Taxi Riders Hotspot Prediction

Lars Fikkers (108012050) & Pascal Roose (108012051)

Table of Contents

1. Introduction

For our term project we decided to join the Taxi Riders Hotspot Prediction competition. The goal of this project is to predict the amount of taxi customers there will be for an hour slot in various locations in the Neihu district in the northeast of Taipei.

2. Project approach

The first step we took was defining the problem. More specific, what was the desired output and what were the sub issues we had to solve to get the desired output. We then started dividing the workload into two parts, one for each. We saw two major areas of work to be done, these being data analysis and calculating the predictions. Pascal worked on data analysis, and Lars on calculating the predictions.

After we had defined the problem we started analysing the data so we could formulate a calculated approach. We then proceeded by making a flow diagram (see appendix a). We then started researching for a correct model, and prepared it for use. We then started to think of a few model setup and tested them to see if they had the desired results. We collected the results and made our conclusions.

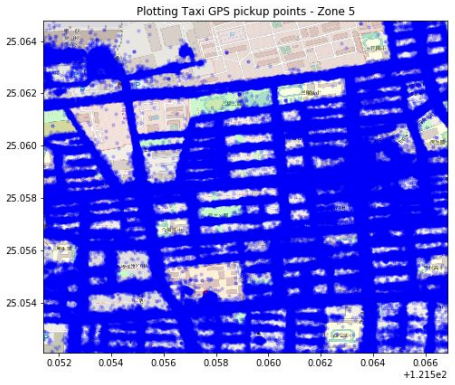
3. Environment setup

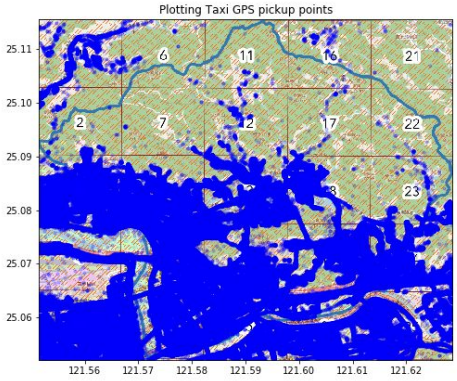
The environment we used to run our test was a Jupyter Notebook. It was run on a Ubuntu 18.04 LTS machine hosted by DigitalOcean. The framework we used to program was PySpark (Spark for Python).

We switched between different hardware setups throughout testing. Meaning we had a differing amount of virtual CPUs, RAM & storage capacity while running our test. This was to speed up or slow down during periods where one of these options was more favorable. Because DigitalOcean isn't free and more resources cost more money per hour, we only sped up during the testing phase and mostly ran on the basic setup during the research and design phases.

4. Data analysis

Before we could start designing the AI that would predict the amount of passengers for a given place, we would know the nature of data we're dealing with. We wanted to have low entropy variables for input, so whatever algorithm we would end up using would be able to make more sense of the data so to say. We therefore started looking at the structure of the files, and determining the variables we had to work with. We got confused as to the purpose of the zones and the meaning of a 'hotspot'. This lead us to plot the data on a map of the Neihu district. The resulting images are shown in the following figures.

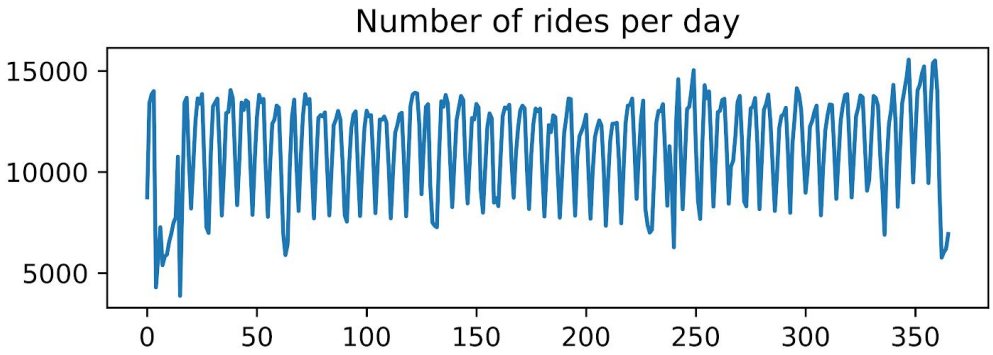




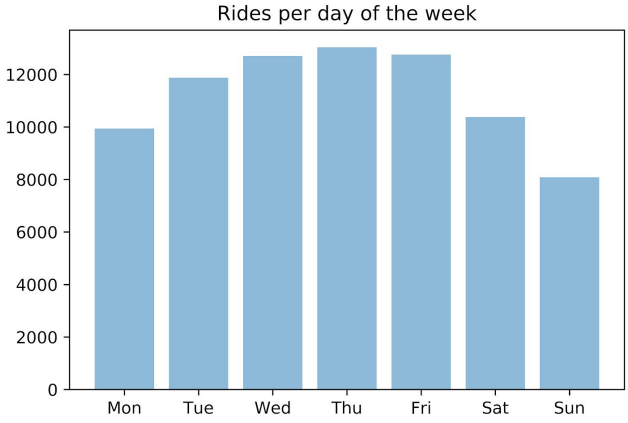
One of the things noticeable straight away, is the lack of certain places of high frequency use, and low frequency places to counter. We initially thought this would be the case. This stemming from our European expectations of a city with dedicated streets or districts where people use taxis. This being places for things like nightlife, entertainment and dining out. While in Taipei in particular, there isn't a real nightlife district. Restaurants can be found everywhere. And the thing that comes closest to an entertainment district in Taipei is Ximen, which is not in the Neihu district. We therefore dropped the idea of real 'hotspots' in our thinking.

Taking a look at the start location of the rides, we can see that some zones get significantly more customers (the urban southern zones) that others (the mountainous northern zones). We therefore we would use the zone id number as an input variable.

The next focus got laid upon time. The timestamp field in particular. We plotted amount of taxi rides per day unto the days in the year, giving us the following graph:

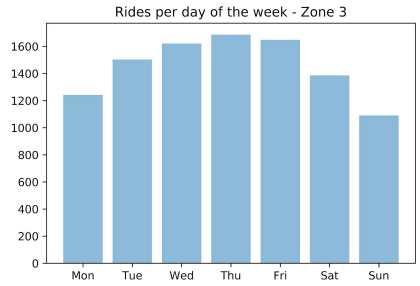


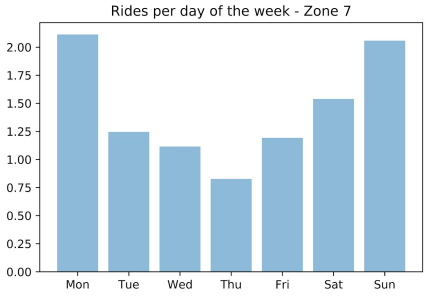
What stands out that there isn't a real big trend, it's more or less consistent for the better part of the year. The only noticeable difference being Chinese new year in the beginning of the year. The almost wave like pattern did make us suspicious day to day differences were present during the week. That's why we decided to plot this as well.



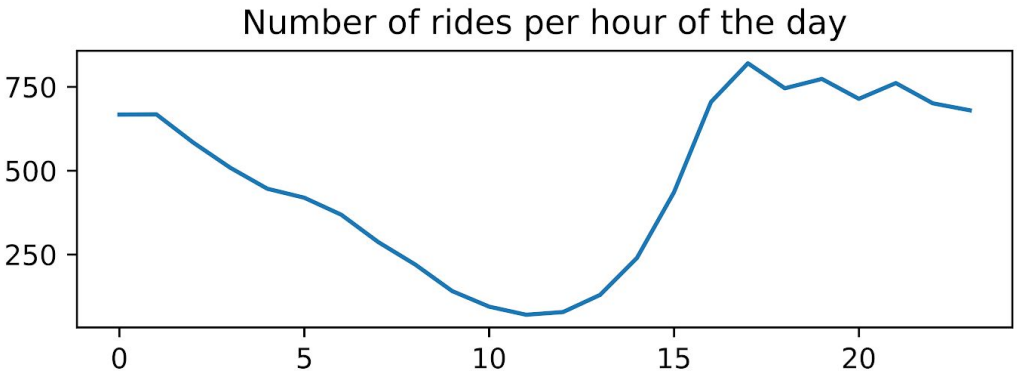
We can certainly see the wave like trend back in this graph. We initially suspected it being a different shape though. With peaks from friday to sunday, corresponding with weekend activities (going out, dating, family visits). But again it was our European expectations that led us down.

One thing to note though, is that not every zone is the same, some zones have very different shapes. This of course being down to the zone itself. The mountainous zones having more traffic during the weekend from people who go hiking or something like that. And the southern zones having more traffic akin to the figure above from people going to work and stuff like that. This can be seen in the following images, where we plotted traffic for zone 7 and 3.

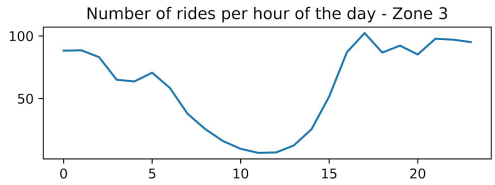


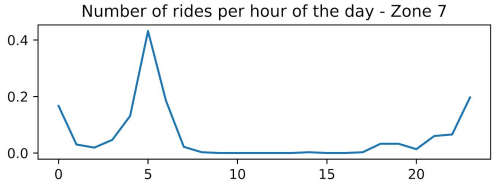


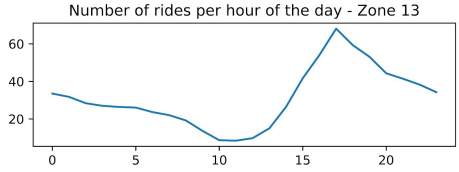
We then started to look at the hours in the day. We plotted the number of rides to the hours in the day. Giving us the following graph.



We can see a significant drop in the middle of the day, and a plateau from 17 to 2 o'clock. With a steady decline from 2 to 12 and a sharper rise from 12 to 17. We figured this decline is due to people being at work. Just as with days in the week, we saw different trend for different zones. This time we plotted zone 3, 7 and 13.







The variables we would end up using for our model are as follows:

- DayOfTheWeek
- HourOfTheDay
- Month
- ZoneID

-

5. Model preparation

We started by doing some research on big data and artificial intelligence. We ended up considering two types of AI. These being a support vector machine (SVM) and a decision tree algorithm. These were covered in the slides that the professor provided.

The SVM was eventually not chosen. This had 2 main reasons. The first being its inability of handling categorical variables, which is basically al we have so that makes it incompatible with our objective. The second reason why we didn't choose an SVM algorithm for our project, is that SVM makes a binary choice, yes or no. This is not what we required. The algorithm has to give an estimate for the amount of taxi riders for a given hour slot in a given zone. This means a continuous output variable is required, not a binary one.

We therefore started researching a decision tree algorithm in depth. A decision tree is a 'fancier' version of the normal binary tree. It starts off with all of the data at the top of the tree, and by making 'smart' decisions it subdivides the data into smaller sets of entries that have increasingly more homogeneous output variables.

The 'smart' decisions are made by looking by the amount of information gain per decision. Information gain is determined by looking at the subsets after a decision and what the average output variable is. When after a decision a subset has become so small, or all the elements have the same output variable, then the tree stops at point in the tree and make a 'prediction leaf'.

There were 2 different decision trees we could use, a classifier or a regressing decision tree. We first tried a classifier, this would be a mistake as we would find out in a later stage. We thought of the output as an integer value (or in other words, only round numbers), which sounded like a categorical value to us. And thus we thought we had to use a classifier. After getting really bad results we rethought our decisions and discovered our mistake. We therefore started using a regression decision tree. The regressor gives us a continuous output variable, which is exactly what we want.

A decision tree usually has a set number of maximum layers, this is to reduce computational time. The maximum number of layers of a tree is called the depth of the tree.

Decision tree algorithms usually employ what is called a forest. This is because a single tree can be overfitted, but if you have 100s of trees, this is much less likely. This algorithm gathers all the outcome from all the trees, and gives an average outcome as final output.

6. Results

We tried different layouts for the algorithm. We experimented with the amount of trees we would use, and the maximum depth of a tree. We measured the performance of the different setups by looking at the root mean square deviation (or RMSE) of the predicted hire count. We gathered all this data and made the following table:

No. of trees	Max depth	RMSE
101	5	21.925095
75	20	14.316507
200	12	14.597748
200	7	17.763558
100	15	14.421029
100	10	12.958575

Afterwards we tried a different approach. We didn't actually try to predict anything in this approach. We computed the average for every hour of the day and every day of the week, and just filled in the average. We did this because we saw that the amount of passengers just didn't jump all that much. The only exception being chinese new year, something which can be easily computed separately. The amazing (or kind of embarrassing) thing is, we outperformed our AI with this technique. We got RMSE as low as 10.95. Which, as of writing this document, the third of the competition overall.

7. Conclusions

Looking at the results of the regressor decision tree algorithm we've employed, we saw that a max depth of 7 or under just isn't going to cut it. We've got our best results a max depth of 10. And increasing the max depth after 10 gave us increasingly disappointing results. We figured this is due to overfitting.

The number of trees didn't give much improvement in performance. We think this is because the number of trees we'd be using is already after the 'tipping point'. Or in other words, that the major leaps in performance are already achieved with a lesser amount of trees. We didn't have the time to confirm this conclusion, but we guess that a tree number of around 40 to 70 trees is the sweet spot.

Reflecting on our performance and our project in general, we concluded that we had adequately achieved the goal of the project. While not an amazing score, with the material we were offered and our knowledge regarding the subject, we think there isn't a lot of room for improvement. We're only off by about 1.5 RMSE points compared to the current leaders of the leaderboard.

We think there are 2 ways to increase RMSE. The first one being using another type of algorithm to predict the amount of riders. This second one being to gather more features to use in the current decision tree algorithm. We think that to properly perform, the decision tree should have more low entropy features to it's disposal. In the slides we got from the professor, it's said that a decision tree algorithm should have

somewhere from 10 to 100 variables to work with. We are currently only using 4. With 1, the month variable, not being of particularly low entropy.

A different type of algorithm may be able to make a better prediction with the data currently available. Though are knowledge of AI and machine learning doesn't allow us to give well constructed suggestion.

8. Appendix

a. Data flow diagram