FIT5145 Introduction to Data Science - Assignment 4

PROJECT REPORT

Student Name - Simran Singh Gulati

Student ID - 31125301

Project Description

In this report, we propose a new project that aims at building an application that lets the user find the nearest *available* parking spot for their car.

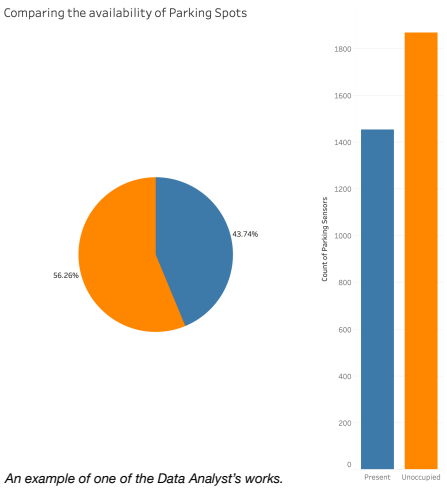
The services would be limited to the city of Melbourne due to the restricted data sources. Nonetheless, we extend the capabilities of our application by *computing real time probability of finding a parking spot* by considering the historical trends, and providing an estimate parking fee in that area. It encompasses a variety of factors like the parking fare, meter type, duration permitted, disability services, parking restrictions, etc.

Like any other data science project, our application will be built on the fundamentals of data availability and its appropriate utilisation. To our aid, Victorian government publicly shares the parking data through their Open Data portal where we could find a few good sources for our project. We observed four relevant datasets, one from the parking sensors installed in each parking bay, second the parking meter handling a group of parking spots in a particular street, third the parking restrictions specific to time and day of the week, and finally the shape file with geolocations which could be used to deploy a bird's eye view of the city in our application. Though all the datasets do not link with each other *directly* but I could find certain *common* attributes that will enable us to join every dataset with at least one other dataset. Ultimately, we can daisy chain these files to build one big relevant database.

All in all, our application tells the user where to park and how much would it cost to park. Plus the exploratory analysis of current parking trends could help the government to develop future infrastructures with reduced costs, while maintaining driver safety.

Coming to the associated data science roles, the Open Data portal is updated every two minutes, so we would primarily need a *Data Engineer* who would fetch information from the website as well as write a script to pull future data using their API. Thus he is required to get all the historical data as well as automate the process to add newer entries to our database. This database would be crucial in a multi faceted manner. Additionally, this person is required to look for more relevant resources as the more data we have the better we can build our model.

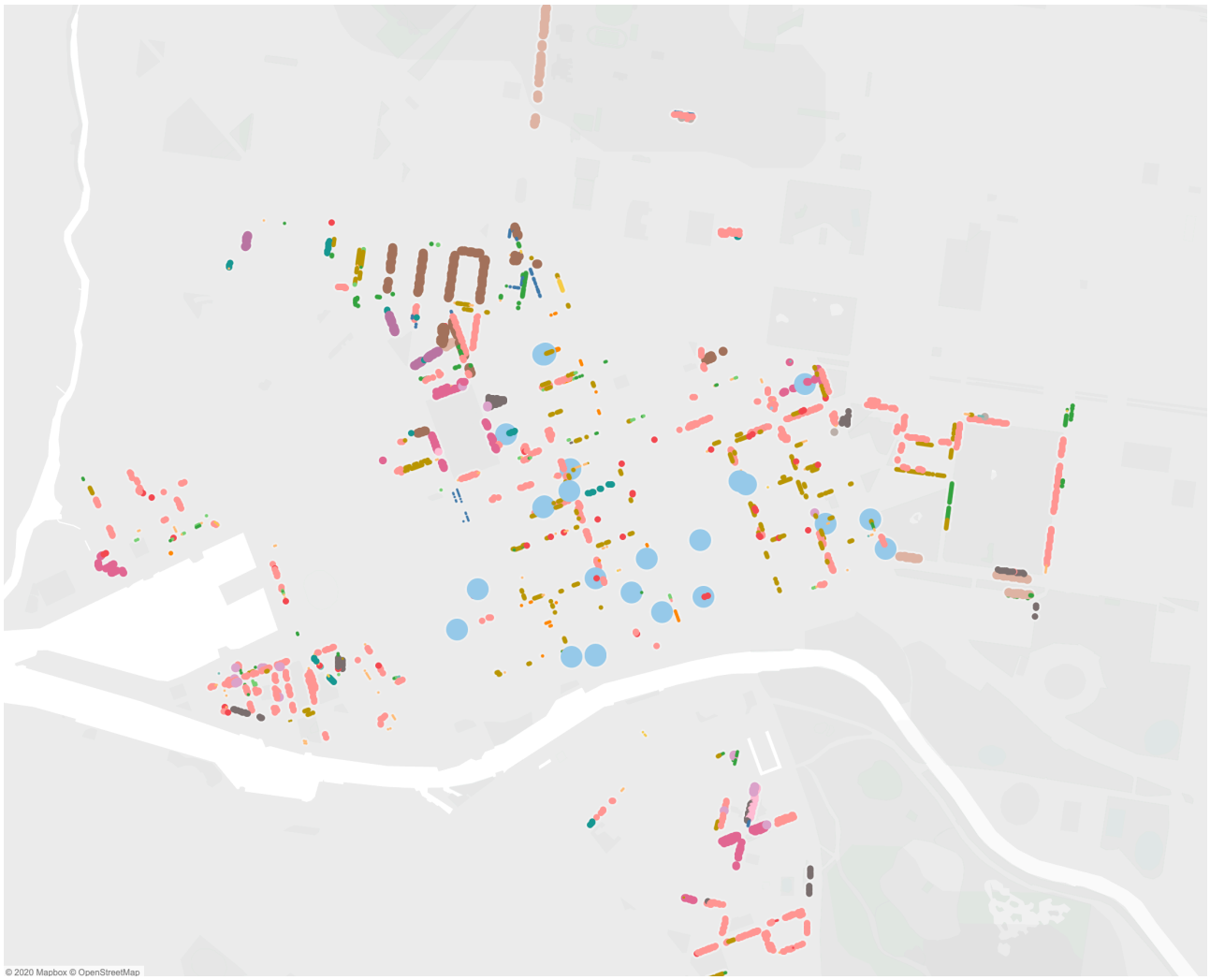
Next we need a *Data Analyst* who would explore and visualise excerpts from our database to build insights. Their task is to wrangle through data and transform it into a useable format, followed by presenting a report that shares their findings. For instance, (1) how does the availability compare across different days of the week or (2) what areas are more populated or (3) what is the parking fare in busier areas or (4) what streets have accessibility services, etc. He is not only required to generate insights from data but also to communicate them in an easily understandable manner to the audience, be it stake holders or the fellow developers.



An example of one of the Data Analyst's works.

An analyst's investigation would be put to use by statisticians or in layman terms, the *Head Data Scientist*. This person is required to make predictions in case all parking slots are unavailable. Hey may consider (1) what is the permitted duration in a particular bay, and (2) when was the first car parked in that area. Assuming, the driver wouldn't want to pay a fine, he's likely to vacate his spot within the permitted time. So the data scientist is required to build a model that predicts how soon can the user expect a bay to be vacated.

Lastly, we need a *Developer* with experience in software design and application development who can collaborate with all the above mentioned players and work in parallel to code a user ready application.



Note : There is no official source for these pictures as they have been produced by me using Tableau by importing the datasets mentioned above.

In this image the circles represent the parking spots across the city wherein the colour represents the *meter type* such as reserved for disabled drivers, no-parking zone, loading zone, etc. and the radius of circle is proportional to the time permitted in that zone.

Thus, the Engineers hunt and fetch the data. Analysts scrutinise the fetched data to gain insights and present visualisations to the Head Scientists, who use their statistical expertise to make predictive models. These people act as a link between developers and the stake holders.

Business Model

The *application* falls under the SaaS category, wherein we use publicly available data from government and mould it to build models that can recommend closest available on street parking to the users. More importantly, even if all the nearby parking spots are occupied, the historical trends and parking restrictions can be used to predict when is an occupied spot likely to be vacated. And gives the user an estimate parking fee in the area.

As a byproduct of this investigation we *can* also collaborate with government to analyse these trends in greater depth, to find if our city is well equipped to deal with the incoming flow of vehicles. Basically, it enables us to compare the demand for parking spots with their availability which can be used regulate parking restrictions, meter fee and even the parking fines. These three factors can be used to build strategies for newer parking facilities to ensure that there's regular turnover of vehicles with minimal exploitation of the public space. Being precise, the project on a whole can be categorised as **Answer as a Service** or commonly termed the AaaS.

An important aspect of our application builds around the information access to third parties. The data used to build models is unrestricted and available via government websites. Considering the privacy guidelines, parking sensors only reveal if a car is present in a bay or not, no other privy details are shared. Walking on these line, our application only reads the user's current location coordinates and uses on device Machine Learning algorithms to find him/her the nearest parking bay. Most importantly, the user specific location is never stored on our database which maintains user privacy as well as cuts down the processing costs.

Consequently, a well thought project lifecycle is critical to data management. Safeguarding the system from outsiders is *as important as* ensuring that there's no misuse of information during the Research and Development phase. Facts suggest that there's inadvertent exploitation of data when companies are focussed too much in achieving their goals tactically. We intend to incorporate these measures by bringing in the business team for strategic plan of action.

They will be working in parallel with the project lead to discuss legal and ethical concerns. To begin with, data collection seems fairly regulated as it is publicly accessible from the government *but* coming to the application development, we need to ensure that data is anonymised and stays within the workspace walls. It may sound confusing due to overlapping domains but let me explain in a structured manner:

- We want maximum information from the government data for rigorous analysis.
- We want minimum information about the user to maintain user integrity.

Regarding user data, we need information regarding how long do they plan to park for and what kind of vehicle are they in. Plus, for those with accessibility needs we may additional information for prioritised parking. And obviously, the user credentials, as some users may prefer the app to save default details to avoid everyday hassle.

Ensuring these attributes stay confidential is necessary and can be achieved through employee monitoring, whitelisting user access, restricted visibility and invigilated data extraction via a human in the loop along with strict regulatory policy.

Characterising the Data and Data Processing

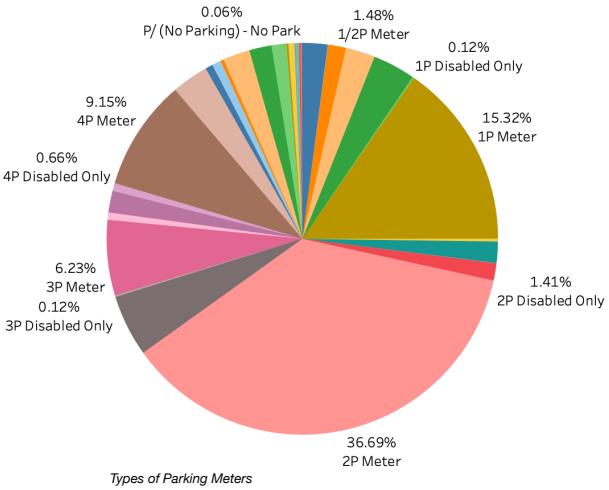
We collect data from *three* separate sources. Primarily we are interested in capturing information about the individual parking bay. Each bay is installed with an electronic sensor underneath which tells if a vehicle is parked in that bay or not. There's metadata attached with every sensor which reveals other crucial factors like location coordinates, unique bay id, governing parking meter etc. in the long form.

Next we talk about the rules in each parking street. Multiple sensors are coupled depending on their proximity and assigned to a parking meter.

Each parking meter contains parking restrictions for the bays associated with it.

The image on the right explains the various meter types where 1P means 1 Hour Parking, 2P denotes 2 Hour Parking etc. While 1/2P allows only 30 minutes, we can spot No Parking as well as Disabled Only meters as well.

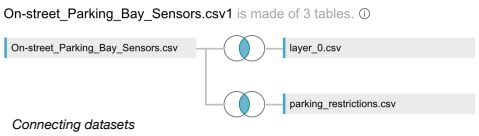
This picture is a general representation but in reality each each meter is coupled with up-to 6 restrictions (in wide form), depending upon the day and time of the week.



Types of Parking Meters

The above mentioned files can be combined using an inner join on the bay's unique id but that's the case for predictive modelling. This report is not limited to facilitate just the analysts but the developer working along side too. Thus we looked up another dataset and found a shape-file marking the parking streets in spatial polygons. The geotagged data would be helpful for generating application layout as well as aid the Data Scientist while delivering reports to the business team. The shape file can can be connected to the meters but not the *individual bays* which is alright in our case.

All in all, the bays can be linked to a meter (and associated restrictions) and the meters can be linked to spatial data. A significant challenge we may face is about missing data.



Connecting datasets

For instance, what if the duration is missing? We may check the meter name and/or type and predict this value in the wrangling phase. While most values can be imputed but *certain attributes like missing id or wrong location coordinates can never be fixed.* For now we do an intersection to eliminate incorrect bay id but we discuss about other attributes later in the report (Data Analysis section).

In conclusion, we have (1) data from sensors which is more likely to be correct and structured and (2) human generated restrictions table which *may* be relatively hard to parse due to typing errors and lastly (3) a shape-file which is assumed to be accurate because it would be impossible to fix otherwise. Thus despite being in tabular format, we have **variety** of data — numerical, textual as well as spatial; potentially due to the varied sources it comes form.

In totality, sensor data (temporal) being electronically generated is less likely to be faulty and shape-file has to be assumed to be error free but the *restrictions table raises a lot of concerns*. Well, human typed data is anticipated to have some (if not a lot) uncertainty. More importantly it contains a mix of different fields including numbers, characters, variables as well as location coordinates. And must have been curated by multiple personnels which makes it more obvious to have contrasting formats (example - date format, terminology, abbreviations) and chances of being missing altogether. Thereby we have to be careful with this semi structural dataset, considering its **veracity**.

As pointed out in the beginning, the sensor's data is updated every minute and would have been fetched accordingly. Thereby making it **voluminous** as well as high **velocity** stream. In contrast, the other two datasets are relatively smaller in size and also static.

Considering growth laws and dynamic nature of our data, we need some sophisticated technologies like NoSQL to manage the high inflow of data. While two of our three datasets (restrictions and shape-file) can be stored on hard disk, real time data from sensors demands faster access which can be achieved using stream processing. And having the option to store intermediate processing like caching (but at larger scale) would result in higher on device performance for the users.

Acknowledging the above mentioned reasons, we suggest use of Apache Spark for our project. In addition to map-reduce methodology, Spark also ensures *fairly quick* access via in-memory processing. And is compatible with both Python and R.

Resources

We have four *major* domains to look resources for, including but not limited to:

1. Data Engineering — Tools to collect, store and manage big data.
2. Data Analytics — Methods to spot and decode trends in the data.
3. Predictive Modelling — Technologies to learn from insights and make predictions.
4. Application Development — To integrate AI workflows into client side application.

Although R is a pretty good platform for statistical measures but our project aims at much broader goals, thus we use Python as primary language for programming and querying needs. Moreover versatility of Python encompasses every domain mentioned above — Spark for data collection and processin, Pandas for Data Analytics, SciKit Learn for Machine Learning and Flask for developing AI enabled user application.

Coming to the softwares that would allow us to deploy these technologies, we will be using Jupyter Notebook for retrieving data from APIs, performing in depth statistical analysis and building a predictive model using Neural Networks. While an application can be built within an Anaconda enabled Jupyter Notebook, we prefer using PyCharm for our Flask project for the ease of debugging in an interactive IDE. Additionally we use Sublime Text as our *go-to text* editor for quick testing of scripts.

As you may have seen our bottom-up approach with respect to project specifications, we finally comment on the Operating Systems relevant to our project. As we are handling *rapidly changing voluminous data* we would need access to a unix based terminal for quick and dynamic parsing of huge datasets. Although it is achievable via the use of Cygwin for Windows, we recommend macOS or Linux based computers for their broader community support and tight integration with the hardware.

The data for our project is publicly available at Victoria Government's Open Data Portal:

1. Data from *Sensors* in each bay
   https://data.melbourne.vic.gov.au/Transport/On-street-Parking-Bay-Sensors/vh2v-4nfs

2. Parking *Restrictions* for each Meter
   https://data.melbourne.vic.gov.au/Transport/On-street-Car-Park-Bay-Restrictions/ntht-5rk7/data

3. Street polygons in the form of Shapefile
   https://data.melbourne.vic.gov.au/Transport/On-street-Parking-Bays/crvt-b4kt

Data Analysis

We begin by inspecting the columns and their data types for our *csv* files one by one.

```
> sensors_data <- read.csv("On-street_Parking_Bay_Sensors.csv")
> glimpse(sensors_data)
Rows: 3,567
Columns: 6
$ bay_id       <int> 1270, 6840, 5074, 1690, 4315, 7873, 8913, 1315, 7052, 6660, 1121, 1287, 5135, 3201, 1280, 9006, 2409, 2570, 1288, 6307, 1285, 1726, 3371, 8949,…
$ st_marker_id <chr> "4462E", "14040E", "9854N", "2240N", "C6966", "4472EA", "18000E", "1315W", "4664Ea", "13415W", "C1158", "4479W", "9948N", "5457W", "4472E", "C7…
$ status       <chr> "Present", "Unoccupied", "Unoccupied", "Unoccupied", "Unoccupied", "Unoccupied", "Unoccupied", "Unoccupied", "Unoccupied", "Present", "Present"…
$ location     <chr> "(-37.813703354077106, 144.95461916833347)", "(-37.796658516509055, 144.95814457090097)", "(-37.82520231330287, 144.9698326288035)", "(-37.8162…
$ lat          <dbl> -37.81370, -37.79666, -37.82520, -37.81628, -37.80939, -37.81308, -37.80341, -37.81708, -37.80570, -37.81966, -37.81375, -37.81296, -37.82591, …
$ lon          <dbl> 144.9546, 144.9581, 144.9698, 144.9599, 144.9562, 144.9543, 144.9601, 144.9587, 144.9595, 144.9498, 144.9601, 144.9540, 144.9674, 144.9544, 144…
> |
```

Sensor Generated Data

```
> restrictions_data <- read.csv("On-street_Car_Park_Bay_Restrictions.csv")
> glimpse(restrictions_data)
Rows: 4,518
Columns: 62
$ BayID           <int> 3101, 4919, 6664, 4972, 8634, 3108, 8167, 8587, 3103, 3928, 8498, 3105, 7978, 8099, 4893, 3947, 8083, 6849, 3106, 7981, 7966, 8152, 8089,…
$ DeviceID        <int> 27429, 21769, 21773, 21813, 28003, 26925, 25546, 27956, 27141, 27594, 27876, 24680, 25357, 25478, 21756, 24656, 25462, 24688, 24872, 2536…
$ Description1    <chr> "1/2P M-SUN 7:30AM-11PM", "2P MTR M-F 7:30-18:30", "2P MTR M-F 7:30-18:30", "1P MTR M-F 7:30-18:30", "2P RPA 1B 7:30-23:00 M-SUN", "1/2P …
$ Description2    <chr> "", "", "", "", "", "", "", "", "", "4P MTR SAT 7:30-12:30", "", "", "", "", "", "4P MTR SAT 7:30-12:30", "", "4P TKT A SAT 7:30-12:30", …
$ Description3    <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "1P SUN 7:30-18:30", "", "2P SUN 7:30…
$ Description4    <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
$ Description5    <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
$ Description6    <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
$ DisabilityExt1  <int> 60, 240, 240, 120, 240, 60, 120, 240, 60, 480, 240, 60, 240, 240, 240, 480, 120, 480, 60, 240, 240, 120, 240, 240, 240, 0, 120, 60, 240, …
$ DisabilityExt2  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, 480, NA, NA, NA, NA, NA, 480, NA, 480, NA, NA, NA, NA, NA, NA, NA, 120, NA, 240, NA, NA, NA, NA, 240,…
$ DisabilityExt3  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 120, NA, 240, NA, NA, NA, NA, 120, NA…
$ DisabilityExt4  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ DisabilityExt5  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ DisabilityExt6  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ Duration1       <int> 30, 120, 120, 60, 120, 30, 60, 120, 30, 240, 120, 30, 120, 120, 120, 240, 60, 240, 30, 120, 120, 60, 120, 120, 120, 15, 60, 30, 120, 120,…
$ Duration2       <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, 240, NA, NA, NA, NA, NA, 240, NA, 240, NA, NA, NA, NA, NA, NA, NA, 60, NA, 120, NA, NA, NA, NA, 120, …
$ Duration3       <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 60, NA, 120, NA, NA, NA, NA, 60, NA, …
$ Duration4       <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ Duration5       <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ Duration6       <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ EffectiveOnPH1  <int> 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1,…
$ EffectiveOnPH2  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, 0, NA, NA, NA, NA, NA, 0, NA, 0, NA, NA, NA, NA, NA, NA, NA, 0, NA, 0, NA, NA, NA, NA, 0, NA, NA, NA,…
$ EffectiveOnPH3  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 0, NA, 0, NA, NA, NA, NA, 0, NA, NA, …
$ EffectiveOnPH4  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ EffectiveOnPH5  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ EffectiveOnPH6  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ EndTime1        <chr> "23:00:00", "18:30:00", "18:30:00", "18:30:00", "23:00:00", "23:00:00", "23:00:00", "23:00:00", "23:00:00", "18:30:00", "23:00:00", "23:0…
$ EndTime2        <chr> "", "", "", "", "", "", "", "", "", "12:30:00", "", "", "", "", "", "12:30:00", "", "12:30:00", "", "", "", "", "", "", "", "19:30:00", "…
$ EndTime3        <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "18:30:00", "", "18:30:00", "", "", "…
$ EndTime4        <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
$ EndTime5        <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
$ EndTime6        <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
$ Exemption1      <chr> "", "", "", "", "Resident Permit Area 1B", "", "", "Resident Permit Area 1B", "", "", "Resident Permit Area 1B", "", "", "", "", "", "", …
$ Exemption2      <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
$ Exemption3      <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
$ Exemption4      <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
$ Exemption5      <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
$ Exemption6      <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ FromDay1        <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…
$ FromDay2        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, 6, NA, NA, NA, NA, NA, 6, NA, 6, NA, NA, NA, NA, NA, NA, NA, 6, NA, 1, NA, NA, NA, NA, 1, NA, NA, NA,…
$ FromDay3        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 0, NA, 0, NA, NA, NA, NA, 0, NA, NA, …
$ FromDay4        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ FromDay5        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ FromDay6        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ StartTime1      <chr> "07:30:00", "07:30:00", "07:30:00", "07:30:00", "07:30:00", "07:30:00", "07:30:00", "07:30:00", "07:30:00", "07:30:00", "07:30:00", "07:3…
```

Parking restrictions typed by humans

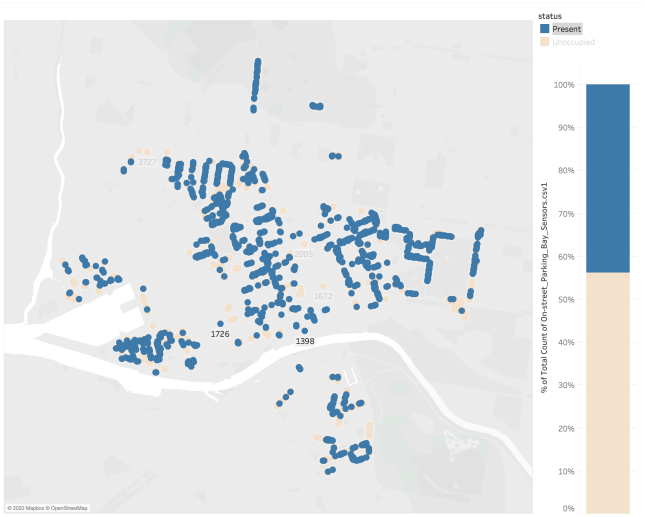
Exploring the first dataset we encounter close to 3600 sensors, each with 6 significant attributes. For each row we have *more than 1* unique identifier — Street Marker ID as well as the Bay ID. Now there's a bit of discussion on which is more suitable to be retained, while we drop the other as there's no point keeping redundant information in our *targeted warehouse.* To start with, unlike Street Markers, Bay ID is a numerical sequence and can also be used to link other datasets in our project.

Secondly, our aim is to generate a model that builds upon the fact — if a parking bay is occupied or not. Right now the *status* column says if a car is "Present" or the bay is "Unoccupied", so we typecast these categorical values into numerical labels (0 and 1) using one-hot encoding or dummies (Pandas). Next we check for redundant or missing values. While this dataset is electronically generated such issues are less likely to exist but we expect discrepancies in the second dataset which has been set up manually.

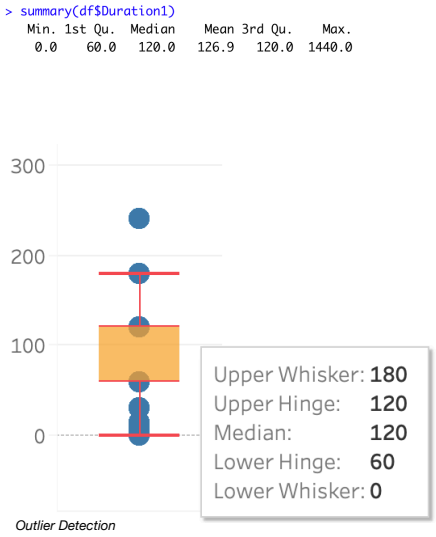
As evident from second picture, it contains a lot of empty values. Note, the empty values do not say that the data is missing. In fact, each bay can have *up to 6* restrictions but it does not mean each one is bound to have multiple restrictions. Most of the sensors, specifically those in the *less populous* areas have *at most one* restriction throughout. A contrasting example could be a busier area like Flinders where the permitted duration changes depending upon the time and day of the week. As weekends tend to experience less traffic in work oriented areas but heavy in flow at the leisure spots like clubs or restaurants. Nonetheless it would be better to replace those empty values with NA.

All in all, the data needs to be wrangled in depth and the above mentioned facts are just trigger points. This followed by a detailed exploration would generate insights for the analysts which would be helpful in pitching updates to the stakeholders and investors of our application.

The image below is one such example, where the analyst can select occupied/ unoccupied parking bays and *correlate* other factors like time, location, vehicle type, etc.



Comparing availability of Parking Bays



Outlier Detection

Combined with statistical tests like quantile functions for outlier detection, this information would be handed over to the Data Scientist who would then decide the appropriate mathematical algorithm to build a predictive model.

At preliminary stage one would see this is as a classification problem where *logistic regression* could be used to tell the chances of a spot being vacated in next 10 minutes. This is just an example and the time frame could be extended or reduced depending upon user preference as well as the *busyness* in that area.

Considering the diversity of attributes I would propose using Long Short Term Memory (LSTM) network, an advanced Recurrent Neural Network which takes into account multiple past events and makes decision accordingly. In our case I would feed it with the record of the last 5 vehicles in that spot and let the computer learn from past behaviour to make a prediction when the current car (parked in that bay) is expected to leave.

References

1. OpenData Portal(2020). *Parking Data.* Retrieved from https://data.melbourne.vic.gov.au/browse?q=parking&sortBy=relevance

2. VicRoads(2020). *Parking: Vic Roads.* Retrieved from https://www.vicroads.vic.gov.au/safety-and-road-rules/road-rules/a-to-z-of-road-rules/parking

3. City of Melbourne(2020). *Parking Rules.* Retrieved from https://www.melbourne.vic.gov.au/parking-and-transport/parking/parking-rules/Pages/parking-rules.aspx

4. ABC News (2018). '*Major Changes' to Melbourne CBD car usage needed, warns council parking report.* Retrieved from https://www.abc.net.au/news/2018-06-21/melbourne-city-council-considers-sweeping-changes-to-push-cars/9892998

5. Wikipedia (2020). *Demography of Australia.* Retrieved from https://en.wikipedia.org/wiki/Demography_of_Austra

6. Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.W.-K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

Presentation

Link : https://www.youtube.com/watch?v=FFoumJDrNVY