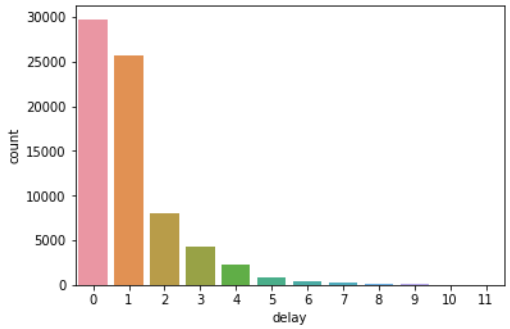

Fig. 6 Wednesday July 25, 2018

```
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
```

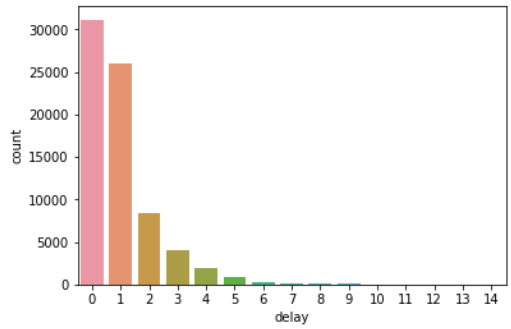

Fig. 7 Thursday July 26, 2018

```
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
```

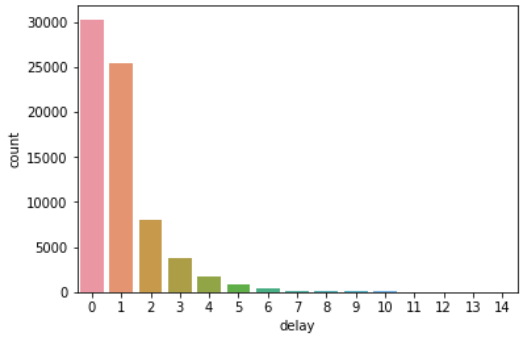

Fig. 8 Friday July 27, 2018

```
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
```

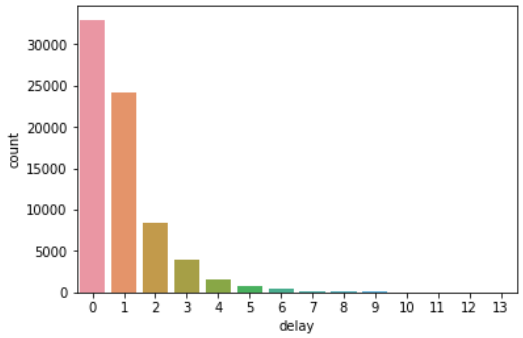

Fig. 9 Monday July 30, 2018

```
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
```

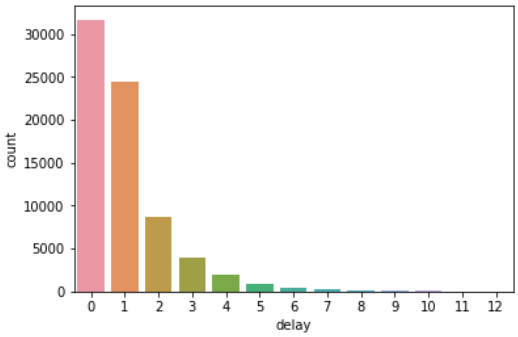

Fig. 10 Tuesday July 31, 2018

```
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
```

Conclusion: The results for all cases are similar. More than half of the trams arrive late, but the vast majority of trams are a little late - up to 2 minutes. The average delay is approximately 1 minute.

Figures showing the distribution of vehicle delays in [min] depending on the time of day:

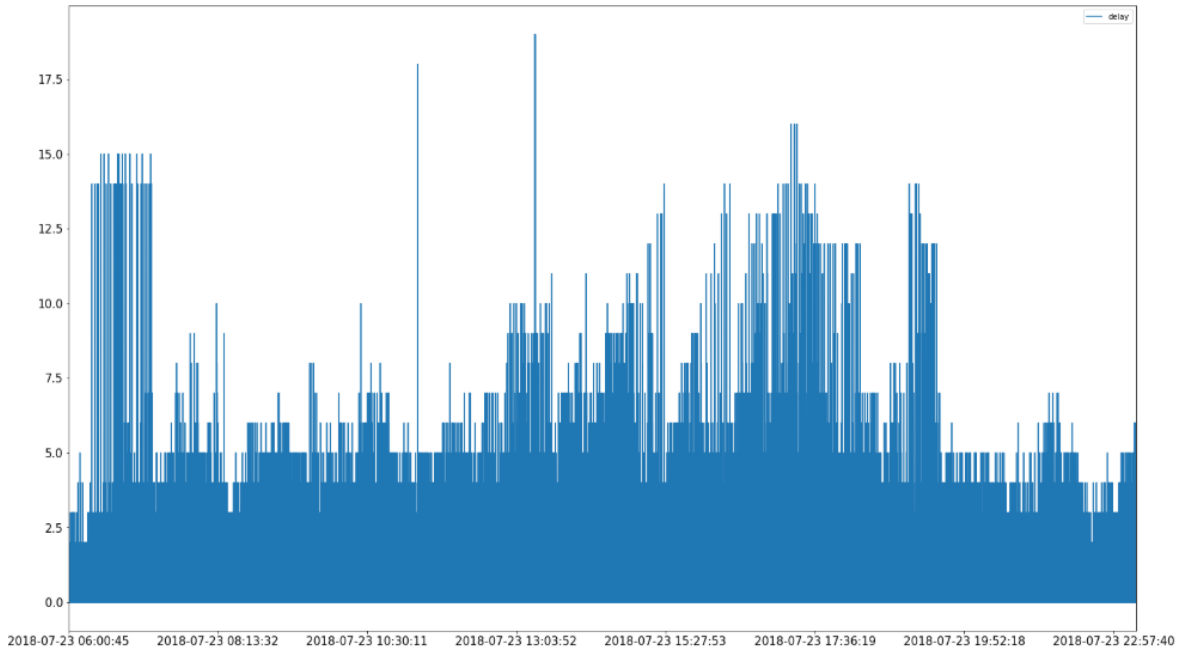

Fig. 11 Monday July 23, 2018

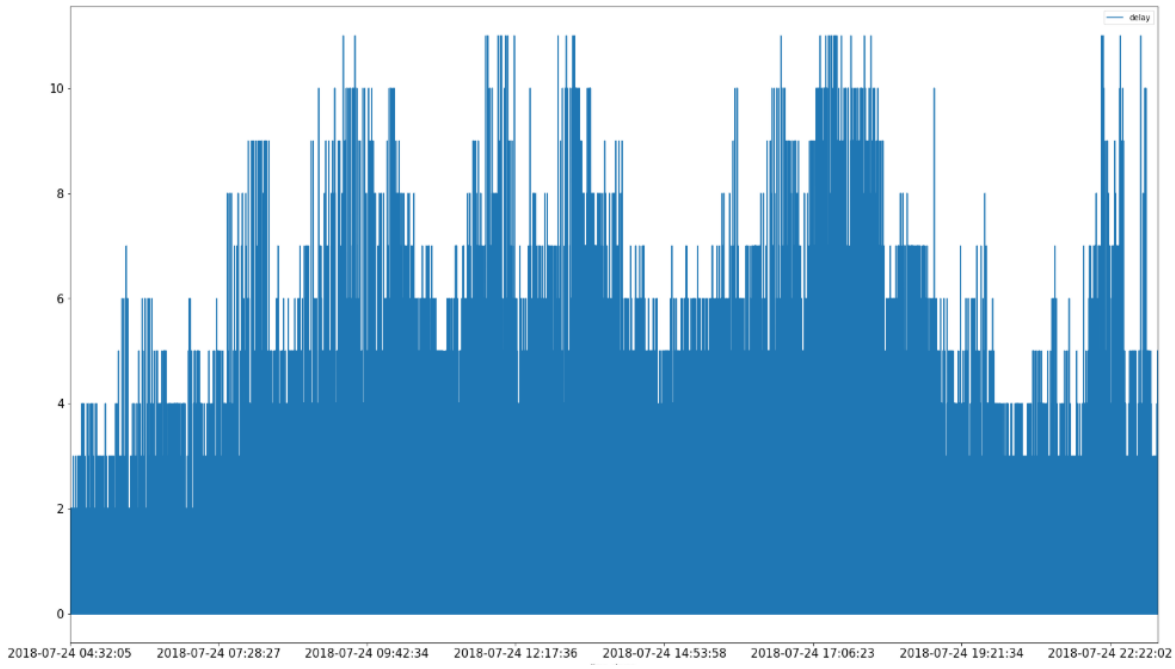

Fig. 12 Tuesday July 24, 2018

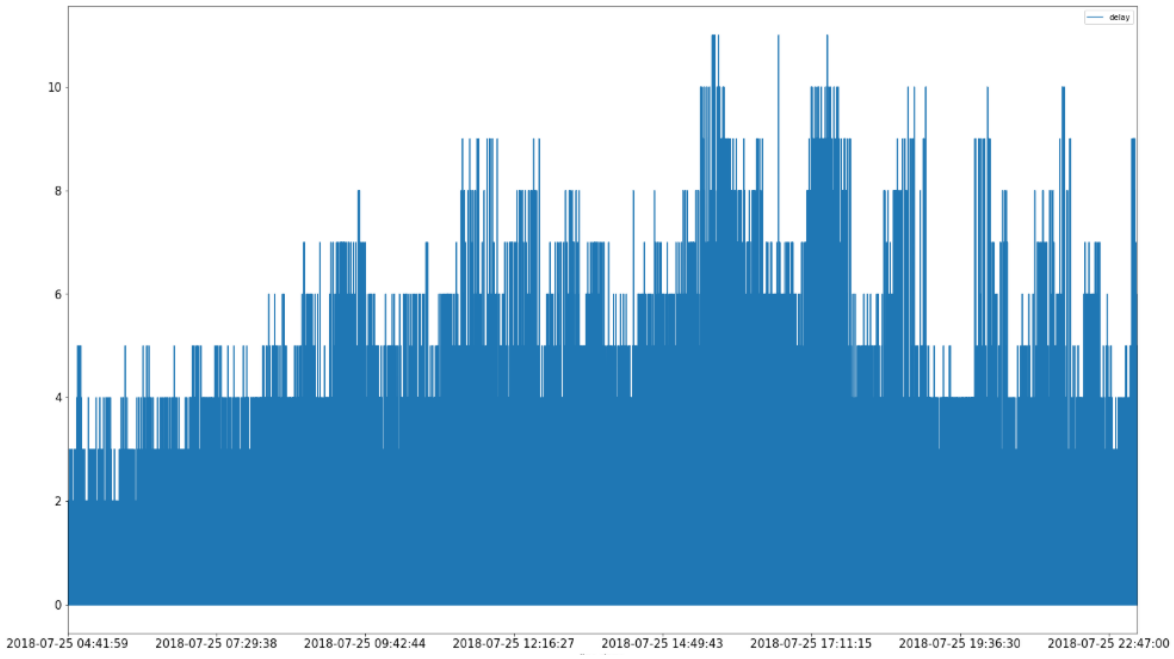

Fig. 13 Wednesday July, 25 2018

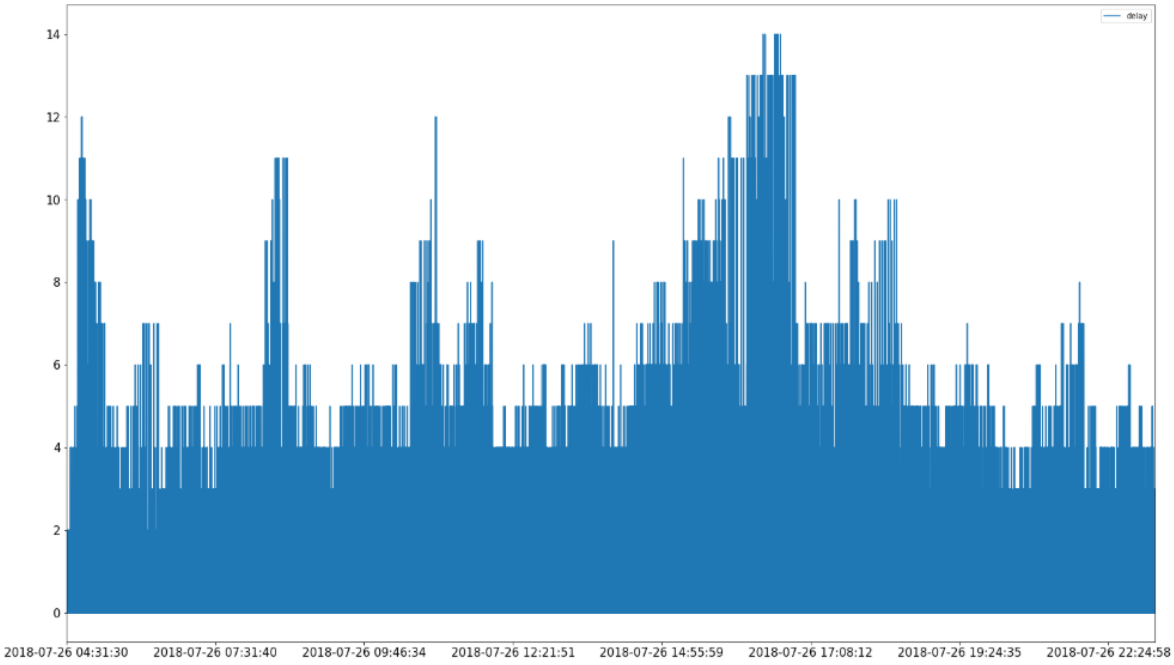

Fig.. 14 Thursday July 26, 2018

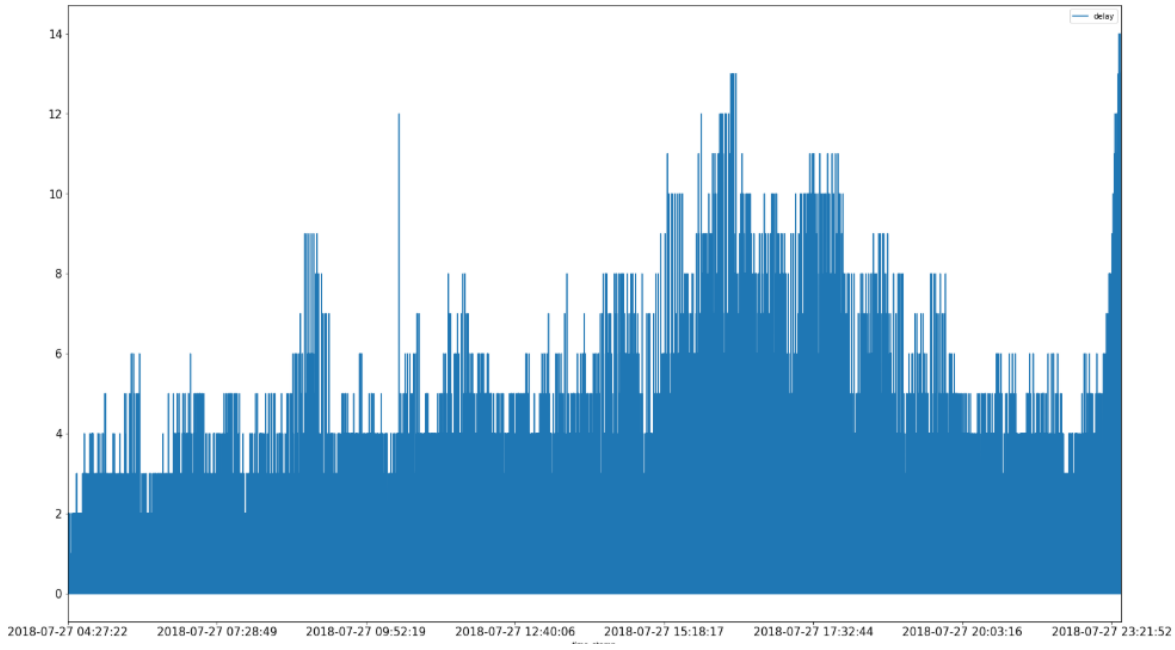

Fig. 15 Friday July 27, 2018

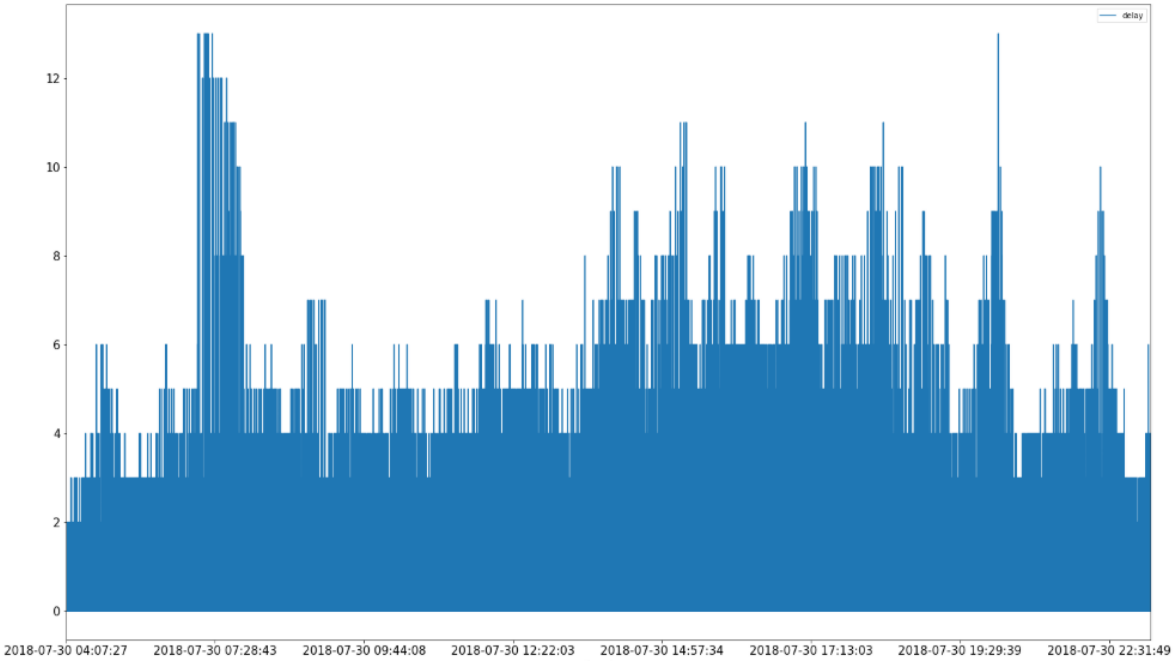

Fig. 16 Monday July 30, 2018

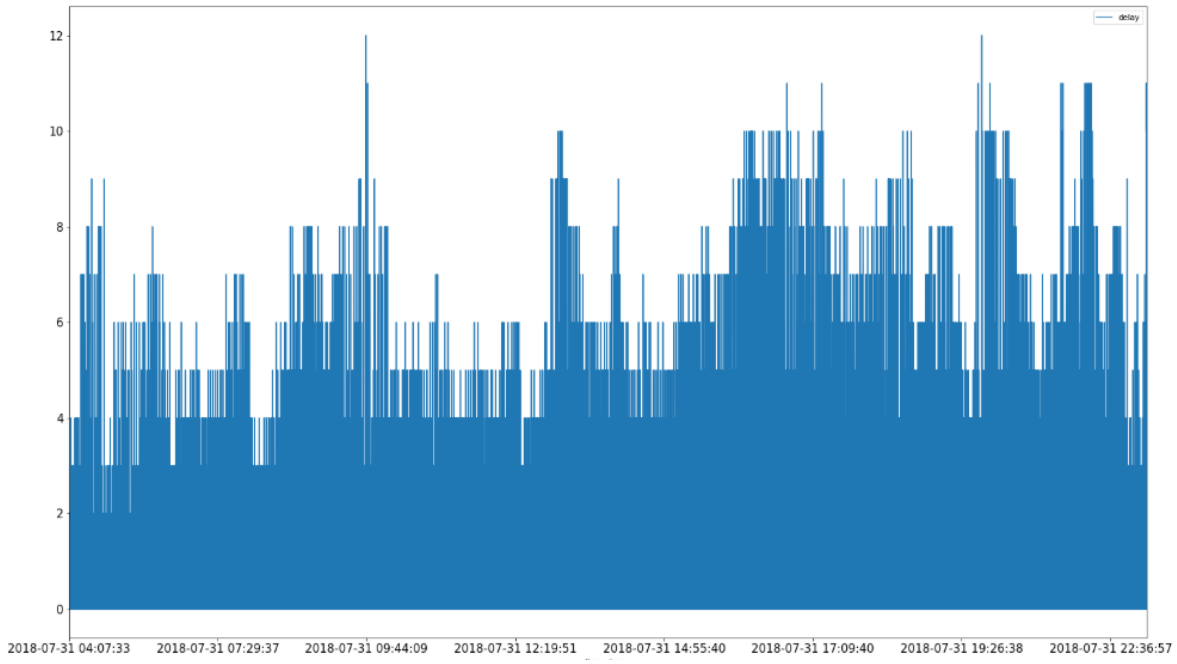

Fig. 17 Tuesday July 31, 2018

Conclusion: Analyzing the distribution of delays during the day in the previous chapter (Monday July 23, 2018), I expected to get similar charts for the remaining days. Guided by intuition, we should observe the peak of delays in the morning and afternoon - for individual days we can observe a tendency to a fairly even distribution of delays throughout the day, e.g. Tuesday July 24, 2018, Wednesday July 25, 2018, or Tuesday July 31, 2018. However, it is worth taking into account the fact that measurements are taken in the middle of the summer vacation, which may show some disturbances related to the fact that some people are on holiday, the academic year has not started, and children do not attend school or kindergarten.

Tables showing the average highest and lowest delay in [min] for a given stop on a given day:

	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

Table 8 Monday July 23, 2018 - average biggest delays of a given stop

	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

Table 9 Moday July 23, 2018 - average smallest delays of a given stop

	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

Table 10 Tuesday July 24, 2018 - average biggest delays of a given stop

	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

Table 11 Tuesday July 24, 2018 - average smallest delays of a given stop

	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

Table 12 Wednesday July 25, 2018 - average biggest delays of a given stop

	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

Table 13 Wednesday July 25, 2018 - average smallest delays of a given stop

	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

Table 14 Thursday July 26, 2018 - average biggest delays of a given stop

	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

Table 15 Thursday July 26, 2018 - average smallest delays of a given stop

	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

Table 16 Friday July 27, 2018 - average biggest delays of a given stop

	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

Table 17 Friday July 27, 2018 - average smallest delays of a given stop

	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

Table 18 Monday July 30, 2018 - average biggest delays of a given stop

	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

Table 19 Monday July 30, 2018 - average smallest delays of a given stop

	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

Table 20 Tuesday July 31, 2018 - average biggest delays of a given stop

	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

Table 21 Tuesday July 31, 2018 - average smallest delays of a given stop

Conclusion: We can see a tendency that the largest average delays occur for stops distant from the loop, while the smallest - for stops belonging to or close to the loop. The biggest delays can be observed at the following stops: Łagiewniki ZUS (average 1.97min), Nowosądecka (1.78min), Piaski Nowe (1.70min), Kabel (1.65min), Dworcowa (1.62min). The least prone stops are: Rakowicki Cemetery (0.06min). Cichy Kącik (0.08min), Czerwone Maki P + R (0.13min), Krowodrza Górka (0.14min), Mały Płaszów (0.16min), Wzgórza Krzesławickie (0.16min).

Tables showing the average highest and lowest delay in [min] for a given line and direction of travel on a given day:

	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

Table 22 Monday July 23, 2018 - average biggest delay for a given line

	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

Table 23 Monday July 23, 2018 - average smallest delay for a given line

	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

Table 24 Tuesday July 24, 2018 - average biggest delay for a given line

	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

Table 25 Tuesday July 24, 2018 - average smallest delay for a given line

	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

Table 26 Wednesday July 25, 2018 - average biggest delay for a given line

	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

Table 27 Wednesday July 25, 2018 - average smallest delay for a given line

	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

Table 28 Thursday July 26, 2018 - average biggest delay for a given line

	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 Thursday July 26, 2018 - average smallest delay for a given line

	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

Table 30 Friday July 27, 2018 - average biggest delay for a given line

	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

Table 31 Friday July 27, 2018 - average smallest delay for a given line

	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

Table 32 Monday July 30, 2018 - average biggest delay for a given line

	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

Table 33 Monday July 30, 2018 - average smallest delay for a given line

	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

Table 34 Tuesday July 31, 2018 - average biggest delay for a given line

	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

Table 35 Tuesday July 31, 2018 - average smallest delay for a given line

Conclusion: It can be noticed that trams with a large number of stops on the route have a tendency to be late and whose route runs through the city center, where tram traffic intersects with car traffic. The largest delays can be observed on lines / directions: 22 / Walcownia / Kombinat (1.70min), 24 / Bronowice Małe (1.61min), 10 / Wanda Mound (1.52min), 24 / Kurdwanów P + R (1.51min), 6 / Salwator (1.45min). The least prone stops are: 11 / Czerwone Maki P + R (0.34min), 2 / Cm. Rakowicki (0.36min), 2 / Salwator (0.50min), 5 / Krowodrza Górka (0.52min).

Below are the average error rates for given combinations for particular days using a machine learning algorithm:

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

Table 36 MondayJjuly 23, 2018 - average error for individual combinations in [sec]

```
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

Table 37 Tuesday July 24, 2018 - average error for individual combinations in [sec]

```
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

Table 38 Wednesday July 25, 2018 - average error for individual combinations in [sec]

```
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

Table 39 Thursday July 26, 2018 - average error for individual combinations in [sec]

```
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 [sec]

```
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

Table 41 Monday July 30, 2018 - average error for individual combinations in [sec]

```
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

Table 42 Tuesday July 31, 2018 - average error for individual combinations in [sec]

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

Conclusion: The results differ but only slightly. We observe that in most cases they have an impact on reducing the error:

- tram number
- stop number
- direction
- vehicle number
- stop sequence on the route
- tram number with direction
- stop number with direction

Summary

Analyzing data from seven working days of the holiday month in which the analysis was made, the following conclusions can be drawn:

1. On average, approximately 43% of trams appear on time.
2. Although more than half of the trams arrive late, it is a slight delay, i.e. 1-2 minutes. On average, 89% of trams arrive late with max. 2min. However, it is recommended to carry out the analysis also for data from the period of the academic year. It can be assumed that average delays will increase.
3. The greatest delays can be seen at stops away from the loop, when the tram route runs through the city center or intersects with traffic, and when the tram route is long.
4. The smallest delays can be noted at stops close to the loop, when the tram route bypasses the city center, has a separate tram lane and is relatively short.
5. The distribution of streetcar delays during the day is variable, but for some of the days analyzed it is difficult to notice the tendency for the biggest delays to appear clearly during peak hours (morning and afternoon). This may be due to the fact that the analysis is carried out during the holiday months.
6. In order to achieve even better results, it would also be worth analyzing what effect the vehicle number has on the delay (we can assume that older trains are slower, and newer ones are able to develop speed faster) and analyze the dependence of the tram delay at a given stop taking into account the direction of travel (it can be assumed that the tram at the stop directly next to the loop will notice a slight delay if the vehicle has just left the loop, while for a tram heading towards the loop at this stop a much greater delay will be noted as the vehicle ends its run).

Bibliography

Books

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.

Online courses

4. Brunner, Rene. Python fuer Data Science, Maschinelles Lernen & Visualization. Udemy.com
5. Vladimir Alekseichenko, The Crown of Machine Learning Challenges, Data Workshop, https://dataworkshop.eu/challenge

Internet sources

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/