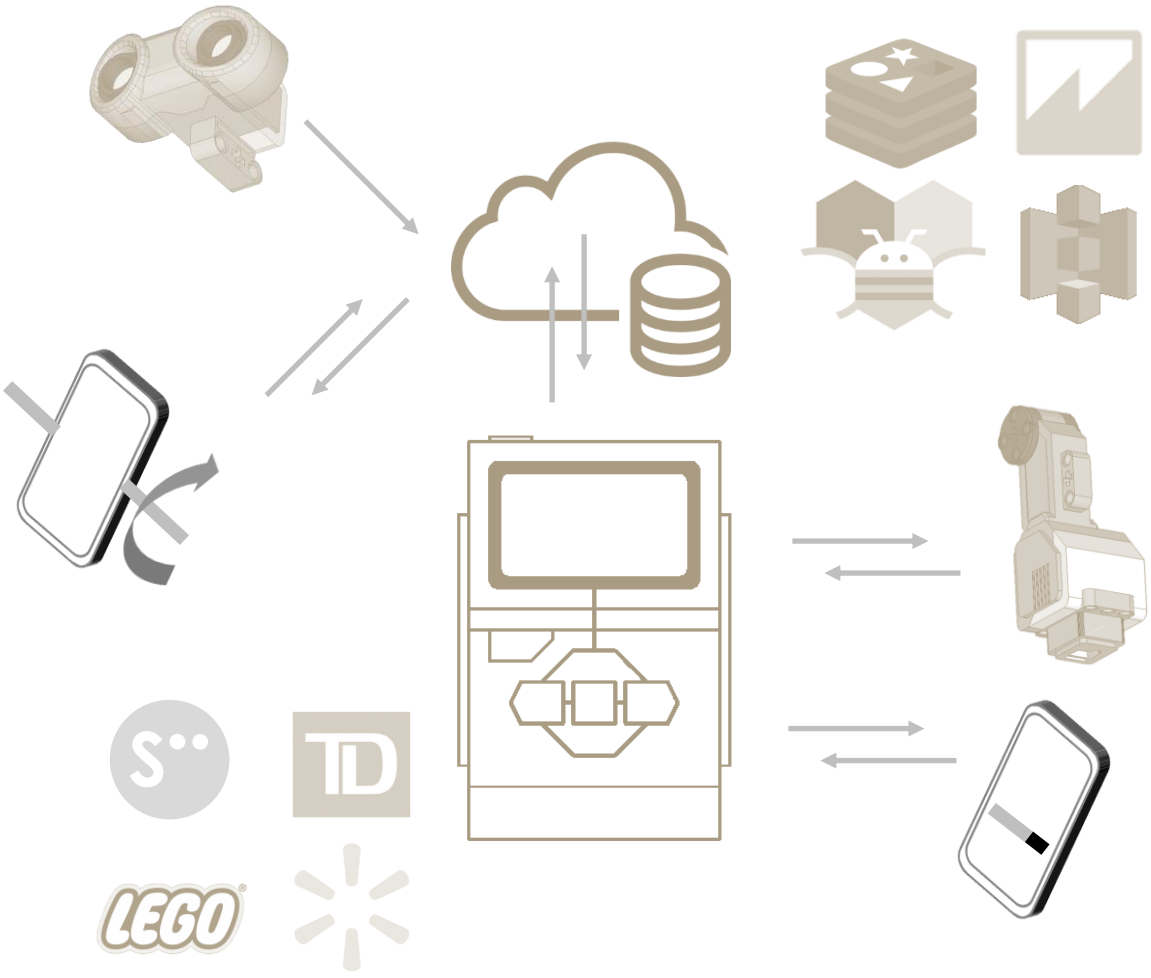
DATA SCIENCE PORTFOLIO



FARIUS TJIOESMAN

February 2nd 2021

FARIUS TJIOESMAN

DATA ANALYST, DATA SCIENTIST

SUMMARY

Data Analyst, Data Scientist and Project Manager with years of experience in retail to power sectors, specializing in building complex model and continuous improvement. Highly effective building long-term relationships with both internal and external customers by being a team player and flexible, delivering results and leading change initiatives that drive the business to grow, enhance the customer experience, meet the objective of the organization.

DATA SCIENCE PROJECTS

- Bell, Indeed and Monster web scarping with subplot, S3, NLTK
- Loan Default with Scikit Learn Decision Tree, Random Forest, KMeans, PCA, Lime, Shap
- Tableau Flow Visualization, RSS Contour with Plotly Scatter 3D
- Mortgage Machine Learning with MLPClassifier and Keras
- Autonomous Driving with Redis Big Data, AI, Lego, S3, Athena, Quicksight, databricks SQL, ML
- SCM and CRM strategy and Azure Sand Dance 3D Stacks

SKILLS

SQL Server, AWS Athena, Redis, Neo4J, SAP, JDE, Salesforce
PowerBI, Tableau, Matplotlib, Seaborn, Sankey, Sand Dance
Python (Anaconda, PySpark), Excel Macro, VBA, C++, Java

WORK EXPERIENCES

Data Science Consultant – Beam Data
September 2020 – present, Toronto, ON
- Developed AI demand requirement planning, sankey charts
- Summarized daily job description with NLP and saved into S3
- Web scraped talent pages and saved the results into Neo4J
- Visualized outlier with boxplot and correlation with heatmap
- Extracted SAP data from SQL Server, used LSTM for prediction
- Utilized pandas to analyze, HTML IPython.core.display for jpg
- Enabled hyper parameter γ slider with ipywidget interaction

Territory Manager – Mersen
April 2011 – July 2020, Toronto, ON and Vaudreuil-Dorion, QC
- Predicted, grew 10% of Eastern Canada sales with big data
- Assisted computational fluid dynamic, finite element analysis

Mechanical Engineer – General Electric
June 2008 – September 2010, Oakville, ON
- Applied VBA, Excel macro, what-if analysis, proposed kaizen

Project Manager – Schneider Electric
November 2004 – March 2008, Batam, Indonesia
- Built factories, used SAP, WBS and gantt chart for monitoring

EDUCATIONS

- Diploma in Data Science, WeCloudData, Toronto, 2021
- Professional Engineer, Professional Engineers Ontario, 2014
- Diploma in Engineering Software, Humber College, 2011

TABLE OF CONTENT

SAKILA

MYSQL BASIC QUERY

As part of early week exercise we did few SQL practices including Sakila Movie database a fictional company. There are couple dozen exercises but only a few I took here from basic ones to nested query, window function

rating	MaxRate
R	4.99
PG-13	4.99
PG	4.99
NC-17	4.99
G	4.99

```
-- Query from film table: largest rent_rate for each rating
SELECT rating, max(rental_rate) AS MaxRate
FROM film
GROUP BY rating
ORDER BY MaxRate DESC;
```

rating	numFilms
PG	194
G	178
NC-17	210
PG-13	223
R	195

```
-- Query from film table seeking number of film for each rating
SELECT
    rating,
    count(*) AS numFilms
FROM film
GROUP BY rating;
```

short	standard	Long
96	438	466

```
-- Query from film table, simple aggregation based on movie length
SELECT
 sum(case when length<60 then 1 else 0 end) as short,
 sum(case when length>=60 and length<120 then 1
      else 0 end) as standard,
 sum(case when length>=120 then 1 else 0 end) as 'long'
FROM film;
```

lastNameJohansson
MATTHEW JOHANSSON
RAY JOHANSSON
ALBERT JOHANSSON

```
-- Query from actor table to select actor with last name Johansson
SELECT concat(first_name, ' ',last_name) AS
lastNameJohansson
FROM actor
WHERE last_name='Johansson';
```

lastNameDistinct
MATTHEW JOHANSSON

```
-- Query from actor table to count distinct Last Name
SELECT count(DISTINCT last_name) AS lastNameDistinct
FROM actor;
```

	LastName	appear
1	ASTAIRE	1
…	…	…
66	WRAY	1

```
-- Query from actor table seeking for Last Name appear once
SELECT distinct(last_Name),
       count(*) AS appear
FROM actor
GROUP BY last_name
HAVING appearance = 1;
```

	LastName	appear
1	AKROYD	3
…	…	…
55	ZELLWEGER	3

```
-- Query from actor table for Last Name appear more than once
SELECT distinct(last_Name),
       count(*) AS appear
FROM actor
GROUP BY last_name
HAVING appearance > 1;
```

SAKILA MOVIE DATABASE

more query

Title	N_ACTORS	CUMM
ACADEMY DINOSAUR	10	10
…	…	…
ZORRO ARK	3	5462

id	NAME	N_FILMS
107	GINA DEGENERES	41
	…	…
148	EMILY LEE	14

film_id	actor_id	title	ACTOR
1	10	ACADEMY DINOSAUR	CHRISTIAN GABLE
1	40	ACADEMY DINOSAUR	JOHNNY CAGE
1	20	ACADEMY DINOSAUR	LUCILLE TRACY
…	…	…	…
186	12 1	CRAFT OUTFIELD	LIZA BERGMAN

film_id	CATEGORY	TITLE	RENT_FREQ	RANKING
584	Foreign	MIXED DOORS	4	1
…				

```
-- Query from actor table to identify number of actors
SELECT
       title, NUMBER_ACTORS,
       sum(NUMBER_ACTORS) over (order by title) AS CUMM
FROM (  SELECT
            film_actor.film_id, film.title,
            count(film_actor.actor_id) AS N_ACTORS
         FROM film_actor
         LEFT JOIN film
         ON film_actor.film_id = film.film_id
         GROUP BY film_actor.film_id
         ORDER BY NUMBER_ACTORS DESC
     ) AS t;

-- Query to find how many films each actor played in
SELECT
 DISTINCT film_actor.actor_id AS id,
 CONCAT(actor.first_name, ' ', actor.last_name) AS NAME,
 COUNT(film_id) AS N_FILMS
FROM film_actor
LEFT JOIN actor
 ON film_actor.actor_id = actor.actor_id
GROUP BY film_actor.actor_id
ORDER BY NUMBER_FILMS DESC;

-- In table Film_actor, there are actor_id and film_id columns. Need
-- to know actor name for each actor_id, film tile for each film_id.
SELECT film_actor.film_id, film_actor.actor_id,
       film.title,
       CONCAT(actor.first_name, ' ', actor.last_name) AS
          ACTOR_NAME
FROM film_actor
LEFT JOIN film
ON film_actor.film_id = film.film_id
LEFT JOIN actor
ON film_actor.actor_id = actor.actor_id
ORDER BY title, ACTOR_NAME ASC;

-- Rank the movie rent frequency to get rid of unpopular titles.
SELECT
  film_id, CATEGORY, TITLE, RENT_FREQ,
  DENSE_RANK() OVER (ORDER BY RENT_FREQ ASC) AS RANKING
FROM
(SELECT
  DISTINCT film.film_id, category.name AS CATEGORY,
  film.title AS TITLE, COUNT(rental.rental_date) AS
  RENT_FREQ
FROM rental
LEFT JOIN inventory
ON rental.inventory_id = inventory.inventory_id
LEFT JOIN film
ON inventory.film_id = film.film_id
LEFT JOIN film_category
ON film.film_id = film_category.film_id
LEFT JOIN category
ON film_category.category_id = category.category_id
GROUP BY film_id
ORDER BY RENT_FREQ ASC) AS t;
```

BREAST CANCER

Likelihood with K-Nearest Neighbor (KNN)

The idea would be applying Pythagorean Theorem on Euclidian n-space as found below with below equation https://en.wikipedia.org/wiki/Pythagorean_theorem

$$\sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_n - b_n)^2} = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}.$$

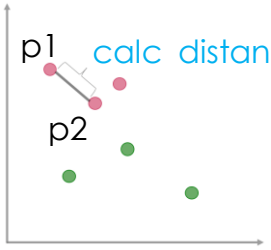
KNN Maths

```
from sklearn.datasets \
      import load_breast_cancer
from math import sqrt

datasets = load_breast_cancer()
X = datasets['data']
y = datasets['target']
```

This code calls example of breast cancer dataset from 569 patients where 30 different features were recorded for each. The variable datasets has a type of sklearn.utils.Bunch which like a dictionary with seven pair of keys and values. Only two of them will be used in this exercise, data and target which are store into X and y variables.

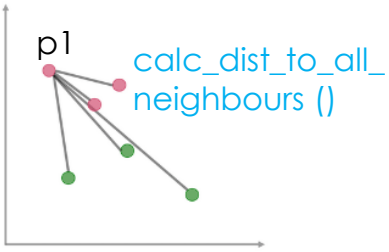
```
def calc_distance(p1,p2):
    squared_difference = 0
    for i in range(X.shape[1]):
      squared_difference +=\
        (p1[i] - p2[i])**2
    return \
      sqrt(squared_difference)
```

This function calculate the distance between p1 and p2 says p1 = X[0], p2 = X[1] which are the first and second patients.



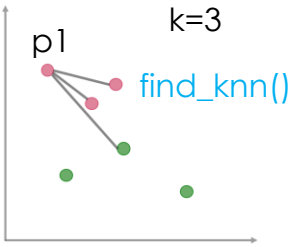

```
def
calc_dist_to_all_neighbours(p1):
 dist_to_all_neighbours = []
 for i in range(X.shape[0]):
  dist = calc_distance(p1,X[i])
  dist_to_all_neighbours.\
       append((i,dist))
 return dist_to_all_neighbours
```

The next function would be listing nested-list of its neighbors and it distances from the reference patient says p1. There would be 569 pairs including p1 distance to itself for this case.




```
def find_knn(p1,k):
  dist_to_p1 =
  calc_dist_to_all_neighbours\
  (p1)
  dist_to_p1.sort(\
    key= lambda x: x[1])
  return dist_to_p1[1:k+1]
```

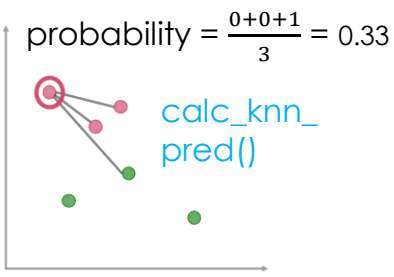
In term of KNN context this is what k stands for, the number of nearest neighbors. The function identifies all points and their distances and later sorted ascendingly and took the first k neighbors to display.



KNN Prediction

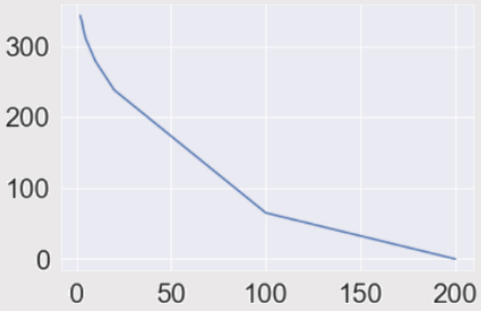
```
def calc_knn_pred(p1, k):
  nn = find_knn(p1, k)
  nn_indices = [x[0] for x in nn]
  nn_cancer_ind = \
    [y[i] for i in nn_indices]
  knn_pred_prob = \
    mean(nn_cancer_ind)
  knn_prediction = 1 if \
    knn_pred_prob >= 0.5 else 0
  return knn_pred_prob,\
         knn_prediction,\
         nn_cancer_ind
```

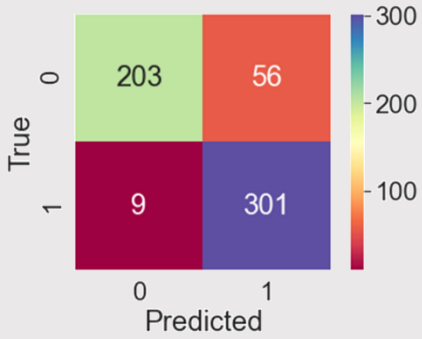
Taking distances of k nearest neighbors the prediction was made positive (1) when the probability is bigger than ½



BREAST CANCER LIKELIHOOD

KNN Verification





	precision	recall	f1-score	support
0	0.96	0.78	0.86	259
1	0.84	0.97	0.90	310

In term of KNN model above, we probably interested to know the decision of judging mean of 0.5 into positive cancer by comparing the whole 569 patient prediction into its target values (truth data).

Determining k value is also a key factor. As k is bigger the prediction worsen as the number of positive case depletes

```
import matplotlib.pyplot as plt
knn={}
for k in [2,3,4,5,10, 20, 100, 200]:
    prediction_list = []
    for patient in range(X.shape[0]):
        probability, prediction, indices =
calc_knn_pred(X[patient], k)
        prediction_list.append(prediction)
    knn[k]=sum(prediction_list)
plt.plot(list(knn.keys()), list(knn.values()));
```

adding few more library imports from sklearn we could verify KNN prediction using confusion matrix and classification report as below

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import\
     classification_report
from sklearn.metrics import plot_roc_curve
import seaborn as sns
import numpy as np

cm = \
   confusion_matrix(np.array(prediction_list),\
   np.array(y.tolist()))
ax = sns.heatmap(cm, cmap='Spectral',\
   annot=True, fmt='d',square=True)
sns.set(font_scale=2)
ax.set_xlabel('Predicted')
ax.set_ylabel('True');

print(classification_report(np.array(\
   prediction_list), np.array(y.tolist())))
```

Out of 560 patients 203 actually negative were predicted correctly having no cancer. 301actually positive were also predicted correctly having cancer. 56 actually had no cancer were identified with one that add unnecessary anxiety, cost. 9 positive cases were wrongly identified early that may cause death, question to the prediction model.

BELL MOBILITY

A week before batch 10 immersive Data Science bootcamp started in September 2020 at WeCloudData we were all assigned for an icebreaking regarding a topic of our choice towards customer journey presentation during this preparation week. My team agreed to pick a mobile phone company Freedom Mobile for this topic with frequent comparison with Bell Mobility as one of its major competitor.

Lexie Li, Manny Brar and myself did a great presentation and I took an opportunity to web scrap either Freedom or Bell website showing variety of latest device offer to lure new customer to join or keep the same customer from churning. I pick its competitor Bell due to its website simplicity compared to Freedom by using BeautifulSoup. After week 3 I got a great idea using Selenium which help me to webscrap Freedom website.

Therefore I presented almost similar web scraping from Bell twice (at week 0 and week 3) where I showed the differences, improvement made as I learned a lot more python functionalities that I could quickly apply to Bell.

There are at least five different codes I would like to share. The first one bell.ipynb was related to week 0 presentation while the next three related to week 3 presentation and the last one shows an idea of using Selenium with Mozilla Firefox browser to webscrap Freedom Mobile.

References:

First Presentation
https://github.com/FariusGitHub/DataScience/blob/master/bell.ipynb

Visualization with Subplots
https://github.com/FariusGitHub/DataScience/blob/master/bell2.ipynb

Sample code with tqdm, mix of iteration and Data Frame
https://github.com/FariusGitHub/DataScience/blob/master/bell3.ipynb

Comparing the one above and with Data Frame only
https://github.com/FariusGitHub/DataScience/blob/master/bell4.ipynb

Practical code for Freedom Mobile Webscrap
https://github.com/FariusGitHub/DataScience/blob/master/selenium.ipynb

BELL MOBILITY

The Motivation

CLOUD-BASED WEBSCRAPING

Out of complexity of machine learning, deep learning and regular data scientist tasks I still believe there are a lot of situation where we could share comprehensive analysis from live or up-to-date streaming data where access to descent computer may not always available. Take two different scenarios where family and friends gathering for a TV time or at a busy restaurant where the only devices available are Nvidia Shield TV console and smartphones.

Without need to run the code locally when webscraping the solution would be cloud-based webscraping such as using Kaggle or Colab. They are also a powerful tool to show our portfolios other than webscraping. I usually share my code and non-confidential data from GitHub.

CONTINUOUS IMPROVEMENT

For the first week we went through MySQL and second week for python, pandas, numpy, seaborn, matplotlib and string manipulation where I found opportunity to add features below to my previous code. Using Data Frame only (no more 'for' loops) it shows my code run 5X faster to visualize Bell latest devices from the web which is around four dozens of them time to time.

Python topics covered at this portfolio during optimization

- BeautifulSoup and Selenium API
- List and Dictionary Comprehensions
- Iteration, TQDM and Enumerate
- String Manipulations (Strip, Split, Array, Index)
- Panda
- Lambda Function
- Matplotlib with pyplot, subplot

BELL MOBILITY

The Website

DEVICE RETURN OPTION

Samsung Galaxy S20 FE 5G

From
$0.00 down | $15.84/mo. for 24 months | 0% APR

Requires an eligible 2-year rate plan with Device Return Option

Device full price: $1,200.00

DEVICE RETURN OPTION

Samsung Galaxy S20 Ultra 5G

From
$0.00 down | $32.67/mo. for 24 months | 0% APR

Requires an eligible 2-year rate plan with Device Return Option

Device full price: $2,300.00

DEVICE RETURN OPTION

iPhone 11

From
$0.00 down | $28.55/mo. for 24 months | 0% APR

Requires an eligible 2-year rate plan with Device Return Option

Device full price: $875.00

iPhone XR

From
$0.00 down | $30.00/mo. for 24 months | 0% APR

Requires an eligible 2-year rate plan

Device full price: $720.00

There are around ten brands of mobile phones offered by Bell week by week, month by month in 2020 ranging from Samsung, Apple, LG, Google, Huawei, TCL, Motorola, Alcatel, ZTE and Doro from the combination of 50 post-paid and pre-paid phones. 10% of those phones are pre-paid and the full price on these devices are not available.

Compared to Freedom Mobile website it was obvious that Bell tries to catch niche market (even make their website easier to scrap, in my opinion). Both try to price their device competitively but what the difference is monthly bill where Bell generally is more expensive probably due to their large number of its clients from corporate customer, Bell therefore has larger business cost (overhead) and also tend to have larger profit margin too (if managed right).

There are subsequent information such as each phone specification, loan details, etc but I limit the scraping project based on three important information available just from one page rather than looping to the index page and visiting subsequent pages seeking for more details. Those three features that I am looking at are : phone names, phone images and phone (full) prices.

The website also offered java script content to tweak the phone case base on the available colors. As you can see on the right three out of these four phones has color selection options which I did not scrap.

The price ranges from $79 to $3,450. Not many cellular provider offered this ultraportable fold smart phone from Samsung (maybe Bell is the only one), but majority of wireless provider offered that cheapest phone for simple communication purpose (low resolution camera, no gyro/accelerometer sensor).

BELL MOBILITY

The Data

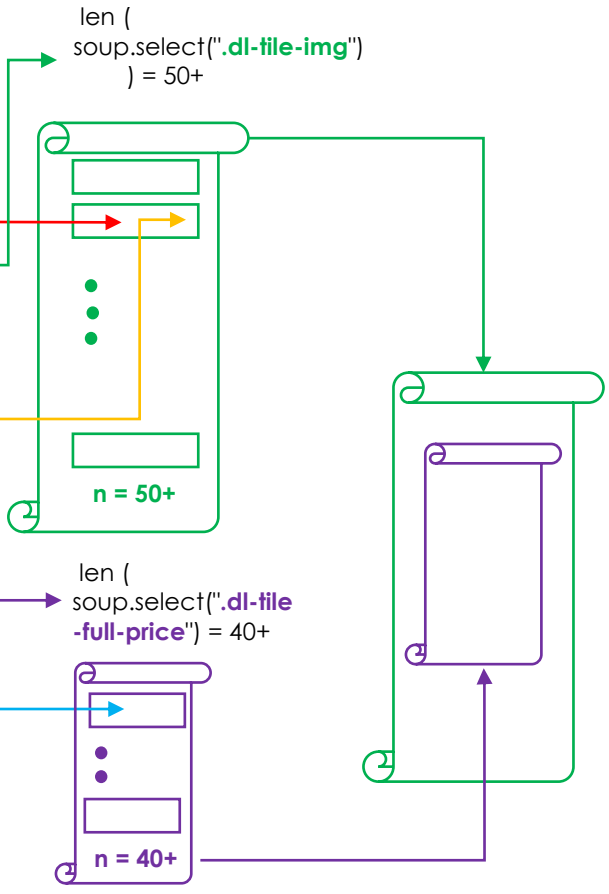
Looking at the source of the data, I would like to share quickly the 6000 lines HTML could that could be found in any typical day associated with 4 dozen phones from Bell.

Phone names and picture can be found in dl-tile-img class while the price can be found in dl-tile-full-price.

To better understand the process I gave numbering as shown to indicate data flow from beginning to the end.

1. Getting HTML source with BeautifulSoup
2. Splitting name/picture and price into 2 lists with soup.select
3. Analyzing and combining the two prior to visualization with iteration or Data Frame.
4. Visualization with Matplotlib

```
<!DOCTYPE html><html>
<head><script language="javascript">
  •
  •
  •
</div>
<div class="dl-tile"><div class="dl-tile-
content"
      data-is-basic="false" data-product-
id
      ="51d8f417-7ba5-4d0e-866e-
3561aa02b8cb">
<a aria-label="iPhone 11" class="dl-
tile-link"
      href="/Mobility/Products/iPhone-
11?INT

=MOB_mobdevpg_BTN_poplink_Mass
_051016
      _mb_details"
id="details_item_iPhone 11"></a>
     <img alt="iPhone 11"
      class="dl-tile-img"
datasrc="/styles/wireless/all_languages
/all_regions/
catalog_images/large/iphone11_purple
_lrg1.png"
     />
  •
  •
  •
</div>
<div class="dl-tile-full-price">
                  Device full price:
                  <span class="qc">
                        $1,005.00
                  </span>
</div>
.
.
 });
   </script>
</body></html>
```



iPhone 11

$ 1,005

•
•
•

n = 40+

1 2 3 4

BELL MOBILITY

Data Analysis

Combining dl-tile.img and dl-tile-full-price lists basically like an equivalent to MySQL left join, for simplification purpose I did not export this data to SQL and import them back but make a simple algorithm to sort out which phone from dl-time-img list that does not have full prices from their website.

By October 2020 there are four prepaid phones from two brands; ZTE and Alcatel as shown below. ZTE Blade A3 L for an example has two versions, one with post-paid and with full price of $79 and the other one with pre-paid. You will see 2 different rows of data at dl-tile-img for Blade A3.

Alcatel U50™ | ZTE Blade A3 L | ZTE Z557 | ZTE Cymbal 2

soup.select(".dl-tile-img")

```
[  .
   .
 <img alt="LG K61" class="dl-tile-img“ data-src="/Styles/images/LG_K61_White_lrg1.png"/>,
 <img alt="LG K41S" class="dl-tile-img" data-src="/Styles/images/LG_K41-S_Titan_lrg1.png"/>,
 <img alt="LG K31" class="dl-tile-img" data-src="/Styles/wireless/LG_K31/LG_K31_Silver_lrg1.png"/>,
   .
   .
 <img alt="ZTE Blade A3 L" class="dl-tile-img"
   data-src="/Styles/wireless/ZTE_Blade_A3_L/ZTE_Blade_A3-L_Black_lrg1.png"/>,
 <img alt="ZTE Blade A3 L - Prepaid" class="dl-tile-img"
   data-src="/Styles/wireless/ZTE_Blade_A3_L/ZTE_Blade_A3-L_Black_lrg1.png"/>,
 <img alt="ZTE Z557" class="dl-tile-img"
   data-src="/Styles/wireless/ZTE_Z557/ZTE_Z557_black_lrg1.png"/>,
 <img alt="ZTE Z557 - Prepaid" class="dl-tile-img"
   data-src="/Styles/wireless/ZTE_Z557/ZTE_Z557_black_lrg1.png"/>,
.
.
]
```

soup.select(".dl-tile-full-price")

```
[<div class="dl-tile-full-price"> Device full price: <span class="qc"> $1,200.00 </span> </div>,
 <div class="dl-tile-full-price"> Device full price: <span class="qc"> $2,300.00 </span> </div>,
 <div class="dl-tile-full-price"> Device full price: <span class="qc"> $2,000.00 </span> </div>,
 <div class="dl-tile-full-price"> Device full price: <span class="qc"> $1,650.00 </span> </div>,
 <div class="dl-tile-full-price"> Device full price: <span class="qc"> $2,250.00 </span> </div>,
 .
 .
 .
 <div class="dl-tile-full-price"> Device full price: <span class="qc"> $110.00 </span> </div>,
 <div class="dl-tile-full-price"> Device full price: <span class="qc"> $99.99 </span> </div>,
 <div class="dl-tile-full-price"> Device full price: <span class="qc"> $79.00 </span> </div>,
 <div class="dl-tile-full-price"> Device full price: <span class="qc"> $89.99 </span> </div>,
 <div class="dl-tile-full-price"> Device full price: <span class="qc"> $699.99 </span> </div>]
```

BELL MOBILITY

Visualization

plt.axis ('on') Samsung Galaxy S20 FE $949.99

plt.axis ('off') Samsung Galaxy S20 FE $949.99

Among four options of codes I pick second one to visualize on this page as it has an overall of rectangle-alike format rather than a line format. It is not fastest one to visualize.

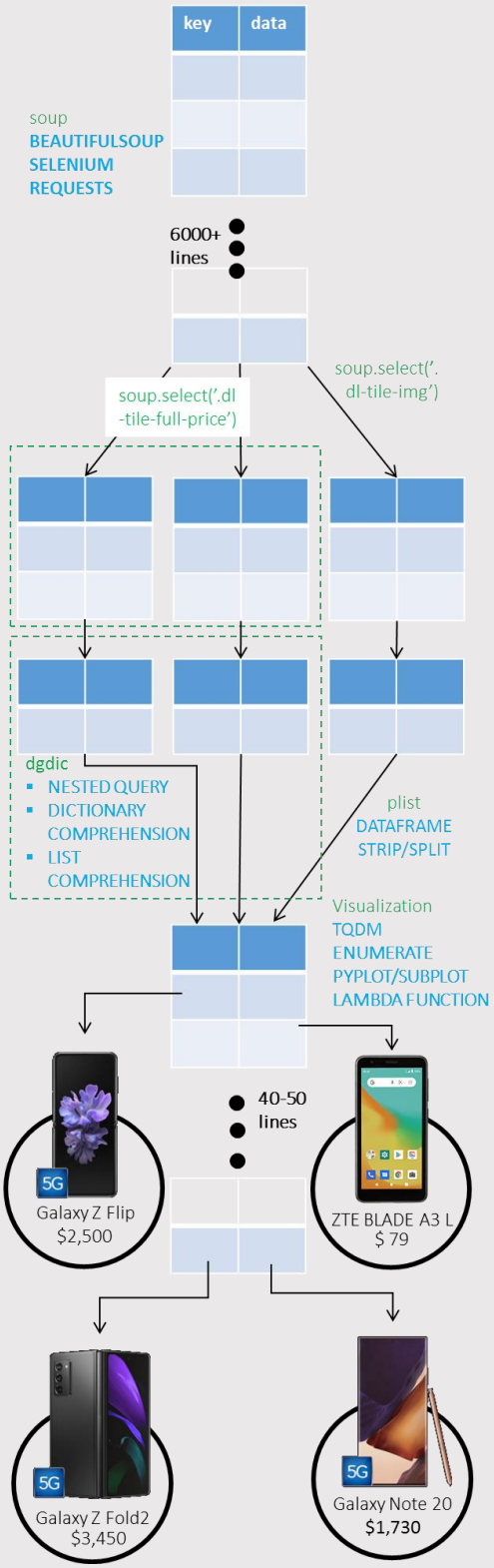
By default subplot has grid axis and for image insert the axis is not relevant and needed to drop with plt..axis('off').

Below code shows the technique of inserting an image into the subplot (instead of float, integers) and adding name and price details through its individual title shown in red, yellow and green lines below.

```
import matplotlib.pyplot as plt
import matplotlib.image as img
…
fig = plt.figure(figsize=(20,8))
for i in tqdm(range(1,len(dgdic)+1)):
  plt.subplot(5,10,i)

  plt.imshow(img
       .imread(list(dgdic.values())[i-1]))

  plt.title(list(dgdic.keys())[i-1]+'\n'+ \
       plist[i-1], fontsize=6)
  plt.axis \
   ('off')
```

BELL MOBILITY

Conclusion



To sum up, web scrapping is a wonderful project utilizing most of python skills for beginners. As you can see on the left herewith is the data flow from HTML into visualization chart, sub-processes shown in green, python tools used in blue font.

Some of popular tools utilized from Python on top of using BeutifulSoup/Selenium/Requests libraries are Data Frame, Strip/Split functions, nested query, list comprehension and dictionary comprehension approaches, TQDM, enumerate, matplotlib subplot/pyplot and lambda function.

In summary BeautifulSoup has done 99% of the scraping (from 6000 lines to 40+ lines) and the subsequent text manipulation may sort out the other 30%.

Bell website is sufficiently and easily scrapped with Beautiful Soup. In contrary Freedom Mobile cannot be scrapped by Beautiful Soup, it needs Selenium to do so.

BELL MOBILITY

Challenges

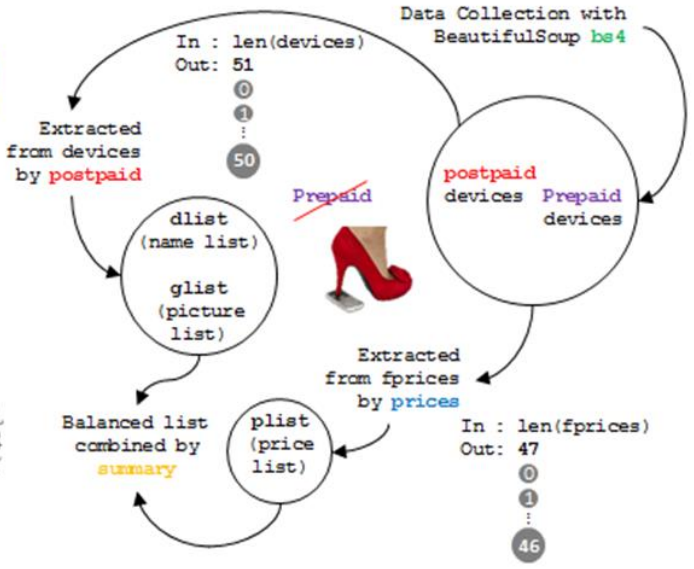
Below taught process was drawn mainly for beginner purpose or education. It probably shows the longest way to scrap as it was coming from my week 0 presentation.

The idea here is learning the basic which trying to track the progress of strings being trimmed to get right values. The color coded were selected to make it easier to follow. For below case it only use split (application of strip and nested query can reduce use of array significantly).

WEB SCRAPING SAMPLE CODE

```
import requests; import bs4
import matplotlib.pyplot as plt;
import matplotlib.image as img
bell_url="https://www.bell.ca/Mobility/\
Smartphones_and_mobile_internet_devices"
soup = bs4.BeautifulSoup(requests.get(bell_url).text,"lxml")

dlist=[]; glist=[]; devices=soup.select(".dl-tile-img")
for postpaid in range(len(devices)):
    if "Prepaid" not in str(devices[postpaid]):
        dlist += [str(devices[postpaid])
          .split('" class="dl-tile-img" data-src="')[0]
          .split('<img alt="')[1]]
        glist += ["https://www.bell.ca/"+str(devices[postpaid])
          .split('" class="dl-tile-img" data-src="')[1]
          .split('"/>')[0]]

plist=[]; fprices=soup.select(".dl-tile-full-price")
for prices in range(len(dlist)):
    plist += [str(fprices[prices]).split()[7]]

for summary in range(len(dlist)):
    im=img.imread(glist[summary])
    plt.imshow(im); plt.axis('off'); plt.show()
    print(dlist[summary]); print(plist[summary])
```

THOUGHT PROCESS DIAGRAM



SCRAP TECHNIQUE IN DETAILS

```
In : devices
Out:0[<img alt="Samsung Galaxy Z Fold2 5G " class="dl-tile-img" data-src=
              "/Styles/wireless/Galaxy_Fold_2/
               Samsung_Galaxy_Z-Fold2_Mystic_Black_5G_lrg1.png"/>,
    1 ...
      <img alt="Doro 824C " class="dl-tile-img" data-src=
              "/Styles/wireless/all_languages/all_regions/
               catalog_images/Dora824C/doro824c_lrg1_en.png"/>]
   50
In : fprices
Out:0[<div class="dl-tile-full-price"> Device full price:
                <span class="qc"> $3,450.00 </span> </div>,
    1 ...
   46 <div class="dl-tile-full-price"> Device full price:
                <span class="qc"> $699.99 </span> </div>]
In : str(fprices[0]).split()
Out:0['<div',
    1 'class="dl-tile-full-price">',
    2 'Device',
    3 'full',
    4 'price:',
    5 '<span',
    6 'class="qc">',
    7 '$3,450.00',
    8 '</span>',
    9 '</div>']
```

LEGEND

`"/>` `<img alt="` `" class="dl-tile-img" data-src="` — scraped areas / python code which removes unwanted text

50 46 0 1 0 1 7 — string position in certain list

```
In:  str(devices[0])\
     .split('" class="dl-tile-img" data-src="')
Out: [
   0 '<img alt="Samsung Galaxy Z Fold2 5G" ',
   1  '/Styles/wireless/Galaxy_Fold_2/
       Samsung_Galaxy_Z-Fold2_Mystic_Black_5G_lrg1.png"/>'
     ]

In:  str(devices[0])\
     .split('" class="dl-tile-img" data-src="')[1]\
     .split('"/>')
Out: [
   0 '/Styles/wireless/Galaxy_Fold_2/
       Samsung_Galaxy_Z-Fold2_Mystic_Black_5G_lrg1.png',
   1 ''
     ]

In:  str(devices[0])\
     .split('" class="dl-tile-img" data-src="')[0]\
     .split('<img alt="')
Out: [
   0 '',
   1 'Samsung Galaxy Z Fold2 5G'
     ]
```

BELL MOBILITY

Next Step

As shown below, the process of visualization normally took 5-8 seconds in the early weeks and as I applied Data Frame in full the visualization tool less than a second long. The overall code maybe about the same length before and after. Probably the next step is trying another website to scrap.

5-8sec

(with Python Looping)

```
…
devices = soup.select(".dl-tile-img")
for postpaid in range(len(devices)):
 if "Prepaid" not in str(devices[postpaid]):
  dlist += [str(devices[postpaid])
  .split('" class="dl-tile-img" data-src="')[0]
  .split('<img alt="')[1]]
  glist += ["https://www.bell.ca/" + \
  str(devices[postpaid])
  .split('" class="dl-tile-img" data-src="')[1]
  .split('"/>')[0]]
…
fprices = soup.select(".dl-tile-full-price")
for prices in range(len(dlist)):
 plist += [str(fprices[prices]).split()[7]]

for summary in range(len(plist)):
 im=img.imread(glist[summary])
 …
 print(dlist[summary])
 print(plist[summary])
```

< 1sec

(by Panda DataFrame)

```
…
from tqdm.notebook import tqdm;
import pandas as pd;
tqdm.pandas()
from IPython.core.display import HTML

…
dgdic = {postpaid.get('alt'): "https://www.bell.ca/"+ \
 str(postpaid.get('data-src')) for postpaid in soup
.select(".dl-tile-img") if "Prepaid" not in str(postpaid)}
…
plist = pd.DataFrame(soup.select(".dl-tile-full-price"))

def y(path):
  return '<img src="' + path + '" width ="200">'
summary = pd.DataFrame({'Phone' : list(dgdic.keys()), \
 'Price' : list(plist[1].progress_apply(lambda x: x
                                .text.strip('\r\n '))),      \
 'Picture' : list(dgdic.values())})
HTML(summary.to_html(escape=False,                          \
 formatters = dict(Picture=y)))
```

When solely using Beautiful soup to webscrap Freedom Mobile I only get 5197 long characters. It grew to 243,465 long characters when I used Selenium which mean 99% data that was truncated are now recovered. The downside of using selenium is more time needed as it simulate the way we browse things from the internet.

```
In [1]: from selenium import webdriver; driver=webdriver.Firefox()# ONLY FOR FREEDOM METHOD (SELENIUM NEEDED)
        import requests                                           # ONLY FOR BELL METHOD (BEAUTIFULSOUP ONLY)
        import bs4                                                # BOTH FREEDOM,BELL METHODS (ALWAYS NEEDED)
In [2]: soup = bs4.BeautifulSoup(requests.get('https://shop.freedommobile.ca/en-CA').text, 'lxml')
        len(str(soup))
Out[2]: 5187
In [3]: driver.get("https://shop.freedommobile.ca/en-CA"); page_source = driver.page_source
        soup = bs4.BeautifulSoup(page_source, 'lxml'); len(str(soup))
Out[3]: 243465
```

99%
unloaded data
NOW recovered

Indeed.ca

After finishing my web scraping project four of us in batch 10 volunteered to help the school consulting arm to do some web scraping given extra bandwidth to spare time within busy class schedules, assignments, preparations. Me and Victor were assigned for one project while the other two on the other on slightly different project.

The scope of the project is simply getting the list of related data science jobs in Toronto area and pulling information associated with this job listings such as date published, company name, job summary, title, location, description and url associated with this job.

date	firm	Summary	Title	Loc	Desc	url
…	…	…	…	…	…	…
…	…	…	…	…	…	http://... jk=...& fccid=... &vjs=3
…	…	…	…	…	…	…
…	…	…	…	…	…	…

Indeed job page url consists of two main components: **jk** and **fccid** numbers as shown on blues in the next page. Beautiful Soup API was applied to simply getting the summary page from all data scientist in Toronto. Further to get the details of each job we need to visited each site individually given the built-up url passed by jk and fccid.

As added work I tried to apply AWS S3 private file sharing which is repeatedly introduced in the course in different context such as when downloading titanic database.

The access key id and secret access key are unique to indeed.csv file located in my AWS file bucket, not for the whole bucket. Connecting python code to S3 with **boto3** might be a comparable approach using *AWS configure* manually done from Linux/ Git Bash through AWS EC2.

The version of this scraping code as shown on next page code but without S3 login information could be found at github.com/FariusGitHub/DataScience/blob/master/SOUP2.ipynb In professional setup IT department will provide role for data scientist to avoid hard coding AWS credential into a code.

indeed.ca

AWS S3

```
import bs4
import requests
import re
from datetime import datetime, timedelta
from tqdm import tqdm
import pandas as pd
import boto3
from botocore.exceptions import NoCredentialsError
import os
soup = bs4.BeautifulSoup(requests.get('https://ca.indeed.com/'
      'jobs?q=data+scientist&l=Toronto,+ON&start=').text, 'lxml')
DSW1=[];DST1=[];DSS1=[];DSL1=[];DSC1=[];DSD1=[]
for l in tqdm(range(0,int(soup.find('div', id='searchCountPages').text\
                          .strip('\n Page jobs').split(' of')[1])//50*50,50)):
    start = l
    soup = bs4.BeautifulSoup(requests.get('https://ca.indeed.com/jobs?q=data+\
            scientist&l=Toronto,+ON&start='+str(start)+'&limit=50').text, 'lxml')

    jk=[]; fccid=[];
    for i in range(len(soup.select('.title'))):
        j = 0 if soup.select('.date')[i].text.strip()[:2].isalpha() \
            else int(soup.select('.date')[i].text.strip()[:2])
        k=(datetime.today().date()-timedelta((j))).strftime("%x")
        jk += [str(soup.find_all("script", text=re.compile("jobmap"), \
              type="text/javascript")[0]).split('\n\n')[i+3]\
              .split("]= {jk:'")[1].split(",cmpid:'")[0].split("',efccid:")[0]]
        fccid += [str(soup.find_all("script", text=re.compile("jobmap"), \
                type="text/javascript")[0]).split('\n\n')[i+3]\
                .split("]= {jk:'")[1].split(",cmpid:'")[1].split("',num:'")[0]]
        DSW1+=['http://ca.indeed.com/rc/clk?jk='+jk[i]+'&fccid='+fccid[i]+'&vjs=3'] #WEB
        DST1+=[soup.select('.title')[i].a.get('title')]                             #TITLE
        DSS1+=[soup.select('.summary')[i].text.strip('\n')]                         #SUMMARY
        DSL1+=[soup.select('.sjcl')[i](class_='recJobLoc')[0].get('data-rc-loc')]   #LOC
        DSC1+=[soup.select('.sjcl')[i](class_='company')[0].text.strip('\n')]       #COMPANY
        DSD1+=['>=30days' if j == 30 else k]                                        #DATE
# GETTING JOB DESC FROM INDIVIDUAL JOB SITE ABOVE
DSE1=[]
print(len(DSW1))
print('grabbing job description from each website ...')
for i in tqdm(range(len(DSW1))):
    try:
        DSE1 += ['<p>'.join(str(bs4.BeautifulSoup(requests.get(DSW1[i])\
        .text, 'lxml').select(".jobsearch-jobDescriptionText")[0])\
        .split('</p>')).lstrip('<div class="jobsearch-jobDescriptionText" id=\
              "jobDescriptionText"><p>')]
    except Exception:
        DSE1 += ['error during downloading']                                        #DESC
# PREPARING REPORT AND OPEN EXCEL
df = pd.DataFrame(list(zip(DSD1, DSC1, DST1, DSL1, DSS1, DSE1, DSW1)),
      columns =['Posted', 'Company', 'Job Title', 'Location', 'Summary', 'Desc', 'url'])
df.drop_duplicates(subset='url').to_csv('indeed.csv', index=False)
boto3.client('s3', aws_access_key_id='AKIAJYUE5PDTZWIWATPA',
    aws_secret_access_key='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')\
    .upload_file('indeed.csv', 'farius', 'upload_files/indeed.csv')
boto3.client('s3', aws_access_key_id='AKIAJYUE5PDTZWIWATPA',
    aws_secret_access_key='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')\
    .download_file('farius', 'upload_files/indeed.csv', 'indeed_eric.csv')

os.startfile('indeed_eric.csv')
```

monster.ca

While Victor agreed working on LinkedIn and Glassdoor job sites, I also taking extra mile to work on Monster job site. Unlike indeed page, monster page does not have a summary section for each job posted. It also has detail description for each from the link given for each post at summary page. Unlike indeed, monster does not confuse the link with two main components.

NATURAL LANGUAGE PROCESSING

As the request coming at the same week we learned NLP as part of deep learning I quickly offer this idea by adding / anticipating missing Monster summary section by default.

Title	Firm	Loc.	Summary	Desc.	Date	Duration	Link
…	…	…	…	…		…	…
…	…	…	…	…		…	…

The key part of NLP to build the summary is referring what are needed to search, such those keywords that can be found in https://farius.s3.amazonaws.com/jargons_01.csv

The entire monster code also can be found at https://colab.research.google.com/github/FariusGitHub/DataScience/blob/master/SOUP3.ipynb

There are two type of tools applied to identified the text: punctuation and ngrams. The punctuation feature takes wording such as 'machine-learning' and 'machine learning' as the same thing where Ngrams is scanning the text based on 1,2 or 3 set of words as one search.

building
training
scoring
monitoring
engineering
statistics
mathematics
statistical
mathematical
optimization
simulation
visualization
regression
cluster
supervised
unsupervised
classification
building
training

Predictive modeling
Predictive attributes
Advanced analytics
Feature engineering
Computer science
Machine Learning
Random Forest
Data Science
Time Series
Neural Net
Big Data

machine learning models
machine learning concept
machine learning algorithm

monster.ca NLTK

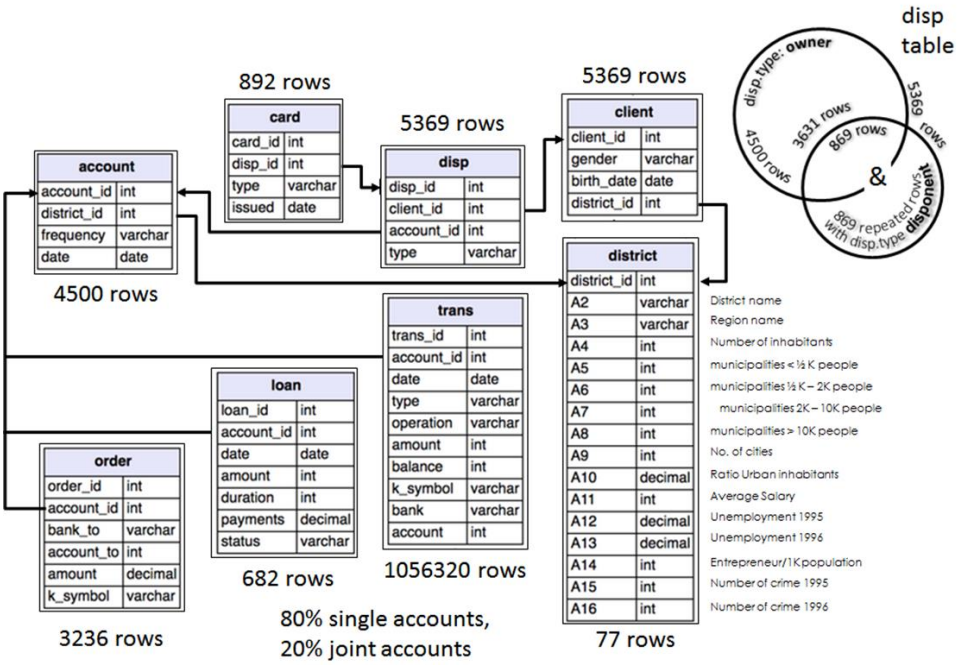
```
import bs4; import requests; from tqdm import tqdm; import pandas as pd; import numpy as np;
from string import punctuation; from nltk.stem import PorterStemmer; from datetime import date

soup = bs4.BeautifulSoup(requests.get(\
   'https://www.monster.ca/jobs/search/?q='
   'data-scientist'
   '&where=Canada&intcid=skr_navigation_nhpso_searchMain&page=10').text, 'lxml')

#inserting Title, Company, Location, Date, Duration column : FIRST iteration
Company=[];Location=[];Title=[];Duration=[]; Link=[];
for i,j in enumerate(tqdm(soup.select('.company'))):
            Company  += [j.text.strip('\n')]
            Location += [soup.select('.location')[i+1].text.strip('\n\r')]
            Title    += [soup.select('.title')[i+3].text.strip('\r\n')]
            Duration   += ['30 days or more' if soup.find_all(class_=\
                        'meta flex-col')[i].text[1:3]=='30' else\
                        'posted today'    if soup.find_all(class_=\
                        'meta flex-col')[i].text[1:3].isalpha() else\
                                          soup.find_all(class_=\
                        'meta flex-col')[i].text[1:3] + ' days ago']
            Link  += [soup.select('.title')[i+3].a.get('href')]

monster = pd.DataFrame(np.asarray(list(zip(Title, Company, Location, \
          Duration, Link))), columns =['Title', 'Company', 'Location',\
          'Duration', 'Link']).drop_duplicates()\
          .reset_index(drop=True)

#inserting 'Date' column : today's date when this code run, no iteration needed
monster.insert(3, 'Date', date.today())

#inserting 'Description' column : SECOND iteration
desc=[]
for i in tqdm(monster['Link']):
    try:
        desc += [bs4.BeautifulSoup(requests.get(i).text,\
                'lxml').select('.row')[3].text]
    except:
        desc += ['error during downloading']
monster.insert(3, 'Description', desc)

#Inserting 'Summary' column : THIRD iteration
jargons=pd.read_csv("https://farius.s3.amazonaws.com/jargons_01.csv",\
                    header=None)
ps = PorterStemmer()

Summary={};ngram1=[]; ngram2=[]; ngram3=[]
for k,j in enumerate(tqdm(desc)):
    cleaned = ''.join([char if char not in punctuation else ' '\
              for char in j.lower() ])
    ngram1=set([word for word in [cleaned.split()[i] \
              for i in range(len(cleaned.split()))] \
              if word in jargons[0].values.tolist()])
    ngram2=set([word for word in [cleaned.split()[i] +' '+ \
              cleaned.split()[i + 1] for i in range(len(cleaned.split()) - 1)] \
              if word in jargons[0].values.tolist()])
    ngram3=set([word for word in [cleaned.split()[i] +' '+ \
              cleaned.split()[i + 1] +' '+ ps.stem(cleaned.split()[i + 2]) \
              for i in range(len(cleaned.split()) - 2)] \
              if word in jargons[0].values.tolist()])
    Summary[k]=list(ngram1)+list(ngram2)+list(ngram3)

monster.insert(3, 'Summary', Summary.values()); monster.style.hide_index()
```

WEB SCRAPING

NATURAL LANGUAGE PROCESSING

Loan Default Modeling

Part of machine learning for most student perhaps working on this dataset where it has a little bit of everything: long dataset, opportunity to revisit SQL, running the model from Python and applying Scikit learn libraries such as decision tree classifier, random forest classifier, pipeline, gridsearch, random oversampling, standardscaler, lime, PCA, SVC etc.



The detail of the quality of the data will be elaborated later in page 23 while The figure above simply illustrates the raw data given for the exercise that mainly using more on loan table among other tables as part of supervised machine learning experience labeling the account that likely to default.

In borrowing and lending industry there is an essential part called trust. Unlike US credit card market where most bank intentionally benefit from late payer, bad credit to get fee from penalties, interest over interest, in this case we would stay traditional by labeling late payer as potential loan default which alerting to decline future credit application.

In this study we depends solely the credit rating from the database so called column 'status' in table 'loan'. No external or live streaming data to update this status.

LOAN DEFAULT

The Objective

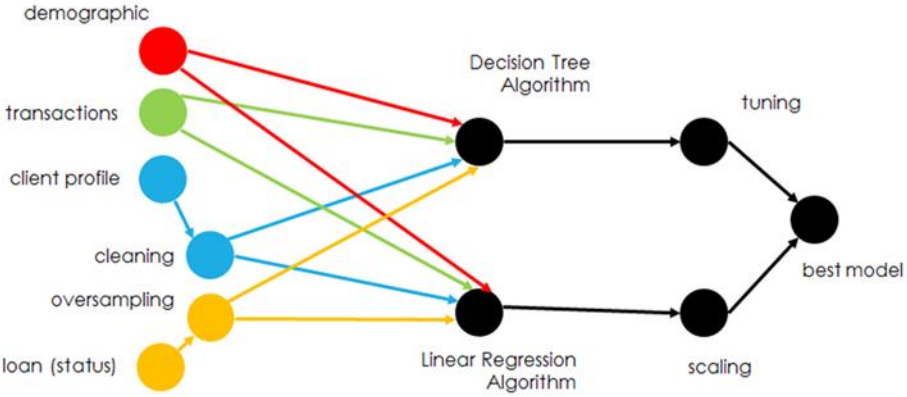
The objective of this project is self-decided, an open ended efforts as there are abundant tools from Scikit Learn to pick while learning supervised machine learning.

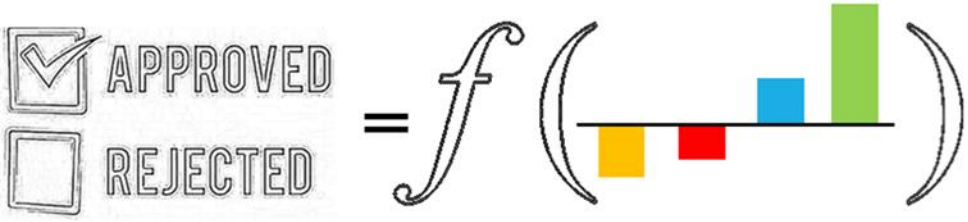
Given the dataset shown in previous page we will cover several data exploratory, selecting feature, adding new feature, selecting the model and interpret the model. Therefore we will cover feature importance, linear regression, several popular model from sigmoid to SVC,

The methodology is therefore an important part to achieve the result. How many features added, exploratory data analysis (EDA), how to resolve imbalance data with oversampling, when to train test split, how to utilize most machine learning tools (pipeline, gridsearch, K-Means) as well as some show casing proficiency in coding Python (data frame, classification), SQL Alchemy, MySQL syntax.

The bottom line would be predicting approval or decline decision based on loan activity in the past, shows feature importance that lead to make that approval decision.

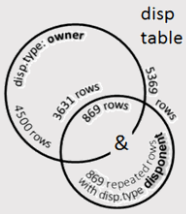
In this portfolio I added my thought about confusion matrix in different view and apply it repeatedly for loan default.





LOAN DEFAULT

THE DATA

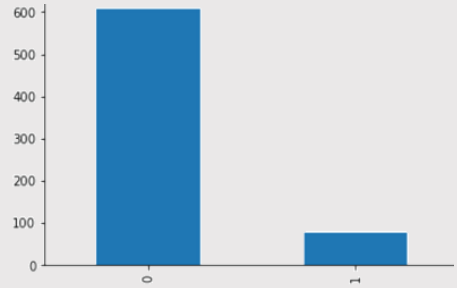


account	
account_id	int
district_id	int
frequency	varchar
date	date

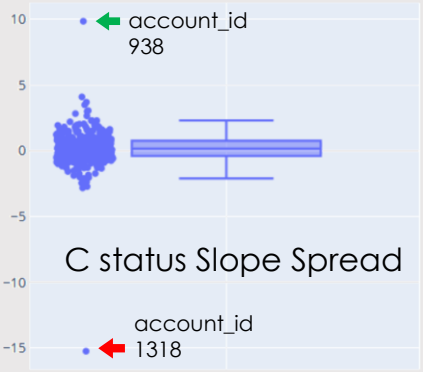
disp	
disp_id	int
client_id	int
account_id	int
type	varchar

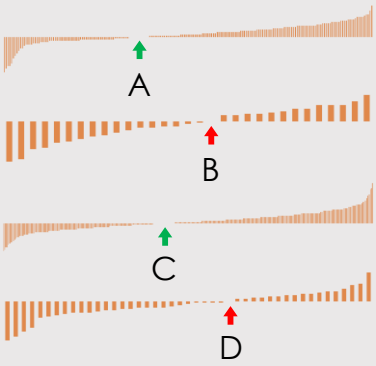
client	
client_id	int
gender	varchar
birth_date	date
district_id	int

IMBALANCE DATA



LINEAR REGRESSION AND BALANCE TREND, OUTLIERS





682 account_id
Scaled Slopes

Based on late 1990s data from a Czech Republic bank, eight tables were given and its relations can be found in page 21. Joining all the 8 table would be helpful such as to know the relation between gender split for few accounts (if it is a family account) in relation to likelihood of being default. There are 4500 rows in account table but 5369 rows in disp or client tables coming from 869 duplicated account_id (an account with husband and wife profiles). See the diagram on the left.

In loan table there are 203 and 403 accounts with status A and C. 31 and 45 with status B and C. In this case 606 accounts labeled as non-defaulter (0) and 76 as potential defaulter (1) due to their late payment reputation. To increase model performance we need to run holdout validation or k-fold validation (page 26) which require random over sampling (ROS) or stratify=y for each case.

Classifying and focusing on particular subset of data or for feature engineering it is necessary run linear regression on transaction table which stored information of cash in and out which powerful to anticipate default (see below)
https://github.com/FariusGitHub/DataScience/blob/master/slope.ipynb

The first chart on the left summarizes balance trends (slope) only for Status C accounts as the other accounts from other status spread closer to each other. I opted out those two outliers to present the chart below. As we have more account for A and C we can see the population are more than B and D. To compare all of the four I use scaling technique to show positive/negative trend from each. As we classify A and C as potential non-defaulters we see the evidence than more half of account holders have positive trend where potential defaulters have opposite situations.

	loan_id	account_id	date	amount	duration	payments	status	slope
371	5784	3967	970329	84600	36	2350.0	D	2.670377
341	6255	6158	970208	247920	48	5165.0	D	1.632903
329	7122	10365	970104	260640	36	7240.0	D	1.500646
279	5606	3084	960828	253512	36	7042.0	D	1.135339
123	5624	3166	950103	177744	48	3703.0	D	1.015049
⋮								
603	5683	3460	980529	140688	48	2931.0	C	-2.723920
602	7202	10812	980529	140688	48	2931.0	C	-2.829720
661	5226	1318	981011	185544	36	5154.0	C	-15.245095
220	6303	6400	960212	72408	24	3017.0	B	1.061436
243	5892	4462	960514	66480	24	2770.0	B	0.787301
⋮								

LOAN DEFAULT

ETL

There are two ways of getting data for this portfolio. The original data were in csv formats. There were later loaded into MySQL in sql format or to S3 in CSV format. See below.

#ETL FROM MYSQL (SQL ALCHEMY)

```
while True:
 try:
  password =
   getpass.getpass('Enter
   localhost password: ')
    engine =
     create_engine('mysql+
     pymysql://root:'+
     password+
     '@localhost:3306/bank2',
     pool_size=50,
     max_overflow=0)
    connection =
     engine.connect()
    break
 except:
  print('try again')

imbloan = pd.read_sql(
'''
select loan.account_id,
       loan.amount,
       loan.duration,
       card.type as ccard,
       client2.gender as
        sexes,
       district.A10 as urban,
       district.A11 as salry,
       (district.A12+district.
        A13)/2 as unemp,
       district.A14 as entpr,
       (district.A15+district.
        A16)/2 as crime,
       loan.status
from loan

join account on
 account.account_id =
 loan.account_id
join district on
 account.district_id =
 district.district_id
join disp2 on disp2.account_id
 = account.account_id
left join card on
 card.disp_id=disp2.disp_id
join client2 on
 client2.account_id =
account.account_id

''',con=connection)
```

#ETL FROM AWS S3 (IF SQL LOCALHOST IS NOT INSTALLED)

```
T1=pd.read_csv("https://midterm-ml.s3.amazonaws.com/
   loan.csv", sep=";")[['account_id', 'amount',
   'duration', 'status']]
T2=pd.read_csv("https://midterm-ml.s3.amazonaws.com/
   account.csv", sep=";")[['account_id', 'district_id']]
T3=pd.read_csv("https://midterm-ml.s3.amazonaws.com/
   district.csv", sep=";")[['district_id','A10',
   'A11','A12','A13','A14','A15','A16']]
T4=pd.read_csv("https://midterm-ml.s3.amazonaws.com/
   disp2.csv", sep=";")[['account_id', 'disp_id',
   'client_id']]
T5=pd.read_csv("https://midterm-ml.s3.amazonaws.com/
   card.csv", sep=";")[['disp_id', 'type']]
T6=pd.read_csv("https://midterm-ml.s3.amazonaws.com/
   client2.csv", sep=";")[['client_id','gender']]
T7=pd.merge(T4,T5, on='disp_id')[['account_id', 'type']]
T8=pd.merge(T4,T6, on='client_id')[['account_id',
   'gender']]
T9=pd.merge(T2, T3, on='district_id')[['account_id',
   'A10', 'A11', 'A12', 'A13', 'A14', 'A15', 'A16']]
T0=T1.merge(T7.merge(T8.merge(T9, on='account_id'),
   on='account_id', how='right'), on='account_id')
T0['unemp']=(T0['A12']+T0['A13'])/2
T0['crime']=(T0['A15']+T0['A16'])/2
imbloan=T0.rename(columns={'type': "ccard", 'gender':
        "sexes", 'A10': "urban", 'A11': "salry", 'A14':
        "entpr"}).drop(['A12', 'A13', 'A16', 'A15'],
        axis = 1)

imbloan = imbloan.set_index('account_id')
```

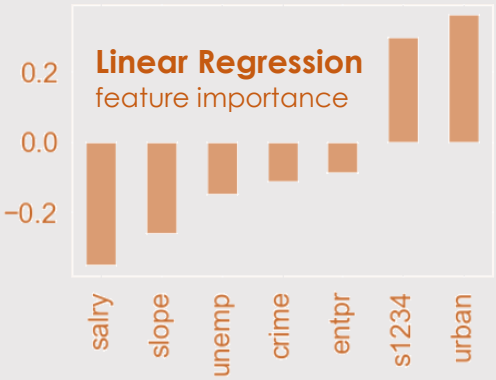
In some cases MySQL could crash and slow down the loading of large data into the sql dataset (especially trans table). The backup plan is loading the whole data into S3.

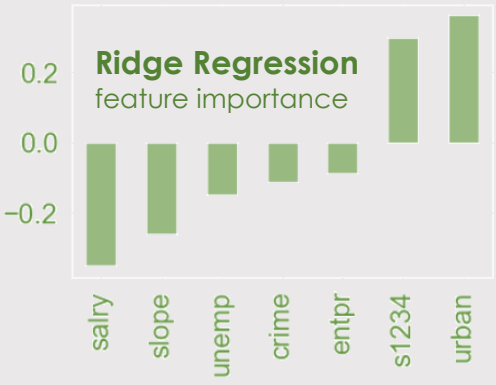
The other two tables that I loaded into S3 are transaction and loan tables were I stored into trans and abcd variable and mainly used for linear regression analysis to find the balance trend (slope). Trans.csv is a large data with more than a million row and does not fit into excel spreadsheet

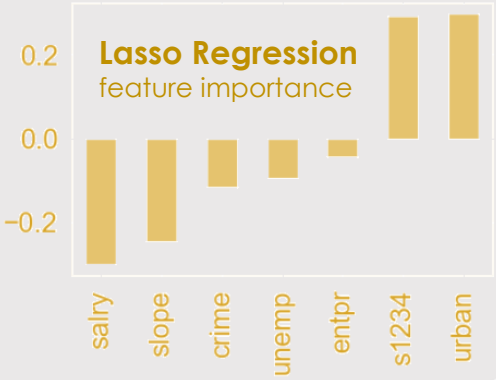
```
trans=pd.read_csv("https://midterm-ml.s3.amazonaws.com/
      trans.csv", sep=";")

abcd =pd.read_csv("https://midterm-ml.s3.amazonaws.com/
      loan.csv", sep=";")
```

LOAN DEFAULT

Feature Selection







Selecting features would be next after data collection and feature engineering (80% of works) for machine learning.

COMMON MISTAKES

Including account_id (numeric data for identification unrelated to load default) would skew the modeling result and difficult to interpret. The best practice would be inserting this info into Data Frame index, not a column

FEATURE SELECTION

Into my model I took seven features from loan, client and district tables which are gender by account type (s1234), urbanization rate (urban), entrepreneur density (entpr) unemployed rate (unemp), district salary (salry), crime rate (crime) and a feature describing cash held trend (slope). I took dflt? label for decision tree, d0123 for linear regression

```
imbloan2['dflt?'] = imbloan2['status'].apply(lambda x: 1
                    if x=='B' or x=='D' else 0)
imbloan2['d0123'] = imbloan2['status'].apply(lambda x: 3
                    if x=='D' else (2 if x=='B' else 1
                    if x=='C' else 0))
imbloan2['s1234'] = imbloan2['sexes'].apply(lambda x: 4
                    if x == 'male owner only'
                     else (3 if x=='female owner only'
                      else 2 if x=='female disponent
                       added' else 1))
```

2-LABEL MODEL

It would be the most discussed in this portfolio by exploring popular model prior to decide, fine tune, cross-validate with 6 features (to avoid strong factor from slope) by using label dflt?, popular SciKit Learn machine learning libraries

4-LABEL MODEL

At further exploratory using 4 labels (d0123) slope feature is no longer the most dominant as the model able to classify the likelihood to default into 4 labels rather than just 2. We use alpha = 0.001 to enable importance of entpr to show.

SCALING THE RIGHT WAY

In feature selection process it would necessary to scale feature importance and scaling both train and test data are necessary after the data was split and oversampled.

```
X_train, X_test, y_train, y_test =
   train_test_split(X, y, test_size=0.20)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

LOAN DEFAULT

Modelling Approach

DATA LEAKAGE (WRONG WAY OF OVERSAMPLING)

```
ros =
   RandomOverSampler()

X_resampled, y_resampled
   = ros.fit_resample(X,
   y)

X_train, X_test,
   y_train, y_test =
   train_test_split(X_res
   ampled, y_resampled,
   test_size=0.33)
```

	precision	recall	f1-score	support
0	0.94	0.61	0.74	122
1	0.18	0.67	0.28	15
accuracy			0.62	137
macro avg	0.56	0.64	0.51	137
weighted avg	0.85	0.62	0.69	137

	precision	recall	f1-score	support
0	0.96	0.60	0.74	487
1	0.19	0.79	0.31	58
accuracy			0.62	545
macro avg	0.58	0.70	0.52	545
weighted avg	0.88	0.62	0.69	545

INCORRECT WAY TO VALIDATE

```
ros = RandomOverSampler()

X_resampled, y_resampled=
   ros.fit_resample(X_tra
   in_full, y_train_full)

model =
   DecisionTreeClassifier
   (max_depth=50)

scores =
   cross_val_score(model,
   X_resampled,
   y_resampled, cv=5,
   scoring='f1')

scores.mean()
```

Defining the label, creating baseline model, comparing models, tuning hyper-parameter, cross-validating scores, pipelining/grid searching would be the approach here.

TEST TRAIN SPLIT

it's needed to *split test data first* (ros, validate, scale, impute)

```
X_train_full, X_test, y_train_full, y_test =
   train_test_split(X, y, test_size=0.20)
X_train, X_val, y_train, y_val = train_test_split
   (X_train_full, y_train_full, test_size=0.25)
ros = RandomOverSampler()
X_resampled, y_resampled = ros.fit_resample(X_train,
   y_train)
```

HYPERPARAMETER TUNING

Use validation data to tune, use entire train data to report

```
model = DecisionTreeClassifier(max_depth=5)
model.fit(X_resampled, y_resampled)

y_pred_final = model.predict(X_val)
print(classification_report(y_val, y_pred_final))

y_pred = model.predict(X_train_full)
print(classification_report(y_train_full, y_pred))
```

VALIDATION

For each validation, it's needed to *split test data for each*

```
kf = KFold(n_splits=5, random_state=42, shuffle=True)
scores = []
for train_index, val_index in
   kf.split(np.array(X_train_full)):

        X_train = (np.array(X_train_full)[train_index])
        X_val = (np.array(X_train_full)[val_index])
        y_train = (np.array(y_train_full)[train_index])
        y_val = (np.array(y_train_full)[val_index])

   ros = RandomOverSampler(sampling_strategy='all',
         random_state=42)

   X_train_resampled, y_train_resampled =
         ros.fit_resample(X_train, y_train)

   model = DecisionTreeClassifier(max_depth=1,
           random_state=42)

   model.fit(X_train_resampled, y_train_resampled)

   y_pred = model.predict(X_val)

   scores.append(f1_score(y_val, y_pred))
```

LOAN DEFAULT

Confusion Matrix

	APPROVED (y_pred=0)	REJECTED (y_pred=1)
Actually not Default (y_test=0)	75 True Positive	47 False Negative
Actually Default (y_test=1)	5 False Positive	10 True negative

	APPROVED (y_pred=0)	REJECTED (y_pred=1)
Actually not Default (y_test=0)	75 True negative	47 False Positive
Actually Default (y_test=1)	5 False Negative	10 True Positive

	APPROVED (y_pred=0)	REJECTED (y_pred=1)
Actually not Default (y_test=0)	101 True negative	21 False Positive
Actually Default (y_test=1)	12 False Negative	3 True Positive

Decision Tree Baseline Model

Jupyter Notebook gives precision, recall, f1 scores for non-defaulters (y_test = 0) and defaulter (y_test = 1) as below

When True Positive is **non Defaulter**

$$precision = \frac{TP}{TP + FP} = \frac{75}{75 + 5}$$

$$recall = \frac{TP}{TP + FN} = \frac{75}{75 + 47}$$

$$accuracy = \frac{TN + TP}{TN + TP + FP + FN} = \frac{10 + 75}{10 + 75 + 5 + 47}$$

$$F1\ Score = \frac{2.TP}{2.TP + FP + FN} = \frac{2 * 75}{2 * 75 + 5 + 47}$$

	precision	recall	f1-score	support
0	0.94	0.61	0.74	122
1	0.18	0.67	0.28	15
accuracy			0.62	137

When True Positive is **Defaulter**

$$precision = \frac{TP}{TP + FP} = \frac{10}{10 + 47}$$

$$recall = \frac{TP}{TP + FN} = \frac{10}{10 + 5}$$

$$accuracy = \frac{TN + TP}{TN + TP + FP + FN} = \frac{75 + 10}{75 + 10 + 47 + 5}$$

$$F1\ Score = \frac{2.TP}{2.TP + FP + FN} = \frac{2 * 10}{2 * 10 + 47 + 5}$$

Random Forest Tree Model and beyond

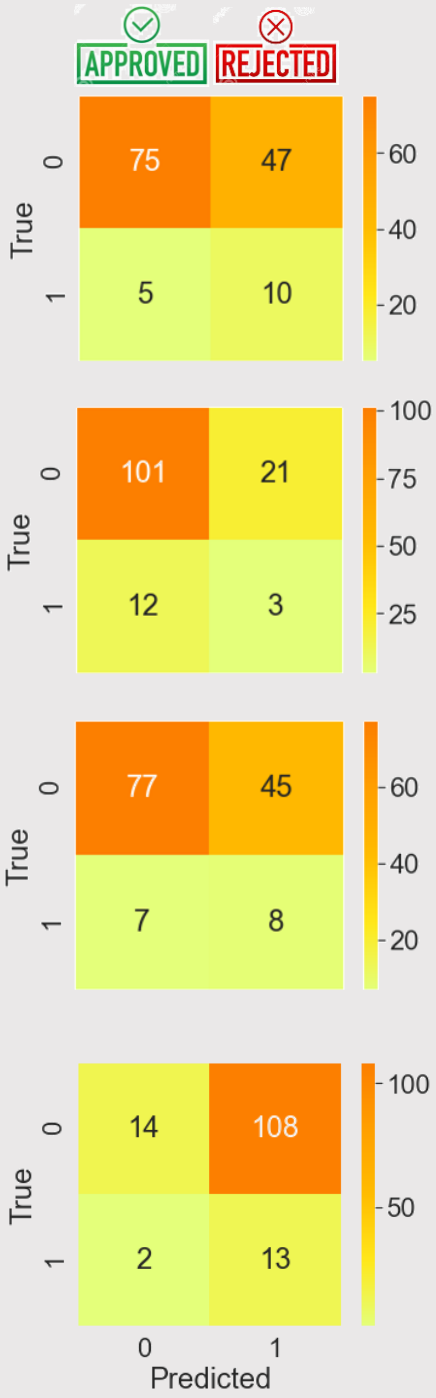
Fine tuning with different model such as Random Forest it gives better precision and recall as below. For this loan default model the precision went down from 18% to 12%, recall from 67% to 20% and accuracy from 62% to 76%.

	precision	recall	f1-score	support
0	0.89	0.83	0.86	122
1	0.12	0.20	0.15	15
accuracy			0.76	137
macro avg	0.51	0.51	0.51	137
weighted avg	0.81	0.76	0.78	137

LOAN DEFAULT

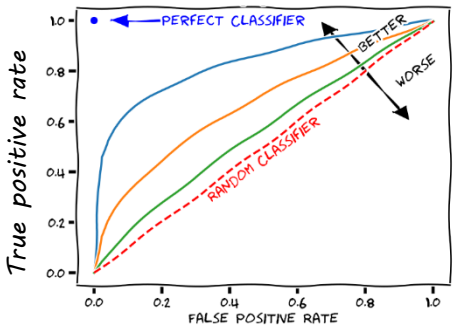
Model Selection

Confusion Matrices of Model Candidates
(x axis is the prediction and y axis is the truth)



Selecting the model would be focus on f1 score values or probably easier to look at the highest AOC (area under ROC curve). The Receiver Operating Characteristic (ROC) below show several scenarios. As shown below the higher AOC the better classifier it would be.

ROC Curve



False positive rate

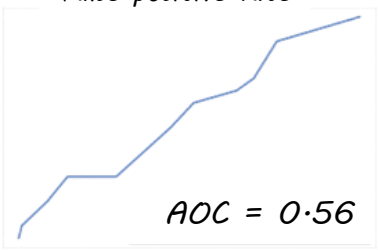
SVC shows overfitting model while logistic regression shows under fitting. Best choices are random forest or decision tree.

In larger database random forest may need longer time, more memory and more cost.

DECISION TREE MODEL (depth=5)

	precision	recall	f1-score
0	0.94	0.61	0.74
1	0.18	0.67	0.28

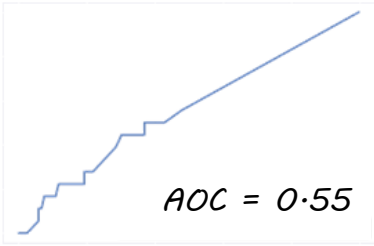
BEST MODEL, SIMPLE, FAST



RANDOM FOREST (estimators =3)

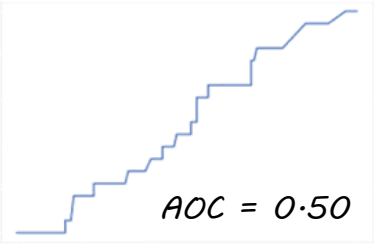
	precision	recall	f1-score
0	0.89	0.83	0.86
1	0.12	0.20	0.15

SECOND BEST, MAY BE COSTLIER



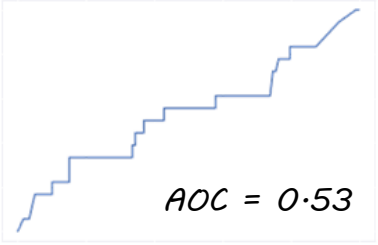
LOGISTIC REGRESSION (UNDERFIT)

	precision	recall	f1-score
0	0.92	0.63	0.75
1	0.15	0.53	0.24



SVC MODEL (OVERFIT)

	precision	recall	f1-score
0	0.88	0.11	0.20
1	0.11	0.87	0.19



Conclusion

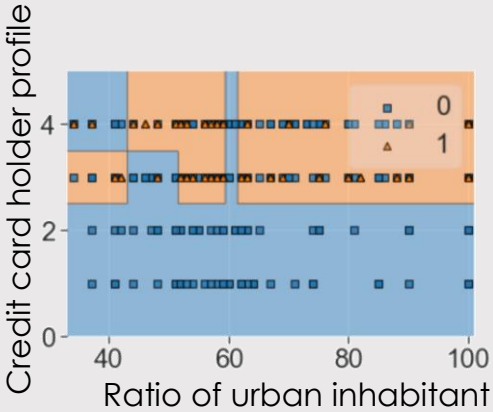
Loan Default assignment was a perfect experience applying many basic SciKit Learn libraries for classification, regression, regularization for machine learning.

Exploring the data itself is equally important without the good knowledge in loan business domain. As mentioned in page 25 adding unrelevant data such as account_id could ruin the model interpretability.

Correct methodologies are the other important part of the logic. We do not want to oversample the test data itself and rather focus on train data and improve the model.

Decision tree model would suit nice for this loan default project and many more works needed beyond data science context, such as revenue generation/loss than just business profit/loss by qualifying false/true good loaner.

Another way looking at decision tree in more focus model is applying MLXtend *boundary region plot* and Graphviz *graph-tool library* from specific two features (such as credit card profile and ratio of urban inhabitants) below. Using max_depth between 10 and 20 it will increase the model score from 60% to 72.5%.




```
imbloan2['s1234'] =
imbloan2['sexes'].apply(lambda x:
4 if x == 'male owner only' else (
3 if x== 'female owner only' else
2 if x== 'female disponent added'
else
1))
```

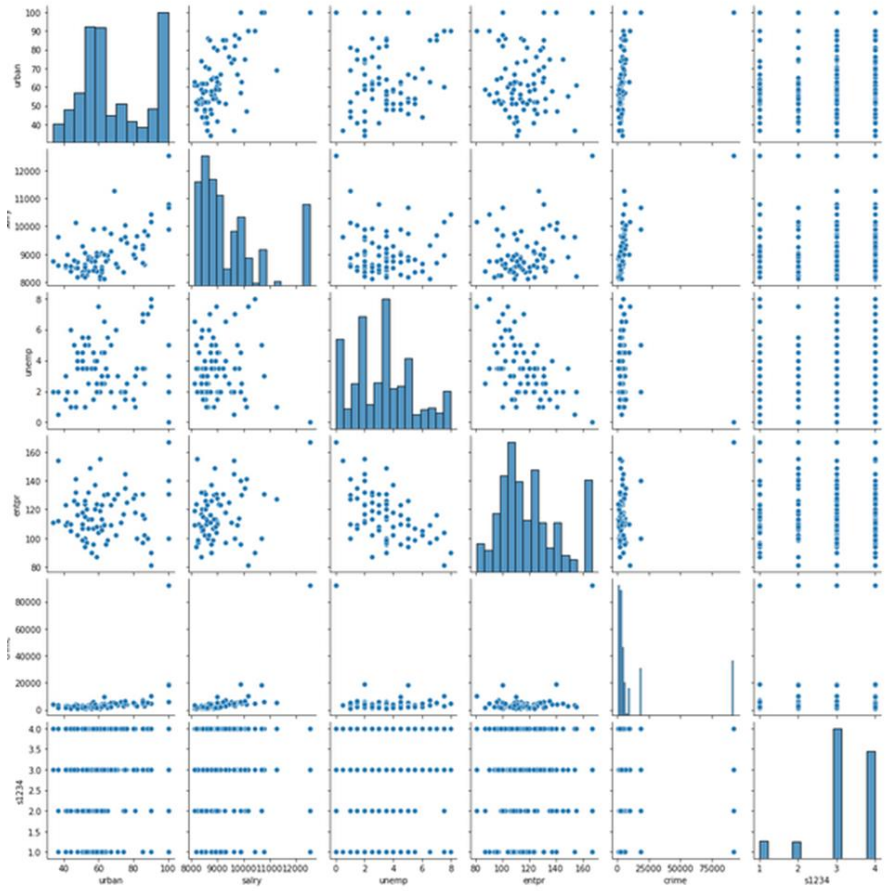
Challenges

DATABASE MANAGEMENT

Managing small to mid size database could be easy but could also experience unpredictable outcome. I smoothly queried and joined the tables within DataGrip and sent them through SQLAlchemy until at some point when I about to tweak the query for faster performance my system went too slowl and frustration escalated as I need to upload the csv data manually such as from local hard.

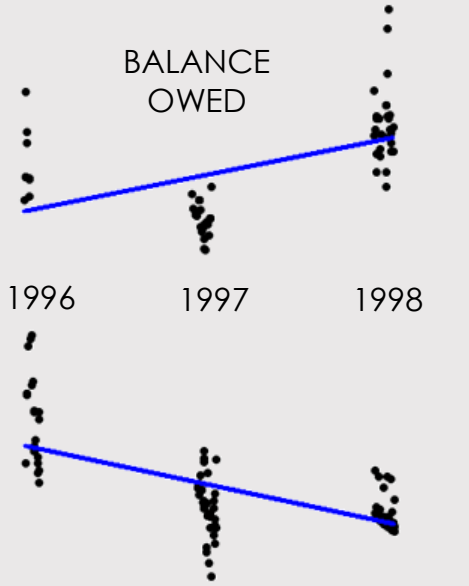
MACHINE LEARNING

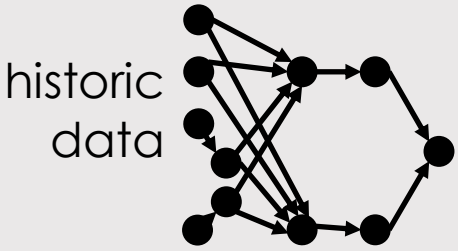
Scikit Learn is a powerful tool for machine learning. Given speculation in the first time that I limited few demographic information it would be very hard to see linear relationship thus harder to run a model. I agree that the demographic information also does not reflect someone ability to repay the loan as it takes the sample of the neighborhood not necessarily one's financial/family situation. Pairplot below shows minimum linear relationships to help me run a model

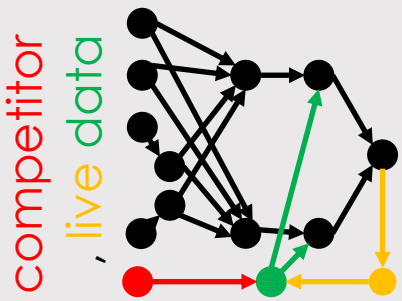


LOAN DEFAULT

Next Steps







LINEAR REGRESSION, FEATURE ENGINEERING

One of the good use of those 'transaction' table with a million row is predicting client ability to keep cash to anticipate unexpected spending while repay loan. It is an important feature to add while approving/declining loan application

	loan_id	account_id	date	amount	duration	payments	status	slope
454	6612	7907	970831	97392	12	8116.0	A	2.115064
18	6876	9236	931221	86616	12	7218.0	A	1.780508
107	6253	6148	941104	59760	24	2490.0	A	1.522994
276	5988	4851	960822	141648	24	5902.0	A	1.469493
290	6995	9814	960926	96168	12	8014.0	A	1.443873
...	...	...	...	...	...	...	...	...
572	6682	8225	980330	36204	12	3017.0	D	-1.540422
663	4967	37	981014	318480	60	5308.0	D	-2.462606
471	6239	6083	970928	360864	48	7518.0	D	-2.781967
635	5991	4858	980727	335184	48	6983.0	D	-3.254940
491	5731	3711	971105	460980	60	7683.0	D	-3.627283

682 rows × 8 columns

CREDIT RATING AGENCY

Looking at someone credit status is a complex business. It would not be as easy as identified where he/she lives but should involve many other personal and intimate including saving, salary, pension, education, family status, etc where credit rating agency TransUnion, Equifax are the leaders.

LIVE STREAMED COMPETITIVE ANALYSIS

Looking at growing lending industry predominantly driven by non-bank institutions it is important to consider the lost of revenue as well as the lost of profit in the first place when rejecting loans. Says if TD rejects someone loan and he/she went to PayDay Loan and pay his debt on time it would indicate a lost of sales and lost of profit to TD.

Adding those factors above would make the prediction model even better, more accurate and productive.

LOAN DEFAULT

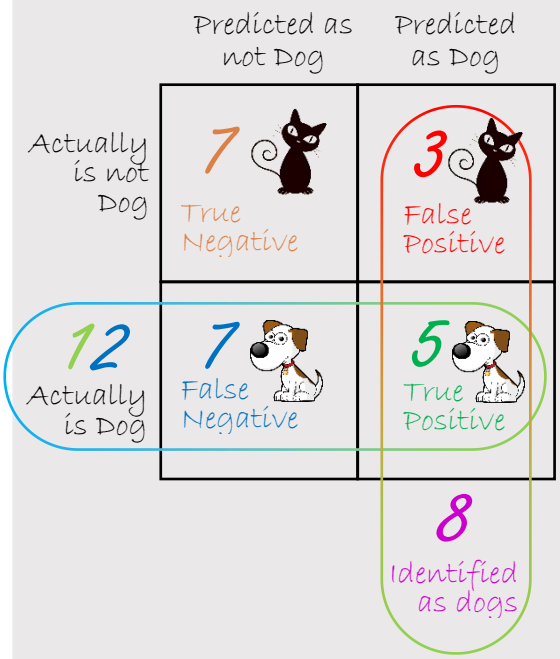
Confusion Matrix and Jupyter Notebook

(Appendix one)

WIKIPEDIA

Taking a popular confusion matrix as a comparable study to this portfolio I would like to bring cat and dog example widely taken by Wikipedia under precision and recall topic https://en.wikipedia.org/wiki/Precision_and_recall

Suppose a computer program for recognizing dogs in photographs identifies 8 dogs in a picture containing 10 and 12 (the relevant element). Of 8 identified as dogs, 5 actually are dogs (true positives), while the other 3 are cats (false positives). 7 dogs were missed (false negatives), and 7 cats were correctly excluded (true negatives).



$$precision\ (how\ valid) = \frac{\text{Relevant} \cap \text{Retrieved}}{\text{Retrieved}} = \frac{TP}{TP + FP} = \frac{5}{5 + 3}$$

$$recall\ (how\ complete) = \frac{\text{Relevant} \cap \text{Retrieved}}{\text{Relevant}} = \frac{TP}{TP + FN} = \frac{5}{5 + 7}$$

$$accuracy = (TP + TN)/(TP + TN + FP + FN) = 6/11$$

$$F1\ Score = \text{harmonic mean of precision and recall} = \frac{2.TP}{2.TP + FP + FN} = 1/2$$

JUPYTER NOTEBOOK

Confusion matrix at Jupyter notebook calculates both true negatives (0) and true positives (1) values for precision, recall and F1 score separately at the same time and tabulated them with common accuracy value.

At loan default when looking at non defaulters they are categorized as true positive and precision, recall and f1-score are shown in row 0. When looking at defaulters we categorize them as the new true positive and those three values are summarized in row 1. See page 27 for details.

	precision	recall	f1-score	support
0	0.94	0.61	0.74	122
1	0.18	0.67	0.28	15
accuracy			0.62	137

LOAN DEFAULT

Pipeline & Gridsearch

(second appendix)

Calling modeling preparation altogether prior to fit to the model Scikit Learn provides a Pipeline function to summarize all preparation as follow. Train-test splitting and oversampling of train data are still need to be done externally.

One of great work done for GridSearchCV was made by David Batista, Wesley Rocha (http://github.com/davidsbatista/machine-learning-notebooks/blob/master/hyperparameter-across-models.ipynb)
Applying a work from cancer research into loan default I can bring some arguments for selecting decision tree (DTR). Based on test data split early ingested into this model we could see three top spots occupied by Random Forest (RFT) model but never reach the maximum. We took AUC as a criteria. On page 28 ROC curves were built based on X_resampled & y_resampled data while for this search I use test data earlier split from remaining data.

As the max scores and high mean scores were held by decision tree models I am confident to qualify decision tree model among SVC, RFT.

PIPELINE

```
X_train_full, X_test, y_train_full, y_test =
   train_test_split(X, y, test_size=0.2)
X_train, X_val, y_train, y_val =
   train_test_split(X_train_full, y_train_full,
   test_size=0.25)
ros = RandomOverSampler(random_state=42)
X_resampled, y_resampled = ros.fit_resample(X_train,
   y_train)

from sklearn.pipeline import Pipeline
pipe = Pipeline([
                 ('scaling', StandardScaler()),
                 ('reduce_dimension', PCA()),
                 ('dt', DecisionTreeClassifier()),
                ])
pipe.fit(X_resampled, y_resampled)
pipe.predict(X_val)
```

GridSearchCV

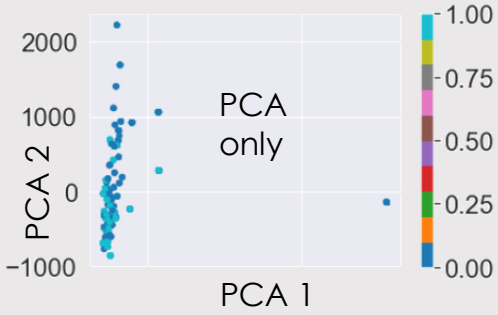
	estimator	min_score	mean_score	max_score	std_score	criterion	max_depth	n_estimators	C	penalty	kernel
13	RFT	0.53	0.61	0.66	0.06	entropy	NaN	NaN	NaN	NaN	NaN
11	RFT	0.51	0.59	0.63	0.06	NaN	NaN	3	NaN	NaN	NaN
12	RFT	0.55	0.58	0.63	0.03	gini	NaN	NaN	NaN	NaN	NaN
9	DTR	0.47	0.57	0.70	0.10	entropy	5	NaN	NaN	NaN	NaN
8	DTR	0.47	0.57	0.69	0.09	entropy	4	NaN	NaN	NaN	NaN
22	SVC	0.43	0.56	0.74	0.13	NaN	NaN	NaN	2	NaN	poly
17	LGR	0.51	0.56	0.62	0.05	NaN	NaN	NaN	2	l2	NaN
15	LGR	0.50	0.55	0.62	0.05	NaN	NaN	NaN	1	l2	NaN
10	RFT	0.45	0.55	0.60	0.07	NaN	NaN	2	NaN	NaN	NaN
1	DTR	0.48	0.55	0.64	0.06	gini	2	NaN	NaN	NaN	NaN
7	DTR	0.47	0.54	0.66	0.08	entropy	3	NaN	NaN	NaN	NaN
19	SVC	0.42	0.54	0.67	0.10	NaN	NaN	NaN	1	NaN	poly
2	DTR	0.48	0.53	0.61	0.06	gini	3	NaN	NaN	NaN	NaN
4	DTR	0.50	0.53	0.55	0.02	gini	5	NaN	NaN	NaN	NaN
6	DTR	0.46	0.51	0.57	0.04	entropy	2	NaN	NaN	NaN	NaN
5	DTR	0.46	0.51	0.56	0.04	entropy	1	NaN	NaN	NaN	NaN
20	SVC	0.46	0.50	0.55	0.03	NaN	NaN	NaN	1	NaN	sigmoid
23	SVC	0.46	0.50	0.55	0.03	NaN	NaN	NaN	2	NaN	sigmoid
0	DTR	0.46	0.50	0.54	0.04	gini	1	NaN	NaN	NaN	NaN
3	DTR	0.47	0.49	0.51	0.01	gini	4	NaN	NaN	NaN	NaN
21	SVC	0.41	0.48	0.54	0.05	NaN	NaN	NaN	2	NaN	rbf
18	SVC	0.36	0.41	0.48	0.05	NaN	NaN	NaN	1	NaN	rbf
14	LGR	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	l1	NaN
16	LGR	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2	l1	NaN

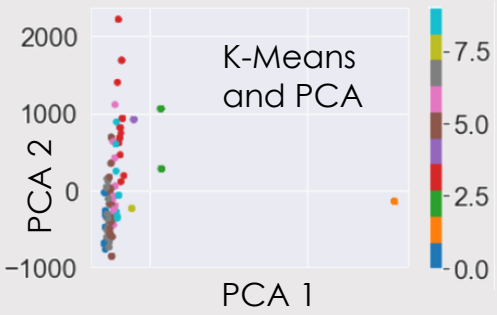
gridSearhCV.fit(X_test, y_test, scoring='roc_auc', n_jobs=-1)

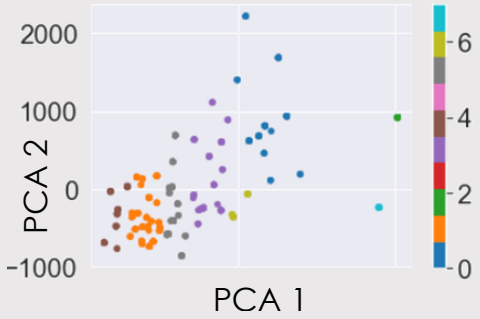
LOAN DEFAULT

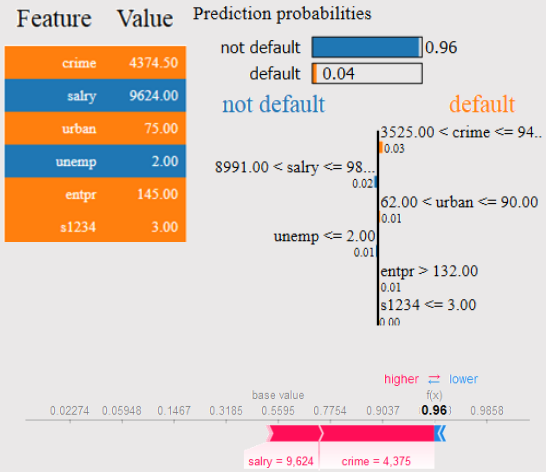
K-Means, PCA, Lime

(third appendix)









K-Means and Principal Component Analysis

Part of exploratory data analysis PCA is a methodology to project data points onto few principal components to obtain lower-dimensional data while preserving as much of the data's variation as possible. Taking the decision tree model discussed before herewith few scenario to visualize

```
from sklearn.decomposition import PCA
pca = PCA(2)
X_pca = pca.fit_transform(X_train)
pd.DataFrame(X_pca, columns=['PC1','PC2'])
      .plot(kind='scatter',
            x='PC1',
            y='PC2',
            c=y_train,
            colormap='tab10');
```

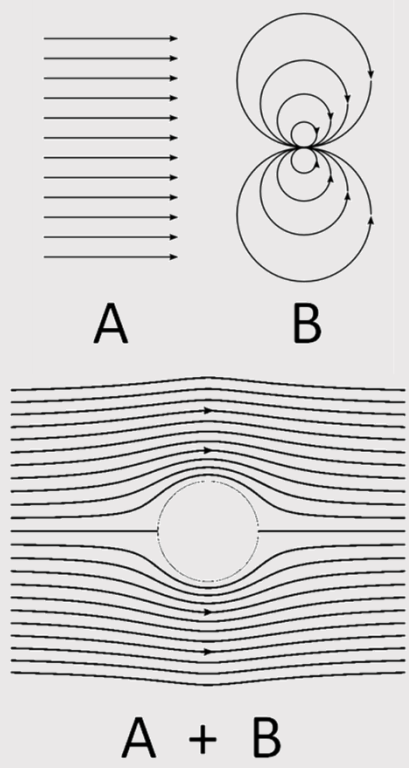
```
from sklearn.cluster import KMeans
model = KMeans(10)
clusters = model.fit_predict(X_train)
pd.DataFrame(X_pca, columns=['PC1','PC2'])
      .plot(kind='scatter',
            x='PC1',
            y='PC2',
            c=clusters,
            colormap='tab10');
```

```
X_pca_nol=np.array(pd.DataFrame(X_pca, columns=
            ['PC1', 'PC2'])[pd.DataFrame(X_pca,
                  columns=['PC1','PC2'])['PC1']<=2000])
model = KMeans(8)
clusters = model.fit_predict(X_pca_nol)
pd.DataFrame(X_pca_nol, columns=['PC1', 'PC2'])
      .plot(kind='scatter',
            x='PC1',
            y='PC2',
            c=clusters,
            colormap='tab10');
```

Model Interpretation with LIME and SHAP

Another popular tool after completing a model prior to human decision making is Lime (Local Interpretable Model Agnostic Explanations) aimed to explain predictions of any classifier and the other one is called SHapley Additive exPlanations (Shap) a game theoretic approach to explain the output of any model. In common cases these tools are trying to explain why certain data points located close to the decision boundaries did not match model prediction, what are the exceptions and what are the model arguments based on every feature importance.

VISUALIZATION



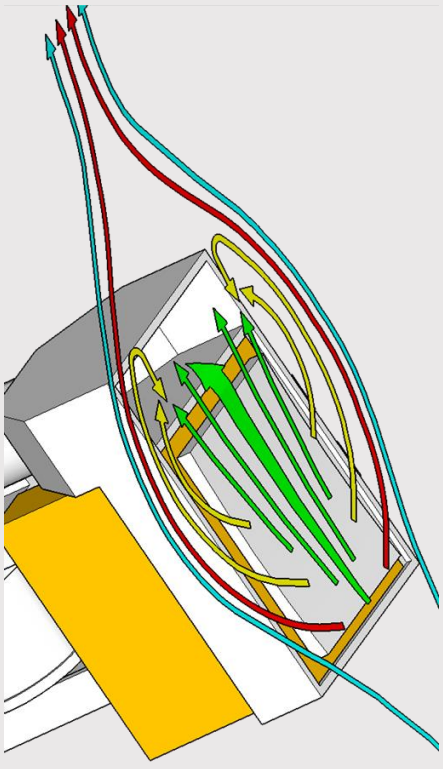
As part of visualization project I picked flow visualization with Tableau using my former personal project as an enthusiast in fluid dynamic where I use the same dataset.

POTENTIAL FLOW

The basic principle is potential flow as you can find below https://en.wikipedia.org/wiki/Potential_flow

Pictures on the left simply shows the superposition of two situation A and B. A is the simple straight flow and B is a set of source dot and sink dot located nearby and making a large bipole as shown.

Combining the two it would make streamline flow like passing a cylinder shape. It is also called Rankine Body. Rankine body was discovered by Scottish physicist and engineer William John Macquorn Rankine, is a feature of naval architecture involving flow of liquid around a body.



MORE COMPLEX MODEL

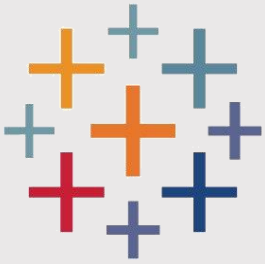
In this project we will pick the source and sink unlike a simple dot shape but more likely two rectangle shape shown below. The flow plane is actually move around a large radius but we will project the flow on a flat plane instead for simplification.

The flow will looks alike the combination of A and B above. As the sinking suction pressure is bigger than the sourcing disperse pressure the streamline after hitting the object will be narrowed aiming to conceal those escape flow (red line) to limited path to avoid contamination.

Blue lines are simply fresh flow that is just diverted as it hit the object. Yellow and green lines are the source flows which are successfully captured.

VISUALIZATION

The Motivation



Continuing my past efforts to learn about flow simulation with main purpose for simple visualization using android system I use the same dataset for tableau visualization.

MARKETING
Given the complex information which are not necessarily needed for general audience, good simple information could improve communication between researchers to end users. The harder part sometime is bringing the simple idea that anyone can understand.

PORTABILITY
Smart phone rules. We can share the idea anywhere, in the elevator, at the park, anywhere even while standing.

AFFORDABLE TECHNOLOGY
ARM processors getting faster and cheaper and smaller.

SIMPLIFIED APPROACH
Thanks to advanced graphic systems driven by gaming industry. We learn from kids to make simplified models (smoke/fire, river flow, rain drops or explosion).

BROAD APPLICATION
Flow applies everywhere. You can see it from logistic, banking, real estate, traffic, weather, health care etc.

DOCUMENTATION
As part of reporting, flow visualization is the one to seek for decision making, review on past event, education.

DATA SCIENCE
In general Visualization not only important of storytelling or project management. At Stat/math or coding/machine learning visualization does also an important role such as gradient descent illustration, neural network modeling.

TABLEAU SKILL
Using tableau and plotly Scatter 3D are the goal of this project. Tableau is like an advanced pivot table that can handle large data and display it many readily available charts and even geospatial data.

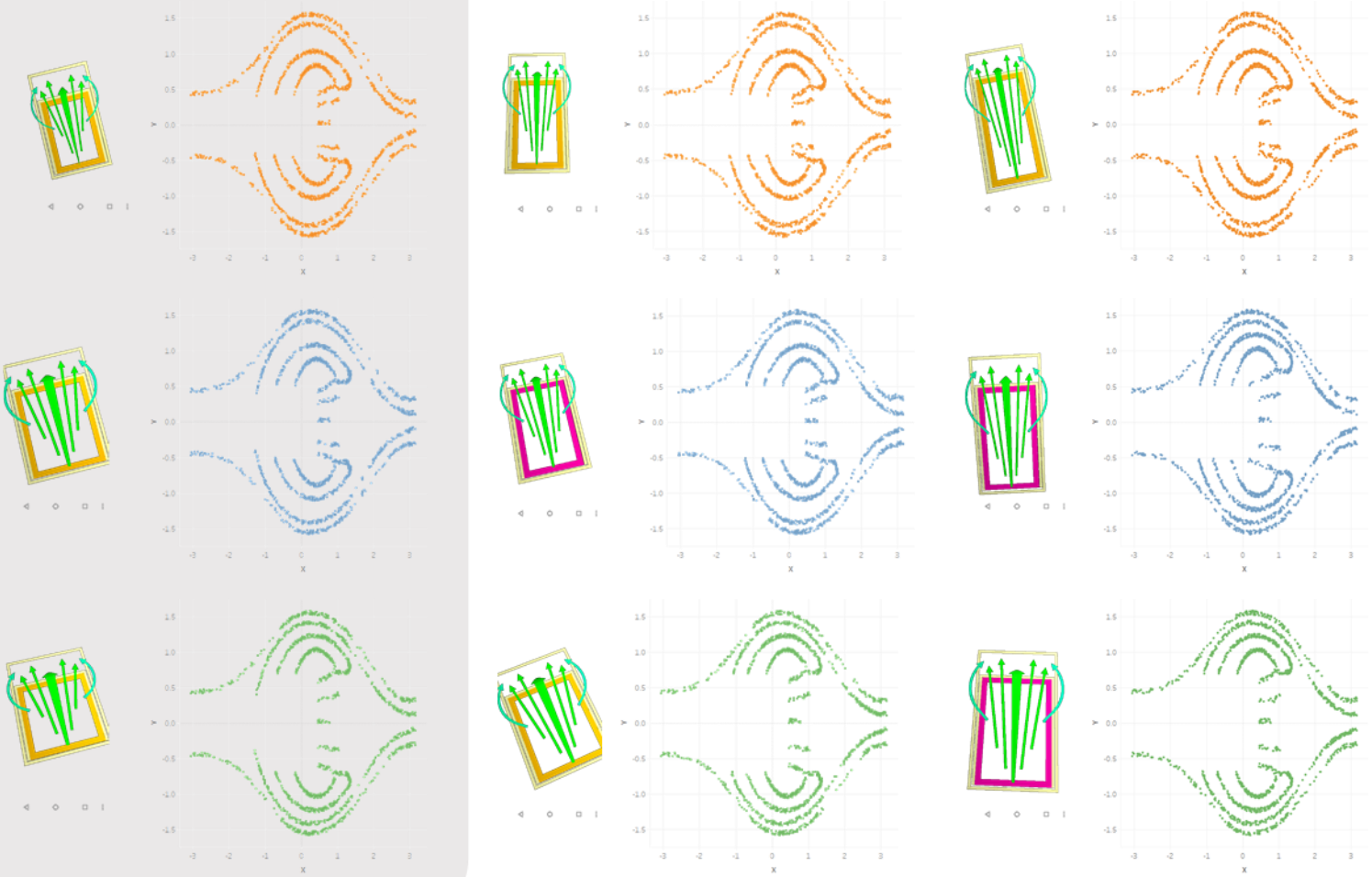
VISUALIZATION

Flow with Tableau

Combining the two screens of Tableau and Android screenshots as below we can see nine combinations of the flow distribution based on different with and length of the source. Most flows are captured by the sink and10% escaped. For Tableau desktop version it would be much easier to interpret the data while using smart phone version would require more UI/UX works to trim the data/visualized data while keeping the same weight of information to convey.

The purple rectangle indicates dummy simulations as that parameters are not a standard product offer. Possible to make but maybe costly.

The orange, blue and green flows are flows grouped by the width of the source (narrow, medium and large).



VISUALIZATION

The Data

For this project there are 9 different setup at which each consist of 1,851 row of data, total of 16,659 row of data. The coordinates were populated from polynomial curve simulated from simple fluid dynamic (potential flow) at steady state which due to comparison with lab test.

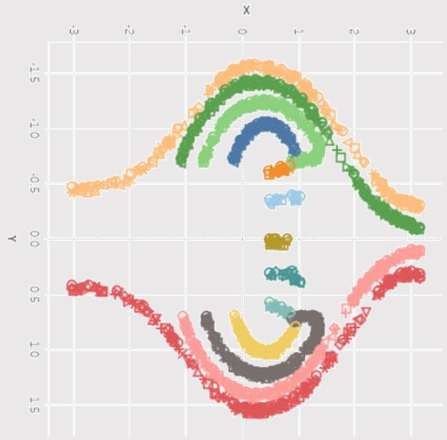
In term of the simulation and populated coordinates, it involves many hardware (PC, android phone, blackberry 10), software (Android Studio, Visual Studio) and API (OpenGLES or Vulkan) which mainly for visualization and marketing purposes. I took android operating system to show an idea cost effectively where Nvidia Shield console which also running Android OS enables it to display the visualization on the TV as well.

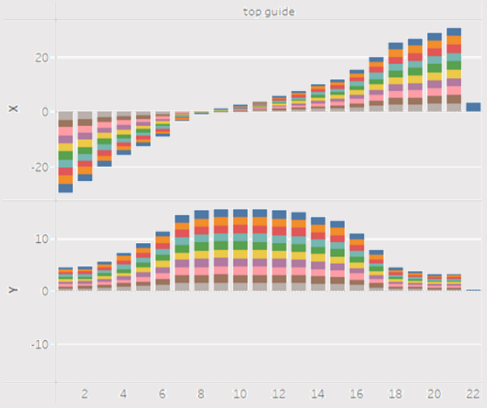
The data was originally came from Sketchup. It would be much easier to prototype the flow using this 3D tool. It was originally aimed for mobile visualization where coordinate files were exported to STL files (that store vertices, normals and colors of 3D object). Projecting the flows into 2D and using polynomial regressions OpenGL ES or Vulkan API can tweak the flow curve according to variable dimension of source width and length. Using same STL files I exported the coordinates to Tableau for this visualization project.

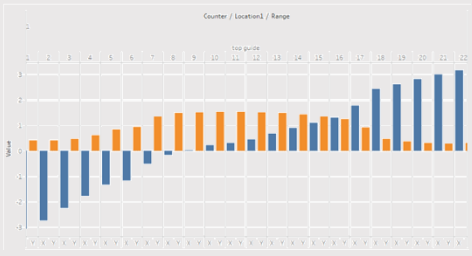
		Range													
		1		2		3		4		5		6		7	
Counter	Location1	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y
1	bottom cur..	-0.179	-0.730	-0.103	-0.824	-0.025	-0.898	0.061	-0.959	0.153	-1.007	0.251	-1.041	0.351	-1.0
	bottom flat ..	0.465	-0.355	0.865	-0.398	1.027	-0.388								
	bottom flat ..	0.462	-0.619	0.669	-0.661	0.832	-0.667								
	bottom gui..	-3.050	-0.471	-2.735	-0.471	-2.250	-0.517	-1.783	-0.653	-1.349	-0.874	-1.155	-0.997	-0.420	-1.4
	bottom out..	-1.107	-0.726	-1.004	-0.862	-0.888	-0.984	-0.760	-1.093	-0.548	-1.229	-0.311	-1.344	-0.138	-1.4
	bottom rev..	-0.701	-0.728	-0.679	-0.762	-0.532	-0.924	-0.363	-1.061	-0.249	-1.122	-0.132	-1.171	-0.010	-1.2
	middle	0.470	0.000	0.564	0.000	0.784	0.000								
	top curve	-0.140	0.692	-0.064	0.787	0.009	0.856	0.089	0.913	0.175	0.958	0.264	0.989	0.356	1.0
	top flat inner	0.476	0.302	0.876	0.291	1.027	0.388								
	top flat out..	0.476	0.567	0.680	0.609	0.832	0.667								
	top guide	-3.050	0.418	-2.735	0.418	-2.240	0.464	-1.763	0.603	-1.320	0.829	-1.155	0.934	-0.515	1.3
	top outer	-1.062	0.697	-0.963	0.827	-0.851	0.945	-0.728	1.050	-0.523	1.182	-0.291	1.294	-0.125	1.3
	top reverse	-0.658	0.695	-0.636	0.729	-0.495	0.885	-0.333	1.016	-0.226	1.073	-0.113	1.121	0.004	1.1
2	bottom cur..	-0.139	-0.737	-0.031	-0.843	-0.001	-0.887	0.125	-0.989	0.222	-0.997	0.295	-1.016	0.384	-1.0
	bottom flat ..	0.697	-0.352	0.936	-0.385										
	bottom flat ..	0.467	-0.577	0.698	-0.629										
	bottom gui..	-2.855	-0.435	-2.315	-0.507	-1.985	-0.590	-1.611	-0.690	-1.288	-0.864	-0.893	-1.138	-0.373	-1.4
	bottom out..	-1.061	-0.779	-0.981	-0.879	-0.868	-0.972	-0.591	-1.153	-0.404	-1.257	-0.299	-1.312	-0.106	-1.3
	bottom rev..	-0.694	-0.692	-0.628	-0.795	-0.502	-0.924	-0.361	-1.011	-0.237	-1.108	-0.117	-1.152	0.035	-1.1

VISUALIZATION

Analysis with Tableau







There are nine different setup depending on the width and the length of the source. I picked the first one here to analyze the coordinate of the trajectory from the thirteen layers of point of origins to their destinations.

At this case (case 1) there are only two type of flows: fresh flows (two from the top and two from the bottom) and captured flow (all the remaining layers). The six other cases has three type of flows: fresh flows, captured flows and escaped flows. Ideally there should not be any escaped flow. Given available technology the losses are well acceptable by the industry so far (may change) and as the competition increases the losses will soon be zero, not necessarily more complex, more expensive. It is team effort between the researchers and markets to enable it.

All the thirteen layers have four 4 different classes of sensor locations. Bottom flat inner, bottom flat outer, middle, top flat inner and top flat outer have three sensor location. Top curve and bottom curve have 13. Bottom guide and top guide have 22. And bottom outer, bottom reverse, top outer and top reverse have 25. Implementing random oversampling would be necessary as at some coordinate the chart would be too dense.

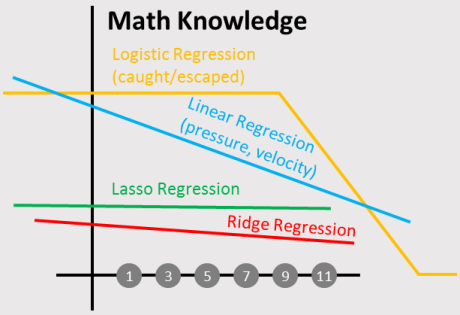
Taking 22 different sensor from the top guide curve we populated another ten random sample between the two using SMOTE. The stacked bar chart shows the ten random samples between that point and the point after. Therefore the last bar chart only has one stack as it does not have its subsequent neighbor.
The height of overall bar indicate cumulative X and Y coordinate values trying to indicate penalty not a reality that the flow disperses due to diversion of previous flow.

The last chart show almost similar ideas from each sample set without stacking X and Y values from the same sensor. As the bar chart is pretty simple Tableau combines the two in the same X-axis unlike for the stacked ones above.

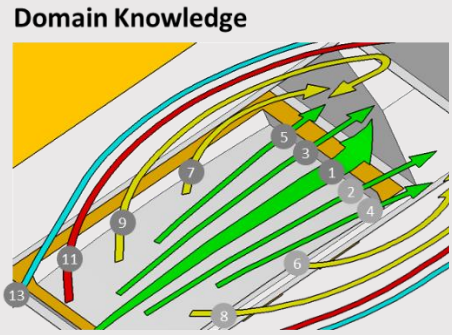
Tableau visualization for this case is good to identify missing data from certain sensor prior to comparison.

VISUALIZATION

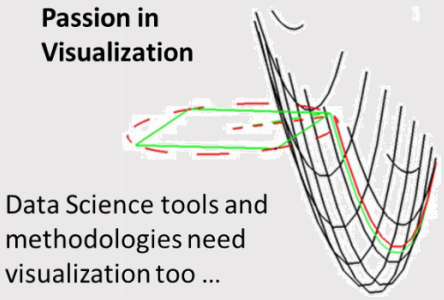
Data Science Approach



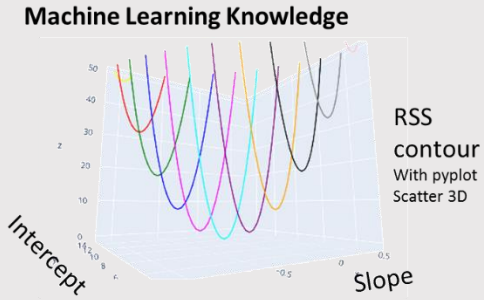
Without knowing the behavior of the flow itself, perhaps the best first step to start is understanding all number from math and statistic point of view. As you can see here I put measurement/sensors from odd numbers. At this case the flow is either caught or escaped (logistic regression) or in term of suction pressure/velocity we can illustrate the value in a linear blue line as shown. Given regularization we also can apply lasso or ridge regressions.



As the domain knowledge developed we could classify the odd and even number as illustrated in dark and light grey. It is basically a symmetrical system where 2 and 3 have the same boundaries conditions, same within 4 and 5, 6 and 7, etc. As the source dimension change (width and length) so do the flow curve where sink dimension remain the same (see page 42 for whole setup).



There is always room for improvement to visualize. Books such as An Introduction to Statistical Learning written by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani and Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow written by Aurélien Géron may give an idea. Students like me find ways to derive the concept while learning Python, Machine Learning.

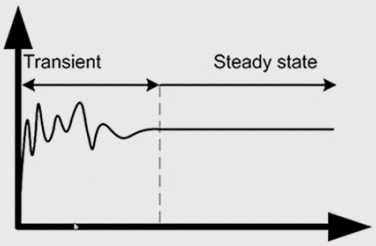


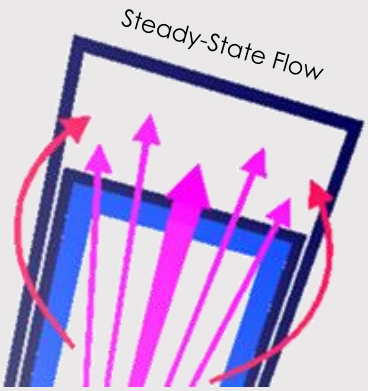
In term of regularization, I found it is interesting to combine many other affordable tools (spreadsheet. freeware 3D modeling) such as to build RSS contour (see appendix about the construction of it). Python itself has descent tool to visualize semi-finished model. It may not the best choice for executive presentation but helps the beginner as Data Science or Machine Learning to understand the concept of regularization, lambda, etc.

VISUALIZATION

Challenges

Transient Flow





Understanding the behavior of flow would require in-depth calculation and time to compute. Simplifying the visualization only for steady state flow would require less computation and less time.

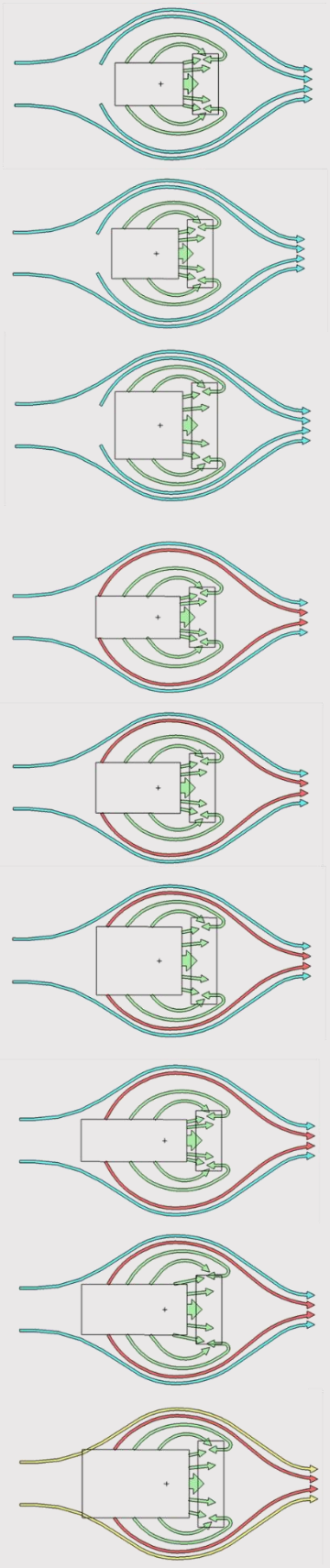
In term of transient flow, it would definitely need more than static images as flow animation (time series) is needed to differentiate one shape to the next shape in very short period of time.

Transient flow could slowly calm down into steady state flow or vise versa given the disruption to the medium. If no noises added to the system then the transient flow could be assumed steady. An analogy steady state could be driving with cruise control on a straight path and the same elevation (no uphill and downhill). While the transient one are where we are just start driving or about to stop driving.

Other challenges would be coming from Tableau as it has short free license. The other option is using PowerBI or stick to whatever Python environment could offer such as PyPlot Dash, Matplotlib, Seaborn, PyPlot Scatter 3D.

Depend on the industry and available tools given, learning visualization would be always interesting ongoing process. Given more time I would probably explore the same thing on PowerBI or Plotly Dash.

Conclusion



Flow applies everywhere. It could be supply and demand flow between the supermarket and the consumer. It could also be transmission and reception flow between cellular tower and cellphones. And it could be as well lending and borrowing flow between banks and consumers. At some points all of them may need some sort of visualization and strategy to grow their business with available technology.

I was working at night and weekends for 3 years at Walmart meat department when I landed in Canada while worked as an engineer at weekday. I applied lean thinking such as just-in-time, zero inventories and visual workplace to attract sales and minimize cost by pushing sales weekly according to weekly flyer. The bottom line is meeting corporate goal to promote business matching flow of supply and demand.

In telecom, I may say similar pattern could exist. I am not the telecom experts but pretty sure the frequency and bandwidth of 5G infrastructure has some sort of flow between the transmission towers to the receptions of final consumers both in urban and rural area. The new infrastructure is expensive to build while consumers are more likely to save money no matter how much convenient offered. There is also competing infrastructure with Wi-Fi or sky-fi making return of investment calculation complicated. I was driving a lot in rural Ontario, Manitoba and Newfoundland and we are not even having 3G yet. Some highway such as from Sault Ste Marie to Wawa in Ontario or from Dauphin to The Pas in Manitoba has no signal at all. Pretty much due to low demand and less inhabitant area.

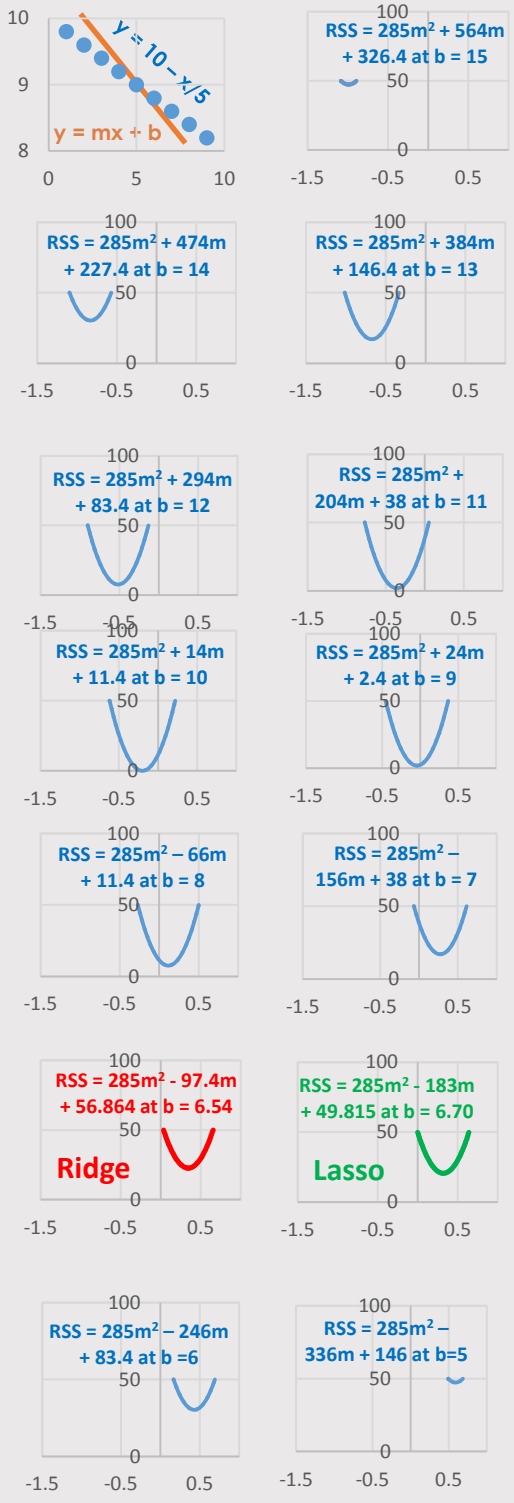
For banks, the borrowing and lending flow also getting interesting. Even before COVID hit banks are traditionally stringent to approve credit for mortgage and loans while non-bank lenders are growing exponentially. Banks are becoming less aggressive seeking for new lenders as they have been dealing with increasing default lenders.

Flow visualization is always demanding and disruptive as it is a center in decision making.

VISUALIZATION

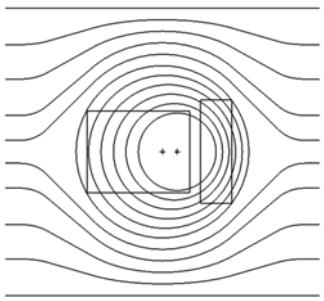
Construction of RSS Contour

(AN APPENDIX)



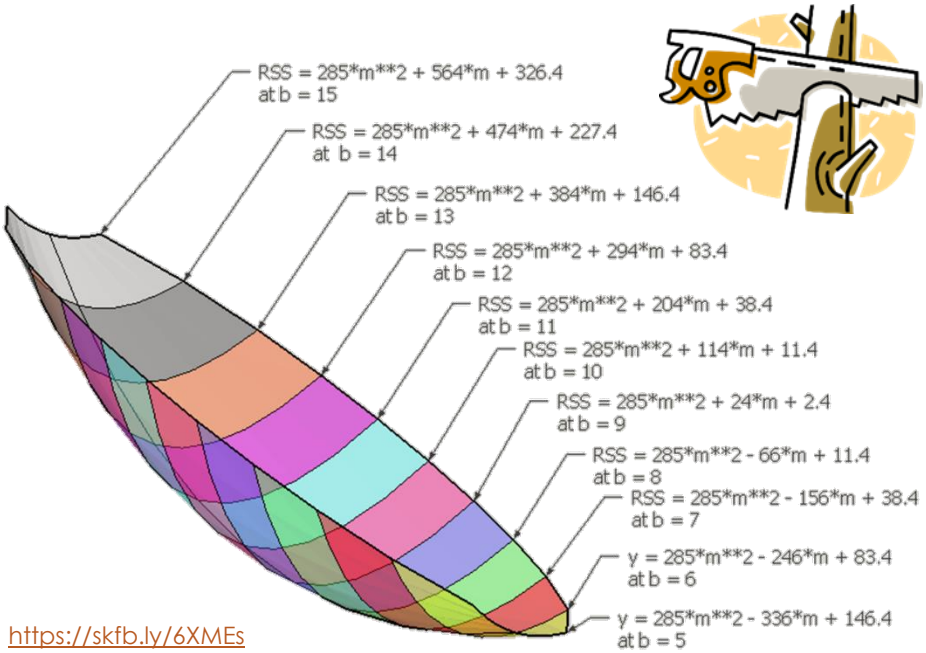
https://github.com/FariusGitHub/DataScience/raw/master/bowl2.xlsx

In term of steady state fluid flow, simple source and sink potential flow could be illustrated as shown on the right where fresh fluid flows from the left to the right while large rectangle acts as source and the small one acts as sink. Many polynomial regressions made to establish the curves.



As the next steps that may relate to flow visualization on the understanding of gradient descent, lasso and ridge regression as part of regularization I would like to bring some works on Plotly Scatter 3D regarding development of Residual Sum of Squares (RSS) contour. Residual here is difference of actual flow (lab observation) compared to curves (under fitting, over fitting, somewhere in between).

Further explanation of lasso (L1) and ridge (L2) regression mainly on the construction of RSS bowl could be found at github.com/FariusGitHub/DataScience/blob/master/Lasso_Ridge.ipynb
Terrence Parr from University of San Francisco also add an interesting idea https://explained.ai/regularization/index.html



https://skfb.ly/6XMEs
(3D Interactive model)

MORTGAGE COMPLETION MODEL

FIRST NATIONAL FINANCIAL LP

Selecting a machine learning project for portfolio was the most difficult one as it reflects an industry I like to fit in the future.

As an engineer it was very tempting to pick a technical subject (again) but not necessarily an industry I would like to work again as a data scientist. Financial and retail are at least the two industries WeCloudData has very strong client base so far. Combining the two I picked one of an achievement me and my family made lately which was paying off the mortgage. It is a combination of completing retail mortgage loan and dealing with largest (non-bank) financial lender.

Landed in Canada back in April 2008, me, my wife and my son quickly settled in an apartment close to where we live now. Thank to our realtor agent about the importance of equity buildup rather than keep paying the rent. It was a rough process as we were very new in Canada with almost zero credit history. Thank to our mortgage agent who introduced us to First National and we moved two months before my daughter was born. We started a mortgage in July 2009 and initially aimed for June 2044 completion with 35 years amortization. With many ups and downs in my wife and my careers it trained us to be smarter and spared extra cash to speed up the payment. We finally completed the mortgage in March 2019, 3.5 times faster, about a year before covid 19 outbreak and lockdowns and largest recession hit.

As I tried to build a model based on my own true story, I have abundant of test data and keen to share the simple prediction model long before I learn data science as below.

As of beginning 2018 I still had $79,634 outstanding balance.

- **Scenario 1** (for an example)
  - With 2.87 fixed rate (changed to 3.89 in July 2019)
  - Normal payment 2,972 (or $3,418 after July 2018)
  - Only one additional payments (Jan 2018)
  - Mortgage completes in 25 months (Feb. 2020)
- **Scenario 2** (real one case)
  - With 2.87 fixed rate (changed to 3.89 in July 2019)
  - Normal payment 2,972 (or $3,418 after July 2018)
  - Multiple additional pay (10 times over 15 months)
  - mortgage complete in 15 months (Mar. 2020)

See differences of the 2 scenarios in page 49.

MORTGAGE COMPLETION

The Motivation

There are many ways that motivate me to share my own story as part of my machine learning project. Beside trying to apply Scikit Learn, pandas, numpy, Keras Tensorflow and MLPClassifier neural network the essence of this portfolio is sharing the good discipline that could turn into good protection, life-long investment and productive lifestyle.

FINANCIAL FREEDOM
Being mortgage free is on track to retire early. Imagine most of our income now goes for the shelter either for a rent or a house owned. Eliminating this burden quickly is not for everyone, but everyone at some point got to do it.

JOB SECURITY
As the mortgage is prolonged, someone need prolonged source of income too. Long mortgage to complete means longer employment needed to cover. Unfortunately these days everyone change job quickly and in between there is risk of lacking of income that complicate the situation.

BIGGEST EXPENDITURE
From the start of buying the house as down payment to routine pay needed monthly for the rest always consume most of income. Less fortunate individual that relies on rent would be further away as the equity never been built up.

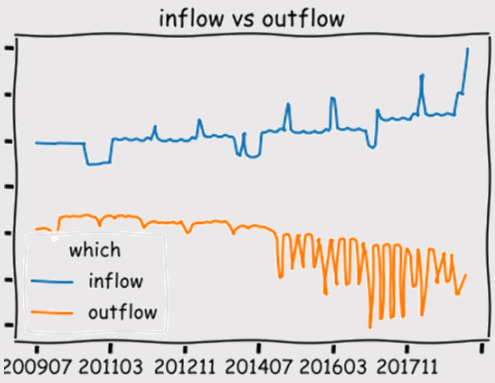
MODERN SLAVERY
Completing mortgage will improve credit score. It makes a difference in getting hired for a new job. Many employers in Canada use credit report as part of their hiring process.
https://www.fool.ca/2020/12/22/what-is-a-good-credit-score/

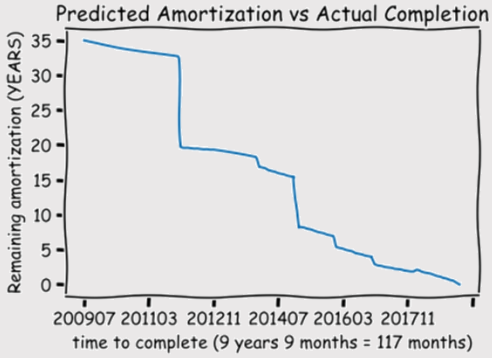
BOOKEEPING IMPROVED
Not everyone born as accountant. Completing mortgage grow you a new skill, in a way similar to junior accountant.

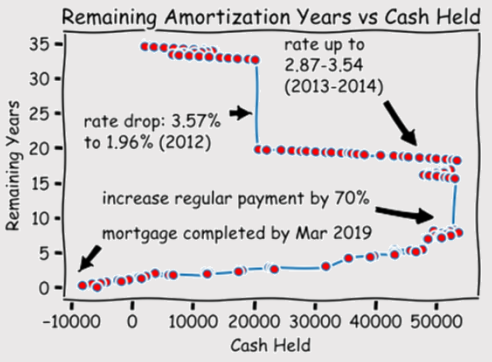
DETAILS ORIENTED
You will track everything to save. Probably another way of saying you are a proven team player and details oriented.

READY FOR RELOCATION
This is the perk earned. You could apply or buy the second one for investment and increase chance for employability.

Exploratory Analysis







Going through evidence of my finance situation through 9 years and 9 months to complete my mortgage I would like to show effects of spending, income, trend and cash helds.

Spending

Spending (orange lines) were severe as income grew by last quarter 2014 as my wife secured good job. Majority of the spending were related to mortgage repayment, nothing fancy, no extravagant lifestyle. Our focus was only one. Repaying the mortgage as soon as early 2019.

Incomes

Even our kids earn something. It was actually from Child Benefits which we invested directly to their RESP accounts. The drop of red line indicate the time I went back to school after being laid off. The two drop from blue lines indicates the situation where my wife also went back to school after left her survival job and the second drop when she was sick. Fortunately, better incomes came in after those hardships. Purple lines are tax returns we earned from our disciplines to put some money aside for RRSP and also our donations.

Amortization Left vs Time Passed

You would see similar chart in different format at feature engineering page (48), mortgage completion model page (49), conclusion page (55), next steps page (50) as this would be the most motivating chart on how well / how discipline the plan was followed. At the time when my mortgage account was still active I will get the remaining amortization every month. As my account now is closed I still can compute remaining amortization month manually.

Cash Held vs Remaining Years

Another way looking at mortgage repayment strategy is cash held during the same period of time. I remember many high interest rate saving accounts were booming when we held 50K in cash. But the tens of dollar interest we earned were not comparable to hundreds of dollars mortgage interest we paid. As I followed suggestion from my wife I bombarded mortgage balance quickly brought it down by half only in the past 2 years' time.

MORTGAGE COMPLETION

The Data

Statement period: Januarv 1. 2018 - December 31, 2018

Mortgage Account

Total Payments Received	**$62,191.67**
Principal	$60,532.43
Interest	$1,659.24
Opening Principal Balance	**$79,633.66**
Advances	$0.00
Capitalizations	$0.00
Principal Paid	$(60,532.43)
Closing Principal Balance	**$19,101.23**

Property Tax Account

Opening Tax Balance	**$(1,549.80)**
Total Tax Payments	$(3,081.00)
Interest - Earned/Charged	$(0.98)
Taxes Remitted on your Behalf	$3,079.65
Closing Tax Balance	**$(1,552.13)**

Maturity Date	June 26, 2019
Payment Frequency	Monthly
Remaining Term	6 Months
Remaining Amortization	7 months
Payment Details	
Principal & Interest	$2,972.37
Property Tax	$256.75
Credit Security Plan	$0.00
Home Warranty	$19.99
Total Monthly Payment	$3,249.11
Fees and Premiums Paid	
Funding Costs	$0.00
Administration Fees	$0.00
NSF Fees	$0.00
Credit Security Plan	$0.00
Home Warranty	$239.88
Other Fees	$0.00

Summary of transactions from January 1, 2009 to December 31, 2009

Opening Principal Balance:	**$**	**0.00**
Advances:	$	263,599.82
Total Payments Received:	**$**	**16,173.89**
Principal:	$	2,219.42
Interest:	$	3,721.12
Property Tax:	$	1,209.24
Life Insurance:	$	310.30
Fees:	$	8,713.81
Closing Principal Balance:	**$**	**261,380.40**
Interest Accrued to December 31, 2009	$	119.44
Opening Tax Balance:	**$**	**0.00**
Total Tax Payments Received:	$	1,209.24
Interest - Charged/Earned:	$	0.25
Taxes Paid on your Behalf:	$	0.00
Closing Tax Balance	**$**	**1,209.49**

Payment Frequency:	**Monthly**
Maturity Date:	**June 26, 2014**
Remaining Amortization (months):	**414**
Interest Rate:	**Maximizer Plus plus 0.60%**
Term of Loan:	**5 Years**
Principal & Interest:	**Variable**
Property Tax: $	**201.54**
Life Insurance: $	**62.06**
TOTAL PAYMENT	**Variable**

The majority of the raw data accounts.csv was built from 4 separate accounts. 2 Chequing accounts from my wife and myself (TD and Simplii Financial), Mortgage accounts (First National) and last but not least my credit card account (Walmart Rewards Master Card).

Mortgage account would be the shortest as it only issue 10 statements for my whole mortgage liefecycle. I would then need to split them into 117 separate deductions as monthly payments and additional payments happen every 26 of the month. First National always deduct twice if that particular month I made both regular and additional payment. Spitting the mortgage payment by month is based on prorated figure from annual statement.

From TD accounts, the inflow would be her income from Tim Hortons until end 2013 before went to school (no income until August 2014), OTMH income from Aug 2014 and additional CVH income from early 2019. She was unable to work due to sick by early 2017. For the outflow there are basically all the utility bills (water, hydro, gas), her own RRSP, some portion of kids RESP, condo fee, donations, kumon and life insurance.

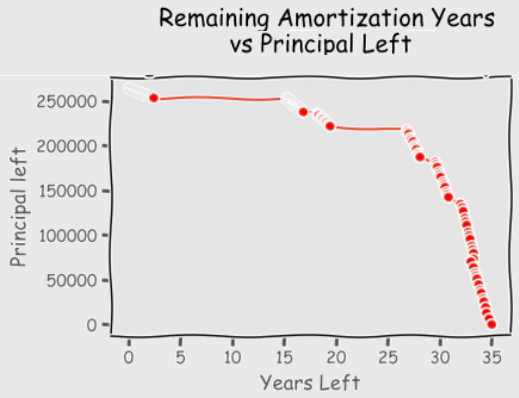
For Simplii accounts, the inflow would be my income from GE until I was laid off in Sept 2010 but continue my survival jobs at Walmart until April 2011. Mersen salary started from April 2011. CCB and all tax returns went to my accounts. As I worked in sales role I have reimbursement going to my account (temporary as I need to pay the credit card). For the outflow I was responsible to pay all regular and additional mortgage pays, my own RRSP, some portion of kids RESP and my wife car insurance (my corporate car insurance paid by the company).

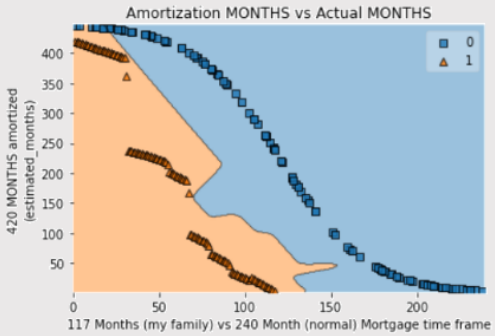
For the Walmart account, it was basically categorized into business and personal. Looking at personal rows it included internet bill (Acanac), clothing, dining out, groceries, electronics, phone bill (Freedom, Koodo), gifts, hardware, Netflix, pharmacy, extracurricular (swimming) and vacation.

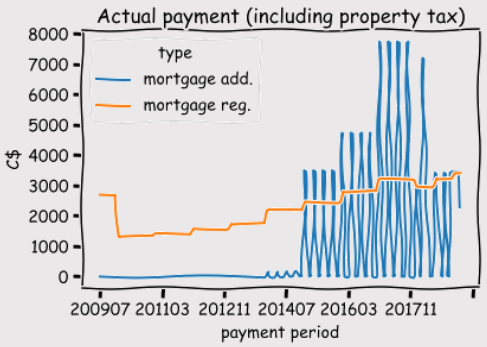
account.csv has 16,443 where 12,093 is related to business. Seems like a hassle but it is part of net cash noises happen to me all the time. The reward is ok as I earned points from Walmart helping me out to buy gift/grocery which I opted out.

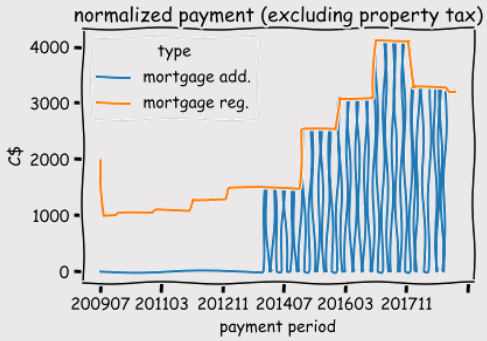
MORTGAGE COMPLETION

Feature Engineering









The larger part of feature engineering for this project is getting amortization months left calculated from external site, such as https://www.calculator.net/mortgage-payoff-calculator.html?cunpaidprincipal=3183.54&cmonthlypayment2=3204.86&cinterestrate2=3.89&cadditionalmonth2=0&cadditionalyear2=0&cadditionalonetime2=0&cpayoffoption2=original&type=2&x=0&y=0#monthlypay which at this final stage the remaining term stays at 1 month and I have to clean the data by convert it to zero. www.calculator.net/mortgage-payoff-calculator.html needs few parameter to calcite which I gave them in the raw csv file. As I build the url with those parameters needed I **web scrap** the page seeking for years and months needed to payoff and stored to new feature (amort) and decimal type of it (remain).

I also added two other features Column time117 tries to digitize period of Jul 2009 to March 2019 as 117 different months which has value of 1 to 117. Column time420 tries to digitize remaining amortization in months, starting from 35 years or 420 months until it reach zero, so it has values 0 to 420.

The important dashboard for this model is shown in the first chart above which was developed from all of the above steps.

The second chart above was built from the first one with an additional sigmoid-alike curve representing the most popular mortgage completion (20 years). In this chance I also would like to present the visualization of data leakage. As I choose tanh activation this algorithm is very keen to find loopholes from its surrounding as you can see a spike on the right. The solution would be **random oversampling**. In this case the 20 year mortgage simply follow the sample population. If sigmoid was made into 1000 row ROS will increase my 117 mortgage row into 1000 too. At that point sigmoid curve will be denser and likelihood to have a spike is lower nor the **leaking data**.

The last two charts are kind of ideal and simplified data I should take into consideration when deducting my net cash. For web scarping purpose I use the last one ('mortpay' feature) and property tax does not do anything to amortization. For monthly net cash calculation (inflow – outflow) I use the second last one as it was the actual money deduction from my bank account ('how much' feature).

MORTGAGE COMPLETION

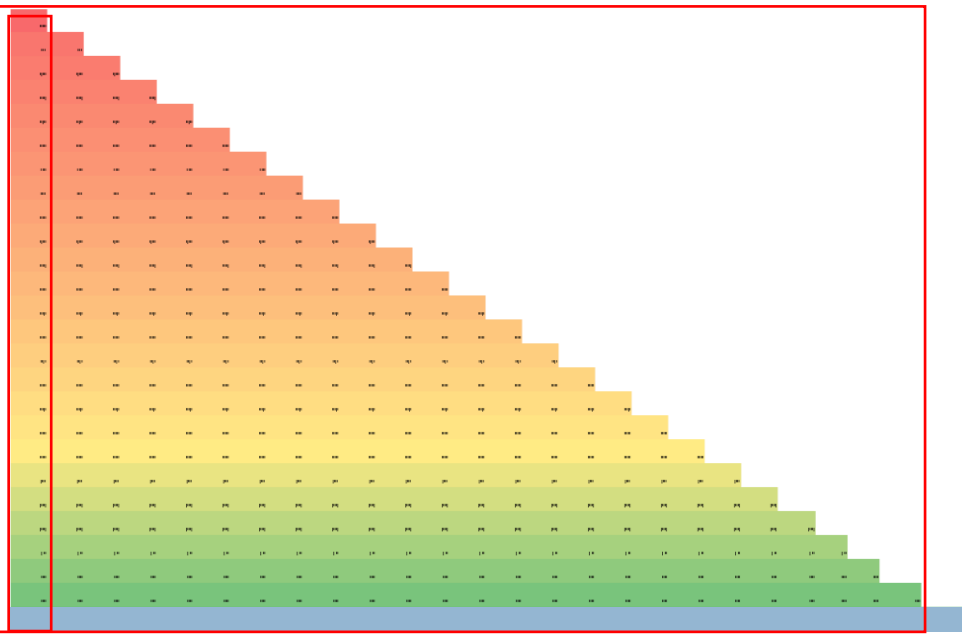
Mortgage Completion Model

After best models selected and validations are accomplished, it is now to visualize the model. Below chart is recommended to monitor plan and stick or improve it depending of savings level.

Scenario One (25 months to complete)
Principals that remained seen on the left side of triangle
End of the road never change, no additional pay made.

With only one additional pay in 2018

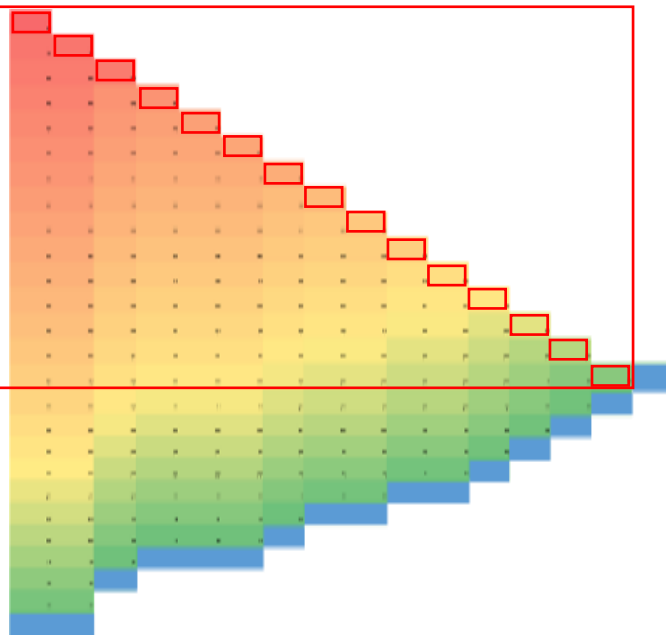
		principal	interest	reg-pay	oth-pay
2018	1	79,633.66	190.46	2,781.91	2,972.37
	2	74,069.83	177.15	2,795.22	
	3	71,451.77	170.89	2,801.48	
	4	68,821.17	164.60	2,807.77	
	5	66,178.00	158.28	2,814.09	
	6	63,522.18	151.92	2,820.45	
	7	60,853.66	197.27	3,220.96	
	8	57,829.97	187.47	3,230.76	
	9	54,786.67	177.60	3,240.63	
	10	51,723.65	167.67	3,250.55	
	11	48,640.76	157.68	3,260.55	
	12	45,537.89	147.62	3,270.61	
2019	1	42,414.90	137.49	3,280.73	
	2	39,271.67	127.31	3,290.92	
	3	36,108.05	117.05	3,301.18	
	4	32,923.93	106.73	3,311.50	
	5	29,719.16	96.34	3,321.89	
	6	26,493.61	85.88	3,332.34	
	7	23,247.15	75.36	3,342.87	
	8	19,979.65	64.77	3,353.46	
	9	16,690.96	54.11	3,364.12	
	10	13,380.94	43.38	3,374.85	
	11	10,049.47	32.58	3,385.65	
	12	6,696.40	21.71	3,396.52	
2020	1	3,321.59	10.77	3,407.46	
	2	- 75.10	- 0.24	3,418.47	



Scenario Two (15 months to complete)
Principals that remained seen on upper side of triangle
End of the road shortened as additional pays were made.
github.com/FariusGitHub/DataScience/blob/master/Mortgage.ipynb

With actual additional pays

		principal	interest	reg-pay	oth-pay
2018	1	79,633.66	190.46	2,781.91	2,972.37
	2	74,069.83	177.15	2,795.22	
	3	71,451.77	170.89	2,801.48	7200
	4	61,621.17	147.38	2,824.99	2,972.37
	5	55,971.19	133.86	2,838.51	
	6	53,266.55	127.40	2,844.97	
	7	50,548.97	163.86	3,254.36	3,418.23
	8	44,040.24	142.76	3,275.46	2100
	9	38,807.54	125.80	3,292.42	
	10	35,640.92	115.54	3,302.69	3,418.23
	11	29,035.54	94.12	3,324.10	
	12	25,805.56	83.65	3,334.57	3,418.23
2019	1	19,136.42	62.03	3,356.19	3,418.23
	2	12,424.04	40.27	3,377.95	3,418.23
	3	5,668.13	18.37	3,399.85	2,286.66
	4	-	-	3,418.23	

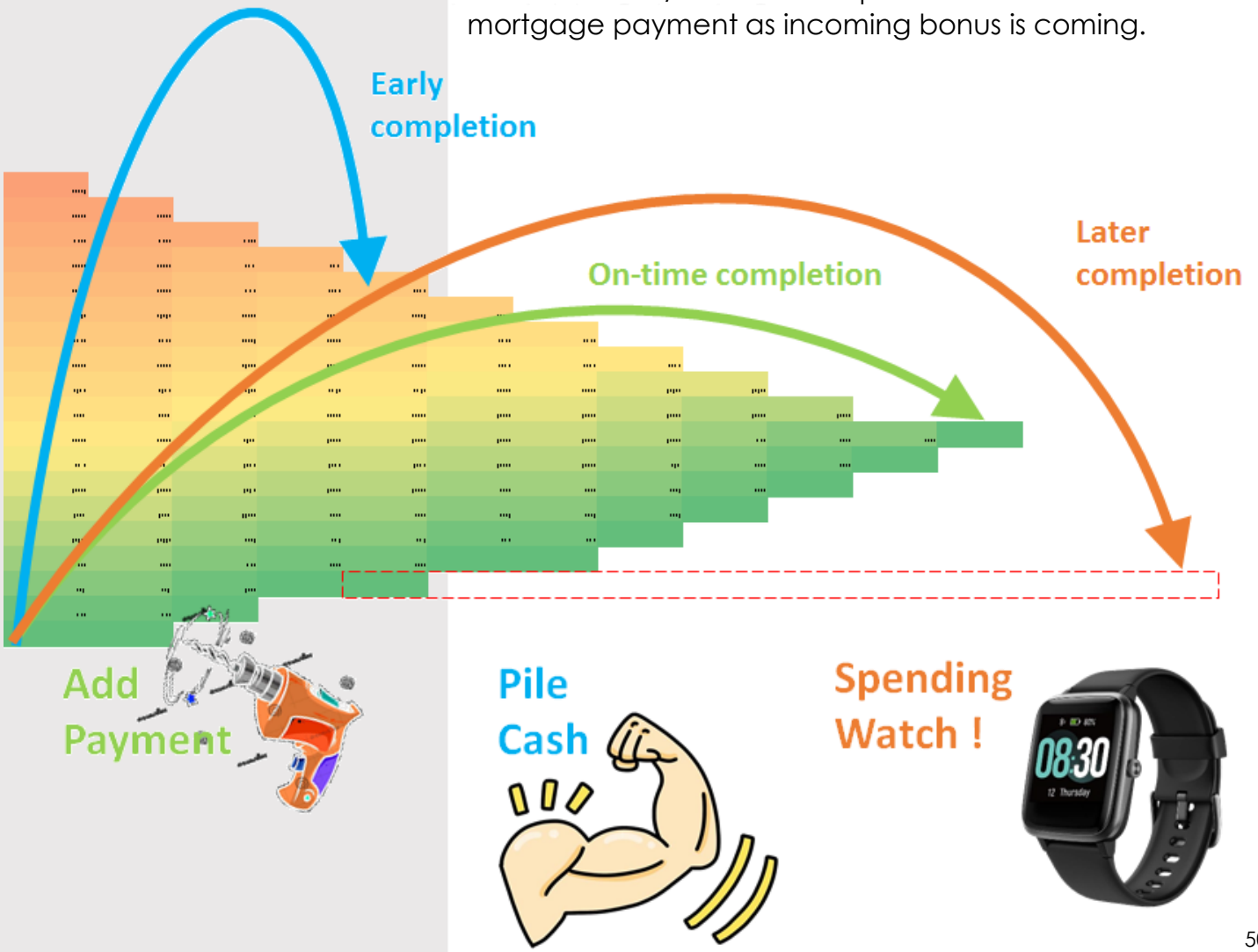


NEXT STEPS

My suggestion for the next step is more related to data engineering which related to capability of mobile decision making with devices such as Android or Apple watches.

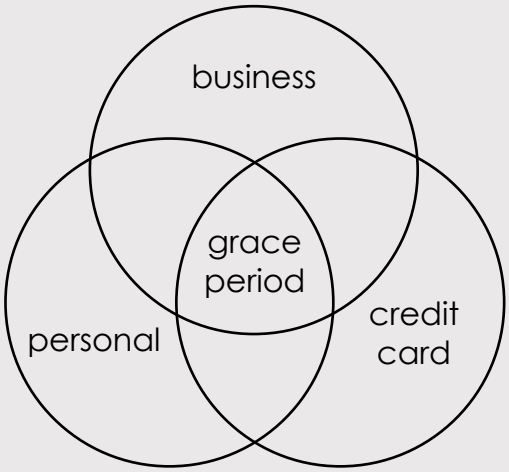
The majority of hold up from speeding up the mortgage payment probably is lacking of cash held which is caused by less sensitive spending or extravagant lifestyle. Some kind of reminder could be helpful rather than just switching credit card to debit card aiming to limit ourselves bluntly.

Using Android/iOS platform we could mimic autonomous driving, internet of things. What I am aiming is autonomous spending, internet of accounts where saving, chequing and loan accounts has its own sensor act as live features feeding latest data for machine learning model to predict whether spending that $200 for shoes still allow someone on track for his/her vacation plan and accelerated mortgage payment as incoming bonus is coming.



MORTGAGE COMPLETION

Challenges



Picking this subject as machine learning project would come at more time to spend building almost real data. My online bank account did not keep back to 10 years ago but I use my memory about past expense to mimic my family situation, saving effort and extra payments.

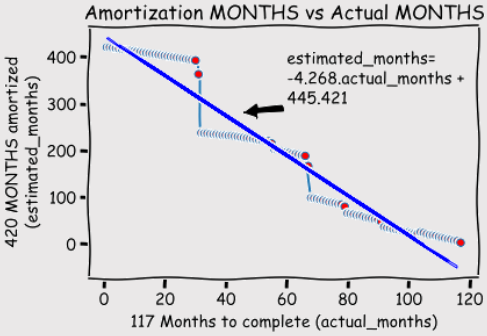
It was a pain given a week long to prepare machine learning presentation while most of my time spent on building the dataset which is around 16K rows including business transactions I managed to reimburse them on time which did not make net worth increased. It helped to be a buffer between 20 days grace period before I need to make credit card payment where I probably could use the fund temporarily to shoot the mortgage and having other source of incoming income to offset this temporary inter-borrowing (tax return, bonus, etc).

Other challenge to bring my own story is keeping all confidential data confidential, such as my salary, my wife salary while the idea is only showing the incoming and outgoing cash not necessarily show exact number during my presentation to the class.

Last but not least maybe the challenge in exploring the data part where most of the money saver know well. It is all about saving money. The money saved will be used to speed up mortgage payment. I did some exploratory studies anyway mainly having same x-axis (actual running months) but with different y-axis (drop in amortization month, total income, net cash). They might be lot of correlations too between those y-axis (more net cash may reflect more drop in amortization month, more total income may reflect to more net cash) but I found it less valuable and tried to limit the discussion towards more motivating dashboard which is amortized months left vs actual months running so far as the time was limited and knowing I spent most of the time building that 16K rows data.

MORTGAGE COMPLETION

Conclusion



Hi Farius,

I am really glad that you would be mortgage free next month. Nobody gets mortgage free without self and financial discipline

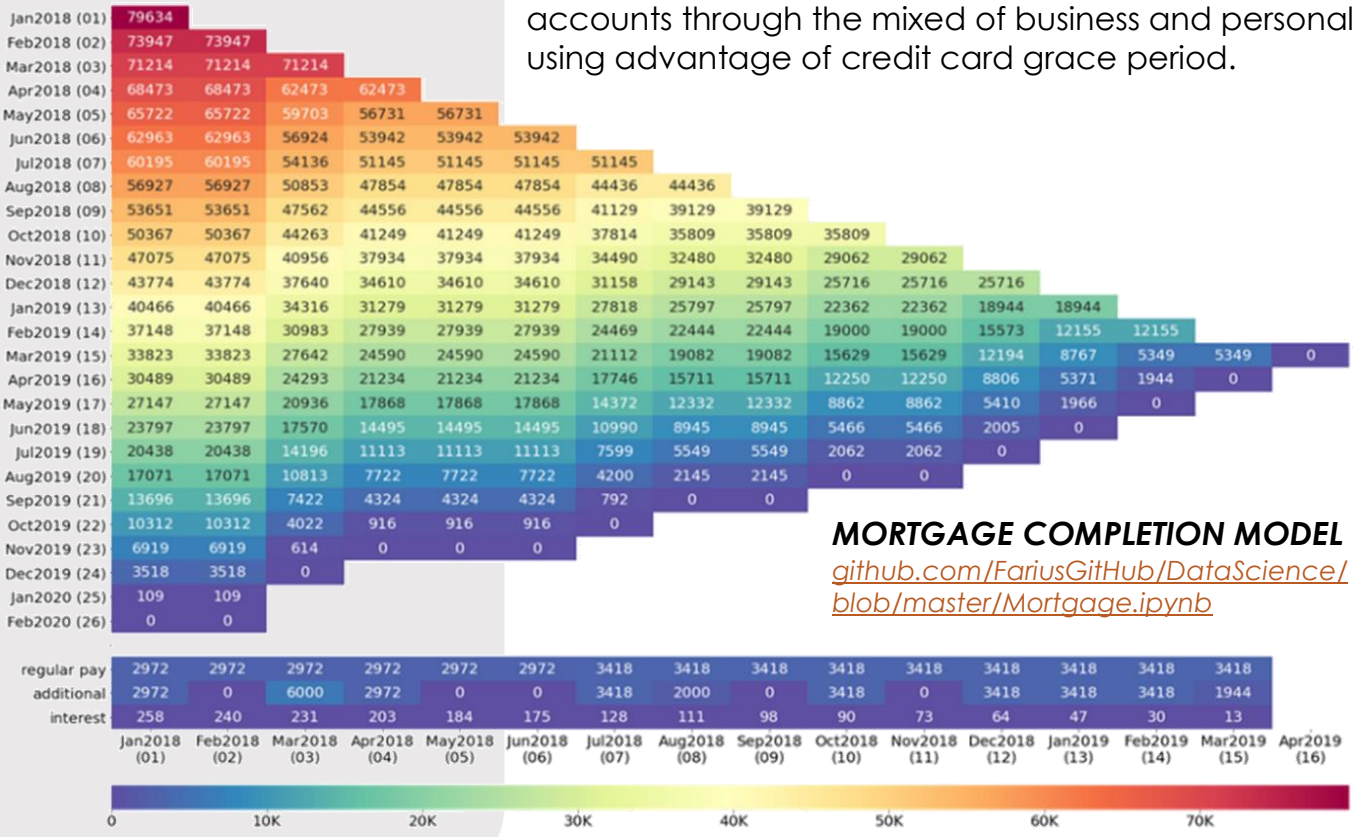
Saurabh Garg, First National Mortgage

It could be argued that the most important feature to complete the mortgage is only one. To save. But of course to go there it would require discipline, many decision trees in everything we buy or not to buy every day. In the left chart it just shows a correlation of amortization months left based on the time passed. It consisted two times either myself or my wife did not work for 7 months period for each of us and an additional for three months or so she was sick and could not generate income.

Comparing my case to others I am taking decision region plot shown on feature engineering page (48) with 117 data points and random sample from others says those 20-year mortgage. MLPClassifier shows prediction whether test data set would fall into 10 year-mortgager or 20-year mortgages with below parameter

```
nn = MLPClassifier((20,10, 2),
     activation='tanh', max_iter=130000)
```

Both linear regression and MLPClassifer suggests I did not have any problem completing the mortgage by March 2019 but it was not the case. Looking at exploratory analysis page (46) you could see that I could potentially hit negative net cash (-10K) if I did not reshuffle my accounts through the mixed of business and personal using advantage of credit card grace period.



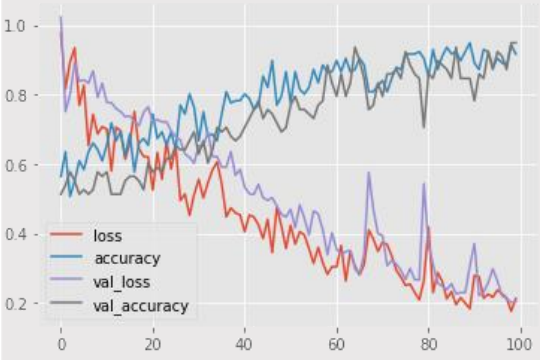
MORTGAGE COMPLETION MODEL

github.com/FariusGitHub/DataScience/blob/master/Mortgage.ipynb

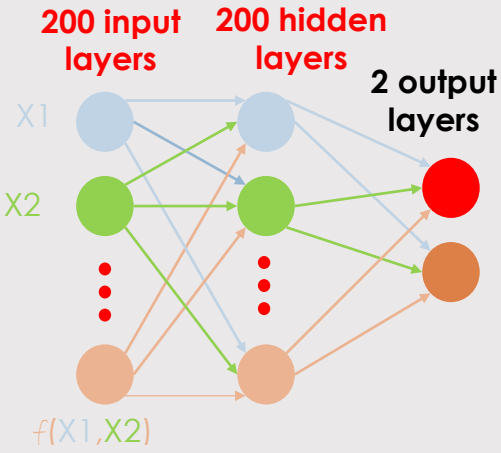
MORTGAGE COMPLETION

Keras TensorFlow

(An APPENDIX)




```
pd.DataFrame(history.history).plot()
```



Considering more feature in the context of neural network and deep learning, I would like to show one possible fine-tuned model that predict likelihood of mortgager like my family to complete loan less than ten years.

```
In  []: X_train, X_test, y_train, y_test =
        train_test_split(X, y, test_size=0.33,
        random_state=42)
        lr = LogisticRegression()
        lr.fit(X_train, y_train)
        lr.score(X_test, y_test)
Out []: 0.987

In  []: model = keras.models.Sequential()
        model.add(Dense(200, activation="tanh"))
        model.add(Dense(200, activation="tanh"))
        model.add(Dropout(rate=0.5))
        model.add(Dense(2, activation="softmax"))
        model.compile(optimizer='adam',
        metrics=['accuracy'],
        loss='sparse_categorical_crossentropy')

        history = model.fit(X_train, y_train,
                  epochs=100,
                  validation_data=(X_test, y_test))
```

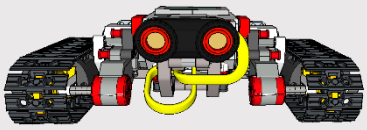
Fine tuning the input and hidden dense few times it seems the accuracy will stabilize at 200 each at tanh activation for both. Slight difference from **lr.score**, **model.evaluate[1]** and **y.pred[0]** may come from simplified X dataset which only feed two unique features where tensorflow need to populate 198 other derivatives from these two to make 200 input layers.

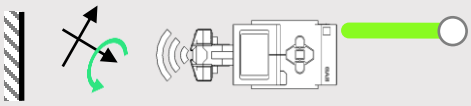
```
In  []: model.evaluate(X_test, y_test)
Out []: [0.128, 0.949]

In  []: y_pred = model.predict(X_test)
In  []: pd.Series(y_pred[0]).
            sort_values(ascending=False)
Out []: 1    0.998
           0    0.019
           dtype: float32
```

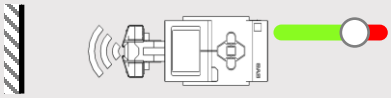
X has X1 and X2. Beside amortization months left (X1) and actual months passed (X2) we could get more features (feature engineering) such as principal left, net saving, etc.

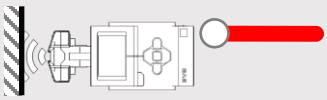
This model says 2-5% from test suggest I would not be on track.

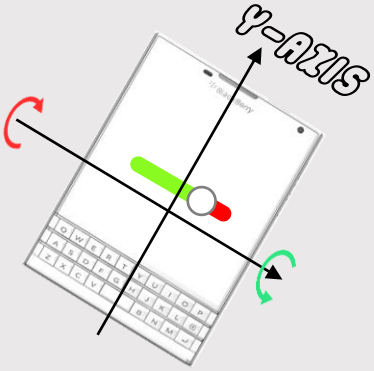
AUTONOMOUS DRIVING with LEGO S3 and SPARK











SELF DRIVING CAR THESE DAYS

Nowadays phone GPS system, and accelerometer sensors are used by many insurance companies like Desjardins and Intact to track clients driving behavior in exchange for 15% discount.

We also noticed from 2017 most new cars, especially Honda and Toyota sold standard feature of semi-automated collision detection by connecting cruise system with ultrasonic sensor to follow or avoid object upfront by adjusting the speed.

In my opinion it would not take very long for insurance companies not only tracking individual driving behavior but also its interaction to other cars it follows in term of safety distance, anticipation to stop safely, with focus, no distraction regardless semi-automated vision is available or not.

BEYOND CARS

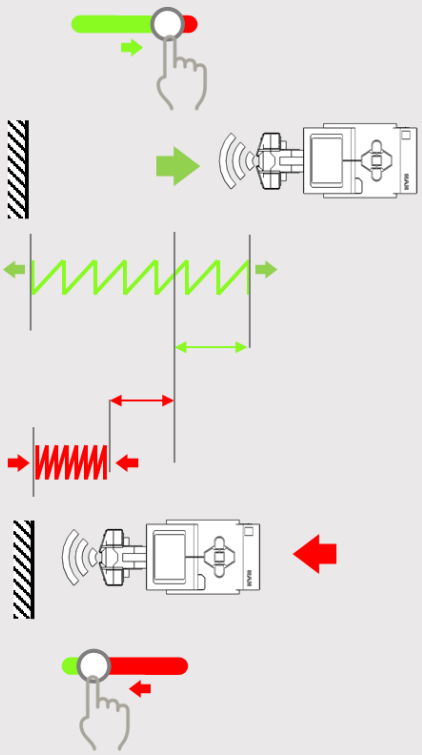
Applying sensors, internet of things (IOT) principles as we know are not limited to engineering, manufacturing and highly technical devices. It is already in our home such as Alexa, Google Home, Samsung Smartthings with AI, NLP assist us optimize hydro bills, close and open garage/doors, sustain entire home security and efficiency.

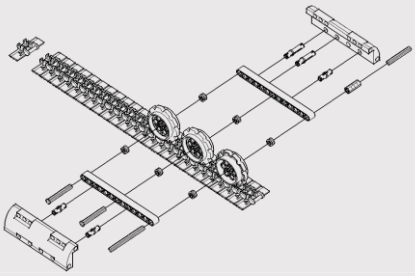
Inspired by deep learning on vision systems presented by Rhys Williams in couple guest lectures I am also keen to bring my past hobby on robotics to big data context for my final portfolio. You could imagine similar interconnected sensory data stream on other fields such as finance, loan default, mortgage completion as previously presented in this portfolio.

Imagine some sort of sensors also attached to our credit cards to trigger overspending and send alert for pension or saving accounts, advise extra contribution to maximize tax returns. Using web scraping, proper cyber security, legal compliance and machine learning technique this long existing feedback and control systems widely used in avionics and robotics not only helping individuals more cautious on daily spending or saving but also helping to build better credit score where at the end helping them landing a new job as many Canadian companies take seriously credit score when hiring candidate.

AUTONOMOUS DRIVING

The Motivation





Picking up the last project for this diploma program would be an interesting one. The reason I choose this AI project was based on many of my interest in the past, hobby and volunteering work at school where I am keen to recompile the story from data science and big data point of views.

Internet of Things
Part of the big changes in life in couple years would be interconnected home to our phones. Internet of things come standard into Lego robot even at debugging stage.

Affordable Experiment
The community of Lego Education enthusiast is getting bigger. The price gets affordable where sometime the wholesaler could even give steeper pricing.

Promoting STEM
It is still part of my passion promoting STEM, not necessarily for tech but encouraging critical thinking which would bring inspirations in art, social sciences and leadership.

Introduce Leadership
Patience and enthusiasm might be needed in whatever we do. I found every part of it represent leadership traits. Like we learn foreign language from mother tongue, kids could learn leadership from this Lego project as well.

Already here, 4 years ago
In term of semi self-driving cars, yes it is already on the road. Simply go to nearest rental car and find the artificial intelligence already come standard, at least since 2016.

Not Limited only to cars
The most important part of this project is seeking similar application to non-tech areas such as personal finance. This is an area where bank hesitates. As financial literacy is increased/automated banks could start losing revenues.

Reprogrammable
Another characteristic of autonomous vehicles is that the core product will have a greater emphasis on the software and its possibilities, instead of the chassis and its engine. This also implies that autonomous vehicles with machine learning are never finished because the product can continuously be improved. https://en.wikipedia.org/wiki/Self-driving_car

Data and Storage

App inventor is used to collect data. Using integrated development environment web application originally provided by Google, now maintained by Massachusetts Institute of Technology there are 4 choices to store data.
http://ai2.appinventor.mit.edu/reference/components/storage.html

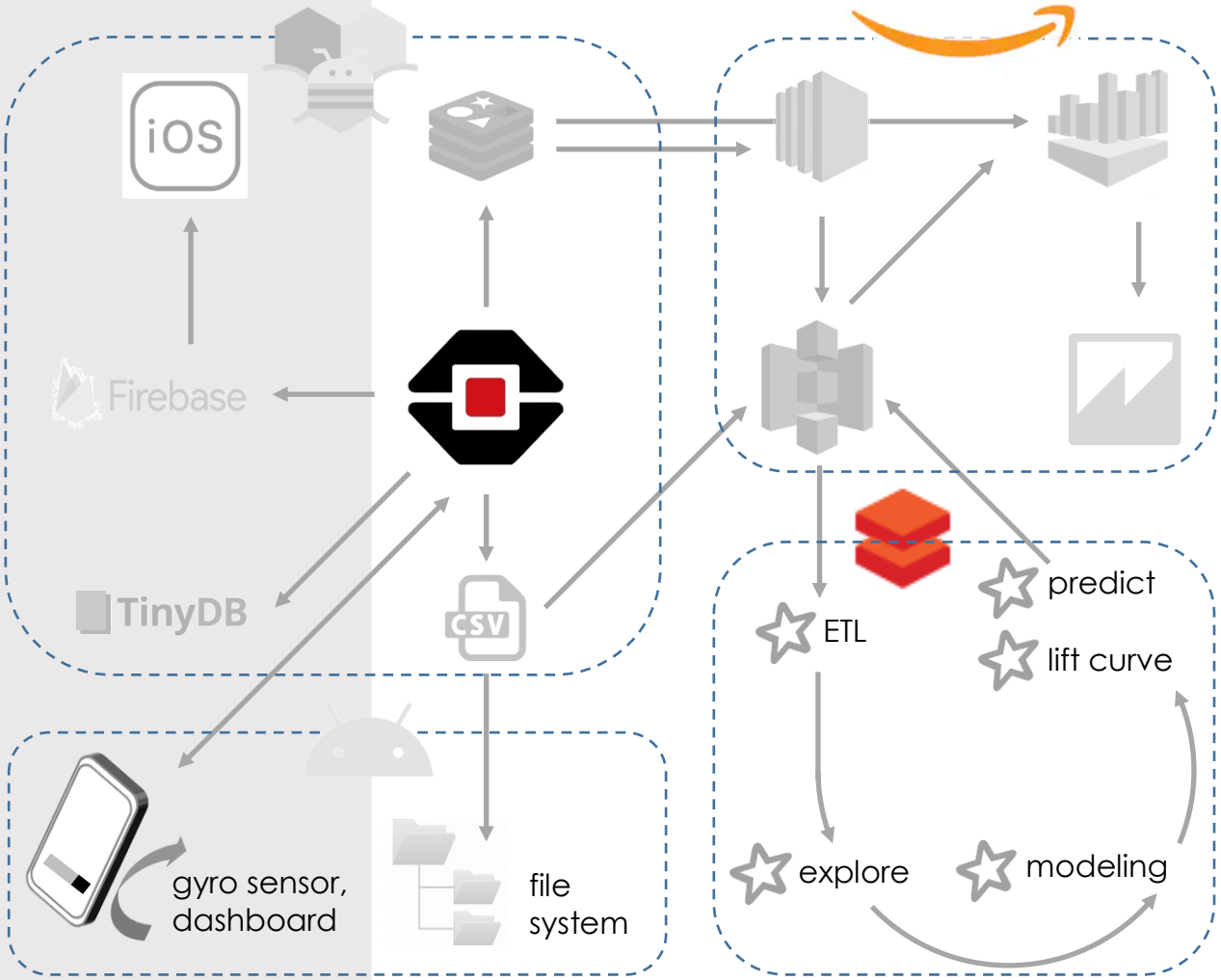
The most popular two would be using CloudDB which is actually based on Redis cloud memory-based storage or simply saving the data into a smart phone local directory.

There are two options to backup Redis data

1. Load the RDB into a Redis - using Redislabs import
2. https://github.com/sripathikrishnan/redis-rdb-tools

On next page options 2 will be elaborated to collect data

Normally within 2 minutes interval EV3 can send 300 rows of data with 20KB text file. To reach 100K rows it needs 12 hours and the file will be 6MB which consists of 8 columns.



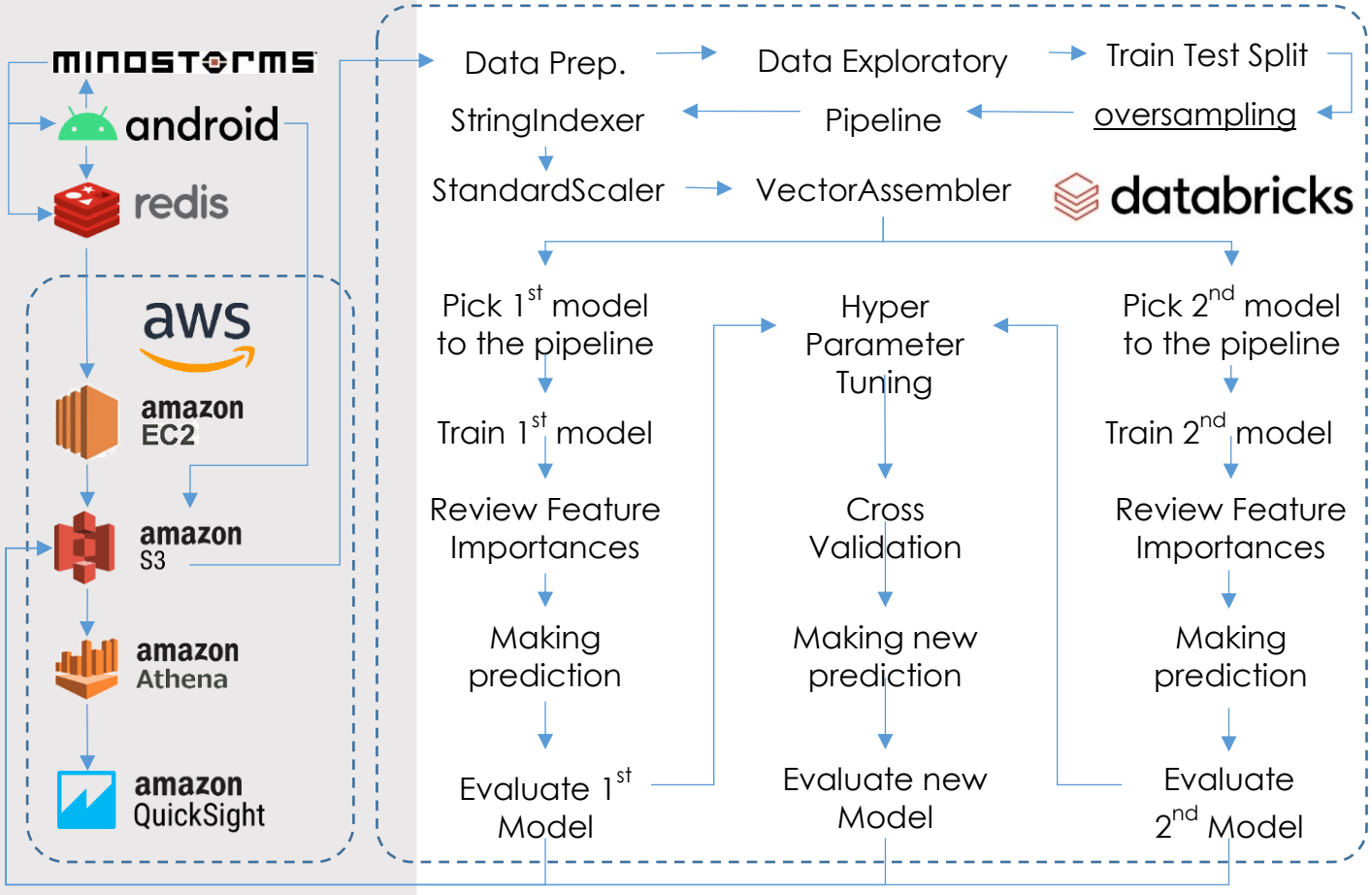
AUTONOMOUS DRIVING

Methodologies

INITIATION
The data was originated from Lego Mindstorms robot controlled by android phone where the distance to the obstacle and force computed in between were stored into Redis database or locally into phone storage.

MACHINE LEARNING
databricks got the data from S3 or from Android making regular preparatory, exploratory, test/train split, ROS, modeling, prediction, validation, simple visualization and send the prediction back to S3

STORAGE / VISUALIZATION
AWS environment such S2, EC2, Athena and Quick Sight would help storages and visualization process. Lego data send to Redis was stored in EC2 memory by default. After receiving the prediction Athena will compile the S3 data for visualization with Quick Sight.



AUTONOMOUS DRIVING

Data Collection

ai2.appinventor.mit.edu

Designer > Palette > Viewer > Properties

https://app.redislabs.com/

Database > View Database > Configuration

Amazon S3 > farius-redis1 > Edit access control

Below is an example of data collection from App Inventor to databricks through redis paid account that enable rdb backup to AWS S3 through S3 ACL (other AWS account).

Storage

- **CloudDB**
- File
- TinyDB
- TinyWebDB
- FirebaseDB

Properties

CloudDB1

RedisPort 16582

RedisServer

redis-16582.c114.us-east-1-4.ec2.cloud.redislabs.com:16582

Token

J0TsiqaOYJoBRrKYDA5Be7hFxPv5

Subscription

#1318886 Essentials/AWS/us-east-1/Standard/100MB

Protocol

Redis

Endpoint redis-16582.c114.us-east-1-4.ec2.cloud.redislabs.com:16582

Access Control & Security

J0TsiqaOYJoBRrKYDA5Be7hFxPv5AM0W

EDIT ACCESS CONTROL LIST (ACL)

Access control list (ACL)

Grant read/write permissions to other AWS accounts

Access for other AWS accounts

Grantee	Objects	Bucket ACL
fd1b05415aa5ea3a310265ddb13b156c7c76260dbc87e037a8fc290c3c86b614	☐ List ☑ Write	☑ Read ☑ Write

```
pip install rdbtools
from rdbtools import RdbParser, RdbCallback
class MyCallback(RdbCallback):
    def __init__(self):
        super(MyCallback, self).\
        __init__(string_escape=None)
    def set(self, key, value, expiry, info):
        print('%s = %s' % (self.encode_key(key), \
                           self.encode_value(value)))
parser = RdbParser(MyCallback())
parser.parse('backup.rdb')
```

Mounting and Data Preview

As the rdb downloaded and converted into csv, we could mount the S3 bucket and make some analysis as below

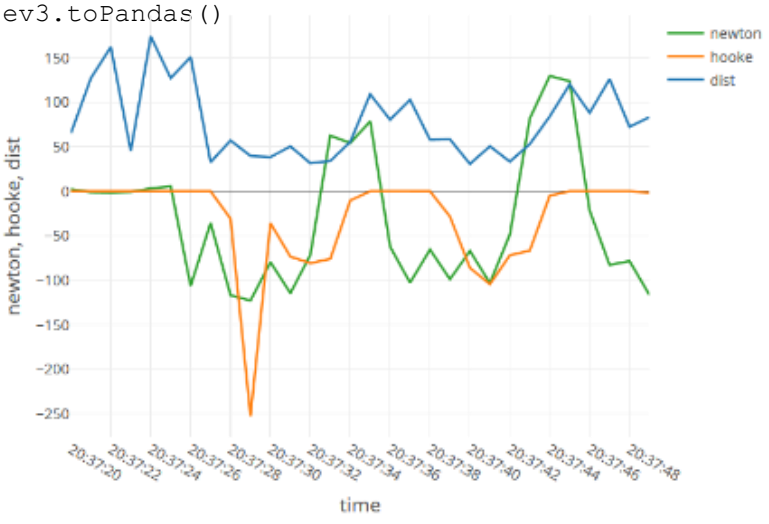
```
def mount_s3_bucket(access_key, secret_key, bucket_name,
mount_folder):
 ACCESS_KEY_ID = access_key
 SECRET_ACCESS_KEY = secret_key
 ENCODED_SECRET_KEY=SECRET_ACCESS_KEY.replace("/","%2F")

 print ("Mounting", bucket_name)

 try:
    dbutils.fs.unmount("/mnt/%s" % mount_folder)

 except:
    print ("Directory not unmounted: ", mount_folder)

  finally:
    dbutils.fs.mount("s3a://%s:%s@%s" % (ACCESS_KEY_ID,
      ENCODED_SECRET_KEY, bucket_name), "/mnt/%s" %
      mount_folder)
    print ("The bucket", bucket_name, "was mounted to",
      mount_folder, "\n")
```

```
# unmounted if mounted
```

```
# login credentials
ACCESS_KEY = "AKIAIIBVF6B2KTMMQGOQ"
SECRET_ACCESS_KEY =
"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"
```

```
# mount bucket
mount_s3_bucket(ACCESS_KEY, SECRET_ACCESS_KEY,
    "farius/EV3", "ev3")

fileName = "/mnt/EV3/ev3.csv"

ev3 = (spark.read.option("header", "true")
                 .option("inferSchema", "true")
                 .csv(/mnt/EV3/ev3.csv).cache())
```

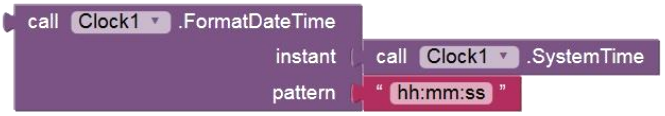
```
# materialize the caching
ev3.count()

from pyspark.sql.functions import col

display(ev3.filter(col('unix').substr(2,\
      14).cast("long")>1609027640000)\
      .filter(col('unix').substr(2,\
      14).cast("long")<1609027670000))

ev3.toPandas()
```



Feature Engineering

Adding few more features would be interesting where below data transformation is aimed to add four more features: rpm, reg, reg2 and quartile. The time captured from Lego always has six digits format as above.

```
from pyspark.sql.functions import monotonically_increasing_id
from pyspark.sql.functions import round, abs, col, when, lag, rank, lit
from pyspark.sql.window import Window

#Adding index column and 24-hr-time-fraction column for rpm calculation
ev3_1=ev3.select(col('*'))\
.withColumn("id", monotonically_increasing_id())\
.withColumn('timx', round((col('time').substr(1,\
 2)+col('time').substr(4, 2)/60+
   (col('time').substr(7, 2)+col('unix').substr(12,\
    14)/1000)/3600
  )/24, 8)).drop('time', 'unix', 'slide')

#Adding four differentials columns for feature engineering
ev3_2=ev3_1.select(col('*'))\
.withColumn("dn",(col('newton')-lag("newton",1).\
    over(Window.partitionBy("model").orderBy("id")))/\
     abs(lag("newton",1).over(Window.partitionBy("model")\
                              .orderBy("id"))))\
.withColumn("dd",col('dist')-lag("dist",1).over(Window\
            .partitionBy("model").orderBy("id", 'timx')))\
.withColumn("da", col('tach')-lag("tach",1).over(Window\
            .partitionBy("model").orderBy("id")))\
.withColumn("di", col('timx')-lag("timx",1)\
        .over(Window.partitionBy("model").orderBy("id")))\
.drop('timx', 'tach')

#Feautures added (reg, rpm, reg2)
ev3_3=ev3_2.select(col('*'))\
.withColumn('rpm', col('da')/360/(col('di')*24*60))\
.withColumn("reg", when(abs("hooke") > 0.1,100).otherwise(0))\
.withColumn("reg2", when(col('dd') ==
                    0,lag("reg",1).over(Window.partitionBy("model").orderBy("id")))\
                    .otherwise(when(col('dn')/col('dd')>-0.2,100).otherwise(0)))\
                    .drop('dn', 'dd', 'di', 'da')\
.withColumn("rk", rank().over(Window.partitionBy("model").orderBy('hooke'))).drop('model')\
.withColumn("cu", monotonically_increasing_id())

rl=ev3_3.select("rk").rdd.max()[0] # number of rankings, not quite the same as sample size

#Adding quartile feature
ev3_4=ev3_3.select(col('*'))\
.withColumn('quartile', lit((round((25+25/(1+2.718**(rl*1/4-col('rk')))
                                    +25/(1+2.718**(rl*2/4-col('rk')))
                                    +25/(1+2.718**(rl*3/4-col('rk')))
          )/25)*25).cast('integer'))).sort('id').drop('rk', 'cu', 'id')

prepDF=ev3_4.select('*').na.drop()
```

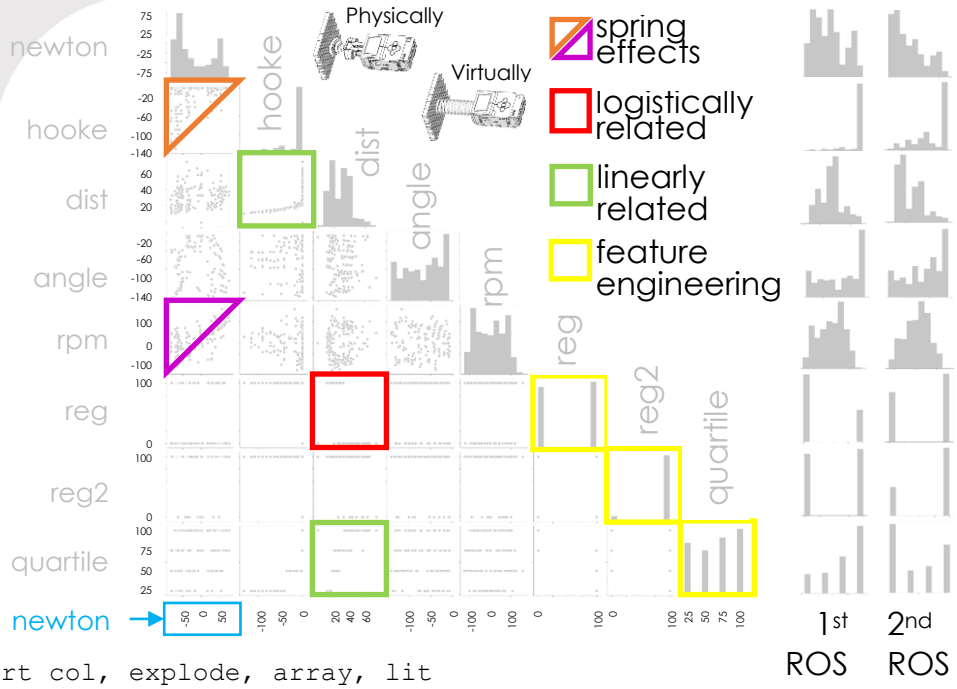


AUTONOMOUS DRIVING

Exploratory

From pairplot we found couple imbalance dataset and been oversampled accordingly. We go forward with combined_df for further modeling processes.

We could also see triangles with half box empty which indicates that the spring effects to avoid collision works.




```
from pyspark.sql.functions import col, explode, array, lit
import seaborn as sns

df = trainDF

major_df = df.filter(col("reg2") == 100)
minor_df = df.filter(col("reg2") == 0)
ratio = int(major_df.count()/minor_df.count())
print("ratio of majorty reg2 compared to minority reg before oversampling:
{}".format(ratio))

a = range(ratio)# duplicate the minority rows
oversampled_df = minor_df.withColumn("dummy", explode(array([lit(x) for x in
a]))).drop('dummy')# combine both oversampled minority rows and previous majority rows
combined_df = major_df.unionAll(oversampled_df)
major_df = combined_df.filter(col("reg2") == 100)
minor_df = combined_df.filter(col("reg2") == 0)
ratio = int(major_df.count()/minor_df.count())
print("ratio of majorty reg2 compared to minority reg aftere oversampling:
{}".format(ratio))

sns.pairplot(combined_df.toPandas())

df=combined_df
major_df = df.filter(col("hooke") >= -25)
minor_df = df.filter(col("hooke") < -25)
ratio = int(major_df.count()/minor_df.count())
print("ratio of majorty hooke compared to minority reg before oversampling:
{}".format(ratio))

a = range(ratio)# duplicate the minority rows
oversampled_df = minor_df.withColumn("dummy", explode(array([lit(x) for x in
a]))).drop('dummy')# combine both oversampled minority rows and previous majority rows
combined_df = major_df.unionAll(oversampled_df)
major_df = combined_df.filter(col("hooke") >= -25)
minor_df = combined_df.filter(col("hooke") < -25)
ratio = int(major_df.count()/minor_df.count())
print("ratio of majorty hooke compared to minority reg after oversampling:
{}".format(ratio))

sns.pairplot(combined_df.toPandas())
```

Data Pipeline

```
# StringIndexer
from pyspark.ml.feature import StringIndexer

regStringIndexer = StringIndexer(
                   inputCol="reg",
                   outputCol="reg_index")
(regStringIndexer
      .fit(combined_df)
      .transform(combined_df))

reg2StringIndexer = StringIndexer(
                    inputCol="reg2",
                    outputCol="reg2_index")
(reg2StringIndexer
      .fit(combined_df)
      .transform(combined_df))

quartileStringIndexer = StringIndexer(
                        inputCol="quartile",
                        outputCol="quartile_index")
(quartileStringIndexer
      .fit(combined_df)
      .transform(combined_df))

from pyspark.ml.feature import VectorAssembler

assemblerInputs  = [
  "newton", "dist", "angle", "rpm",
  "reg_index", "reg2_index", "quartile_index"]

# Vector Assembler & Scaler
from pyspark.ml.feature import StandardScaler
vectorAssembler = VectorAssembler(inputCols=assemblerInputs,\
 outputCol="unscaled_features")
standardScaler = StandardScaler(inputCol=\
 "unscaled_features", outputCol="features")

from pyspark.ml.regression import RandomForestRegressor

# Random Forest Regressor
rfr = (RandomForestRegressor().setLabelCol("hooke")
      .setSeed(27).setNumTrees(3).setMaxDepth(10))

from pyspark.ml import Pipeline

# Pipeline Definition
pipeline = Pipeline().setStages([
  regStringIndexer, reg2StringIndexer,
  quartileStringIndexer, vectorAssembler,
  standardScaler, rfr])

cat_columns = ['reg','reg2','quartile']

cat_indexers = [StringIndexer(inputCol=c,
outputCol=c+'_index') for c in cat_columns]

pipeline = Pipeline().setStages([
  *cat_indexers, vectorAssembler, standardScaler, rfr])

# Train the model
pipelineModel = pipeline.fit(combined_df)
```

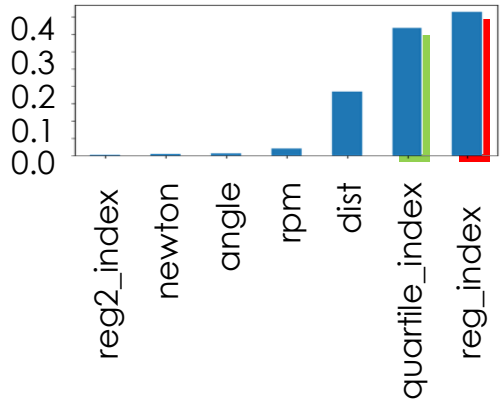
Feature Importance

We could take couple feature importance from the two models as below.

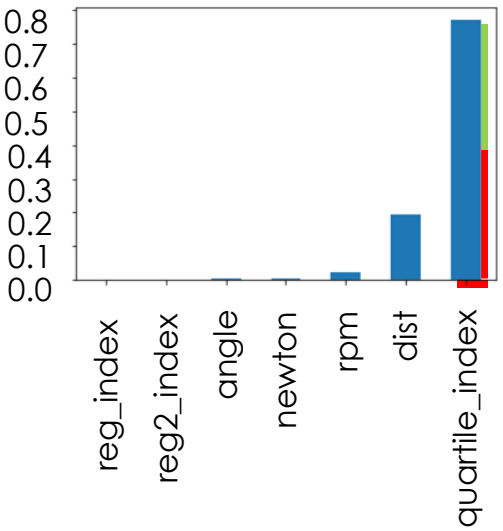
Couple bar charts were shadowed in red and green to illustrate where the feature might be substituted or went elsewhere. In this case I shadowed reg_index with red and quartile_index with green.

Comparing the two models before hyper parameter was tuned we saw that the reg feature became less significant On gradient boost model and quartile feature took all the credit. It explain the red box on page 1 where the logistic regression between reg and dist no longer related if Gradient boost model was selected.

Random Forest Model



Gradient Boost Model

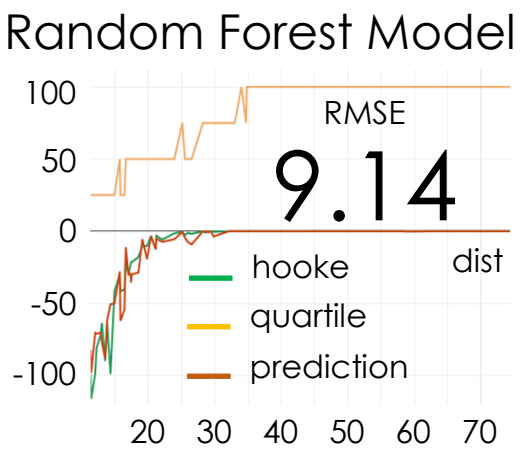


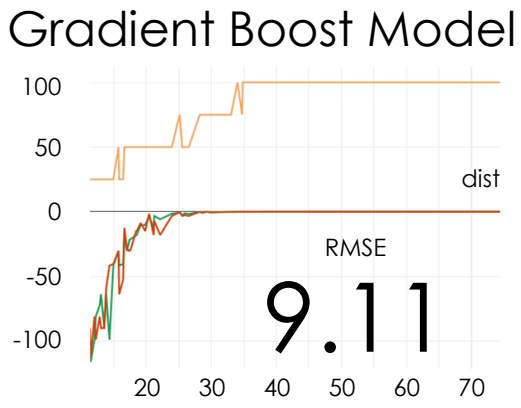
AUTONOMOUS DRIVING

Making Prediction

Ingesting test data into the model would be as follow. We will send the data to S3 for Quick Sight visualization and some basic visualization on databricks below.

```
# TestDF → pipeline
predictionsDF = pipelineModel.transform(testDF)

# Reorder the columns for easier interpretation
reorderedDF = predictionsDF.select("hooke",
"prediction", "newton", "dist", "angle", "rpm", "reg",
"reg_index", "reg2", "reg2_index", "quartile",
"quartile_index")

# Display on databricks
display(reorderedDF)
```






```
# to display on Quick Sight
reorderedDF.coalesce(1).write.csv("dbfs:/mnt/ev3//result
",mode='overwrite',header=True)
```

Parameter Tuning

Below codes tried to fine tuning the results using Spark's ParamGridBuilder and using 3-fold cross-validation to identify optimal maxDepth and numTrees combination

```
# Hyperparameter Tune
```

```
from pyspark.ml.tuning import ParamGridBuilder

paramGrid = (ParamGridBuilder()
             .addGrid(rfr.maxDepth, [10, 20])
             .addGrid(rfr.numTrees, [3, 5])
             .addGrid(rfr.seed, [2018, 42])
             .build())
```

```
# Cross-Validation
```

```
from pyspark.ml.tuning import CrossValidator
from pyspark.ml.evaluation import RegressionEvaluator

evaluator = (RegressionEvaluator()
  .setLabelCol("hooke")
  .setPredictionCol("prediction")
             )

cv = (CrossValidator()
          .setEstimator(pipeline)
          .setEvaluator(evaluator)
          .setEstimatorParamMaps(paramGrid)
          .setNumFolds(3)
          .setSeed(2018))

cvModel = cv.fit(combined_df)
```

```
# Tabulate the results
```

```
para=[]
for i in range(len(cvModel.getEstimatorParamMaps())):
  sub_para=[]
  for j in range(len(list(\
      cvModel.getEstimatorParamMaps()[i].keys()))):

      sub_para+=[str(list(cvModel.getEstimatorParamMaps(
      )[i].values())[j])]

  para+=[sub_para+[cvModel.avgMetrics[i]]]

colname=[str(list(cvModel.getEstimatorParamMaps()[0]\
      .keys())[k]).split('__')[1]
  for k in range(len(list(\
      cvModel.getEstimatorParamMaps()[0].keys())))]

colname.append('rmse')

pd.DataFrame(para, columns=colname).sort_values('rmse')
```

	maxDepth	numTrees	seed	rmse
7	20	5	42	8.276923
3	10	5	42	8.277526
2	10	5	2018	9.274347
6	20	5	2018	9.294609
5	20	3	42	10.127391
1	10	3	42	10.127416
4	20	3	2018	10.132922

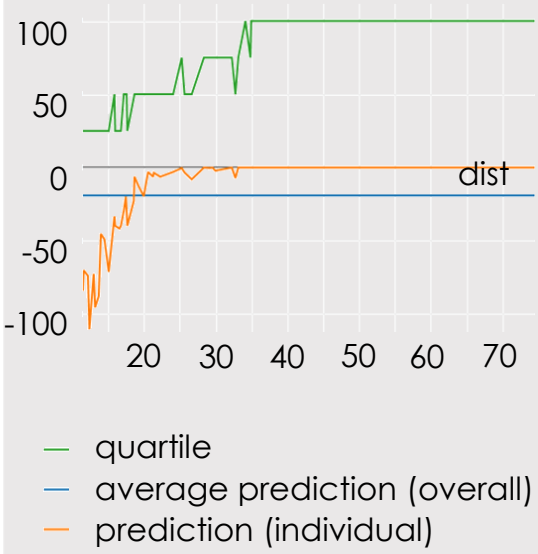
```
# test data → last model
```

```
finalPredictionsDF = cvModel.bestModel.transform(testDF)
```

AUTONOMOUS DRIVING

Lift Curve

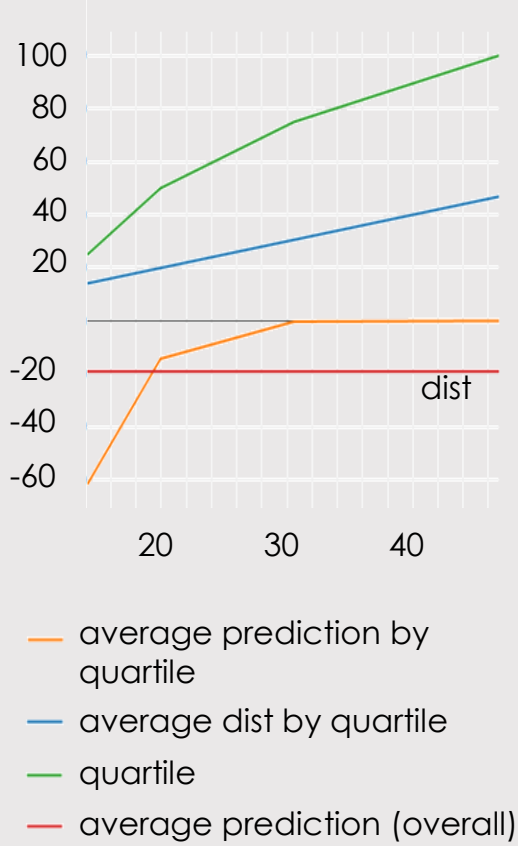
Lift Chart by individual data



As more commonly used in telecom industry or similar to track the churn rate from their customer seeking for best strategy to revisit, the same lift analysis could be used in this exercise. Instead of dividing the prediction into ten (decile) we will use only four partition (quartile).

The first chart on the left show individual prediction and compared to its average prediction line. The quartile green line was inherited from the beginning of the analysis before the data was split into train and test dataframe which was ranked and paretoed from the highest hooke into the lowest and divide them by four quadrant. Ideally we should redo the same exercise focusing on the test dataframe only.
The average hooke force in this case was -20 where the forces normally went to zero after 20 cm.

Lift Chart by Quartile



The better way to utilize lift analysis would be to group those measurement into sub-classes where in this case the first quartile (25) has the highest hooke and the last quartile (100) has the lowest.

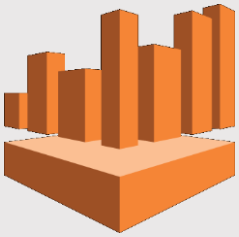
In this case we would be interested to see the average of the first quartile as it was way below the average line. The second quartile was slight above the average and the last two quartiles were normally zeros (no reaction forces).

The measurement was taken mainly near obstacle and entitled to get reaction force interference maybe half of the time. The red average prediction line was therefore pretty high (-20). If the driving was mainly on quite track with no other in front most of the time the average would come to near zero. Likewise if the driving was mainly on traffic jam the average would be higher.

AUTONOMOUS DRIVING

AWS Athena

Probably the advantage of involving Athena between S3 and QuickSight is flexibility of pinpointing the S3 bucket rather than specific file in the bucket. As PySpark save a file back to S3 through the same mount the file name is always randomized.



1. **Data source** — Connect data source
2. AwsDataCatalog ▼
3. AWS Glue data catalog — Next
4. Add table and enter schema information manually — Continue to add table

Databases > Add Table > Step 1: Name & Location

INPUT

Database: Create a new database ▼ — new

Table Name: new — Next

Location of Input Data Set: s3://farius/EV3/result/

Location of prediction csv file

5. Databases > Add Table > Step 2: Data Format — CSV — Next

6. Databases > Add Table > Step 3: Columns

Bulk add columns

```
hooke float, prediction float, newton float,
dist float, angle int, rpm float, reg int,
reg_index int, reg2 int, reg2_index int,
quartile int, quartile_index int
```

Add

OUTPUT


```
CREATE EXTERNAL TABLE IF NOT EXIST new.new (
'hooke' float, 'prediction' float, 'newton' float, 'dist' float,
'angle' int, 'rpm' float, 'reg' int, 'reg_index' int, 'reg2' int,
'reg2_index' int, 'quartile' int, 'quartile_index' int)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.
lazy.LazySimpleSerde'
WITH SERDEPROPERTIES ('serialization.format' = ',',
                      'field.delim' = ',')
LOCATION 's3://farius/EV3/result/'
TBLPROPERTIES ('has_encrypted_data'e'false');
```

AUTONOMOUS DRIVING

Quick Sight

Five steps to visualize Athena table to Quick Sight

1

2

3

4

5

QuickSight

New dataset

New Athena data source

Data source name

create data source

newone

Choose your table

newone

Catalog: contain sets of databases

AwsDataCatalog

Database: contain sets of tables

new

Tables: contain the data you can visualize

new

select

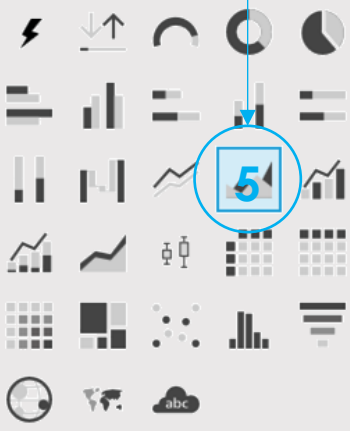
Finish dataset creation

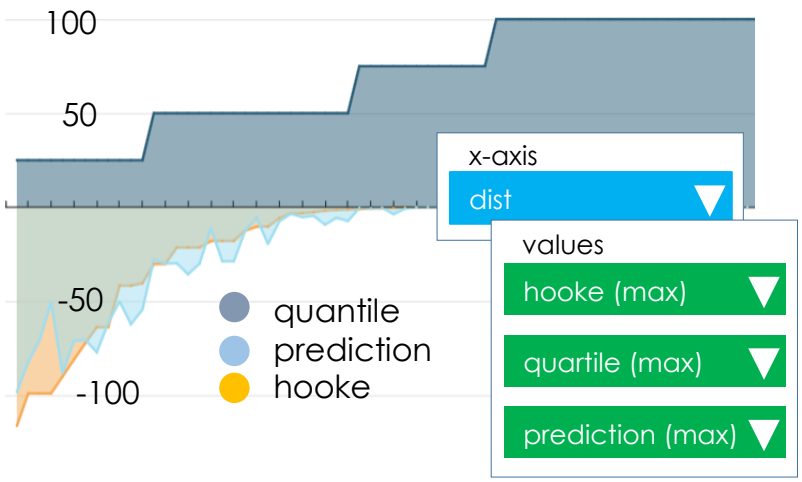
Table: new
Data source: newone
Schema: new

directly query your data

visualize

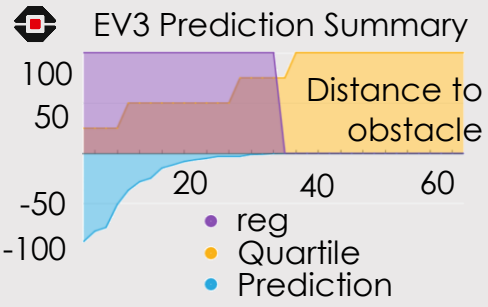
Visual Type

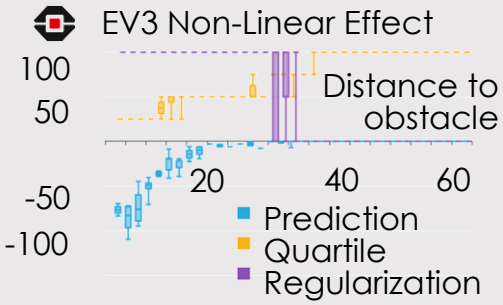


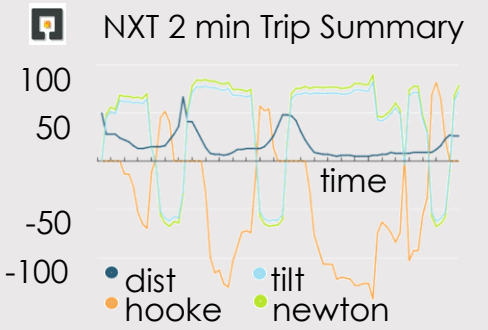


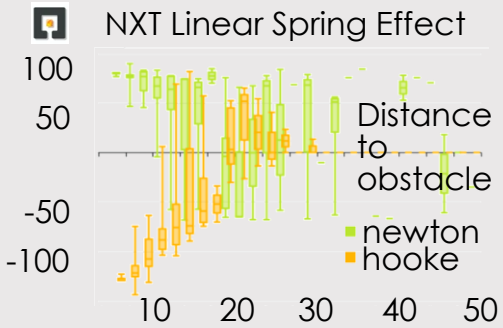
AUTONOMOUS DRIVING

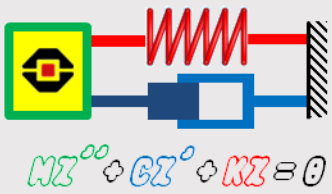
Conclusion











Visualizations below from Quick Sight may explain few situations of this simple autonomous driving using EV3 and NXT where specifically the algorithm of virtual spring reactions were made different. EV3 with sigmoid spring, NXT with a linear one.

This chart simply took an aggregate values such as maximum from reg, quartile and prediction. For reg feature it shows a prolonged classification of hooke reaction which normally occur when the distance to obstacle less than 20 cm. In this case some maximum values affect the visualization summary. Better visualization might be gained from detail box plot below for all cases rather than aggregated as max, min or average.

In detail box plot it show most significant hooke reactions grew from distance to obstacle less than 20 cm. Those minor values scattering after 20 cm might be noises from sigmoid curves inefficiently round down the reaction forces to complete zeros. It explains why the purple summary line above has prolonged classification to 30-40 cm instead. One way to improve the model would be occupying steeper sigmoid curve such as replacing the exponential number e with 10 or 20 for example.

Trip summary here shows a relationship between the distance to the obstacle to the hooke forces which normally went the opposite way. As the spring algorithm for NXT was made linear the spring also pulling the lego car toward the wall on top of pushing back when gets closer. In real world it would be the cruise control mode where certain speed or distance were set. In case of EV3 those orange line would never go above x-axis as the forces were zeroed by sigmoid function.

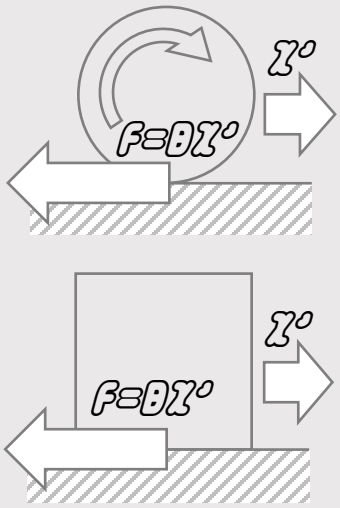
This last chart is similar to the box-plot above for EV3 but it show the springy effect continues above x-axis as a linear algorithm was applied. The counter forces were discontinued when the distances from the obstacle was way too far.

In summary reaction forces from virtual spring enlarged as distance to the obstacle narrowed. There are many way to improve the algorithm. Self-driving technology spent machine learning efforts continuously in this area (reprogrammable).

As part of this project I also saw an opportunity to bring an analogy of other aspects that could be convey to young generation. Area such as project management and leadership values may apply while understanding, building, programming this autonomous vehicle (see last page of the portfolio).

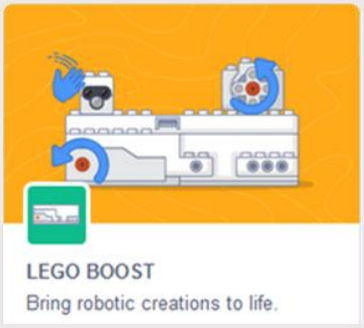
AUTONOMOUS DRIVING

Next Steps



Applying deep learning to find better features according to driving behavior with more sensors is an interesting next step. Also getting streaming data from redis or firebase is another.

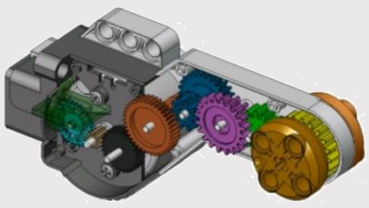
But more important is considering friction between wheel and the track. On the ideal equation mx''+bx'+cx=0 we focus more on m and c with less emphasis on b. Herewith are two scenarios on the left : rolling and sliding. The first one where the wheel really rolls and provide controlled displacement. The second one is uncontrolled displacement where the wheel is sliding due to slippery track. To encounter this situation at least one non-energized tachometer from a motor will record the rpm. In EV3 tachometer comes standard inside the motor, not for NXT.



Up to now we talked more on Redis and App Inventor ecosystems for Lego robotics. More affordable kits for young kids also sold few years ago without interaction to mobile phone but more likely between desktop/tablet and the robot. The cloud software also made by MIT called Scratch which also available for EV3 (not NXT) plus another two sets (WeDo, Boost) I am familiar with Scratch block programming as I made few line of codes according to math pattern for grade 4 students.

example of non-robotic code
https://scratch.mit.edu/projects/250959275

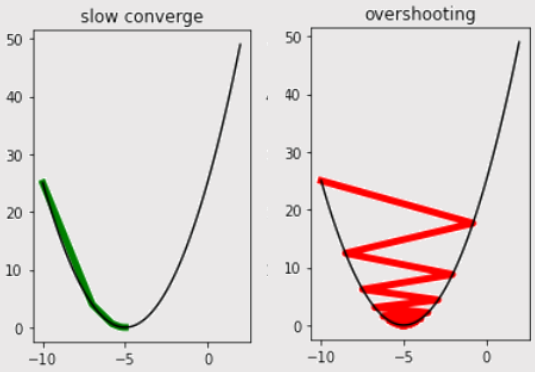
This portfolio is normally not trying to go further into robotics, vision systems or DevOps as a whole such as bringing more robotic sensors such as EV3 gyro sensor, tach rotation sensor attached to large motor to conduct better data exploratory.



Instead it would be a nice brief analogy to apply the idea to everyone daily life such as personal finance, especially credit card expenditure. After a mortgage monthly payment the second largest bill could be the credit card monthly spending.

AUTONOMOUS DRIVING

Challenges



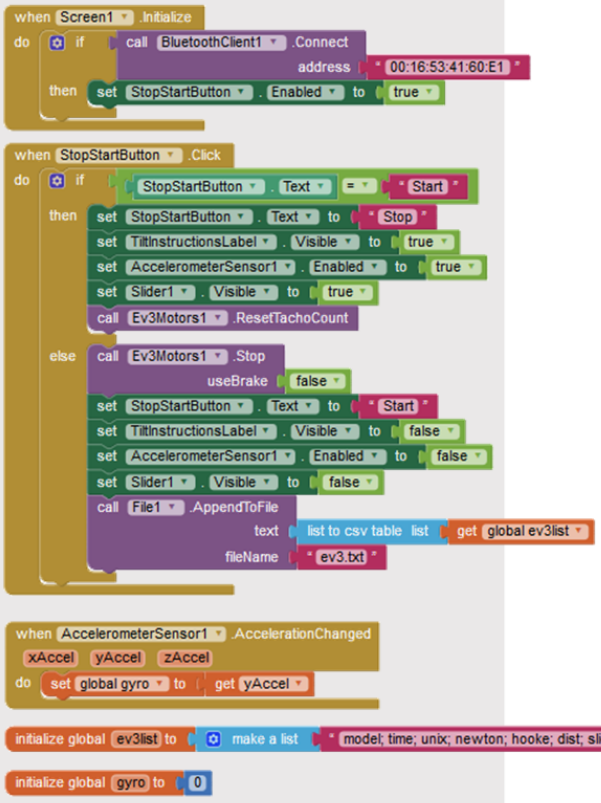
https://colab.research.google.com/github/FariusGitHub/DataScience/blob/master/machineLearning_LearningRate.ipynb

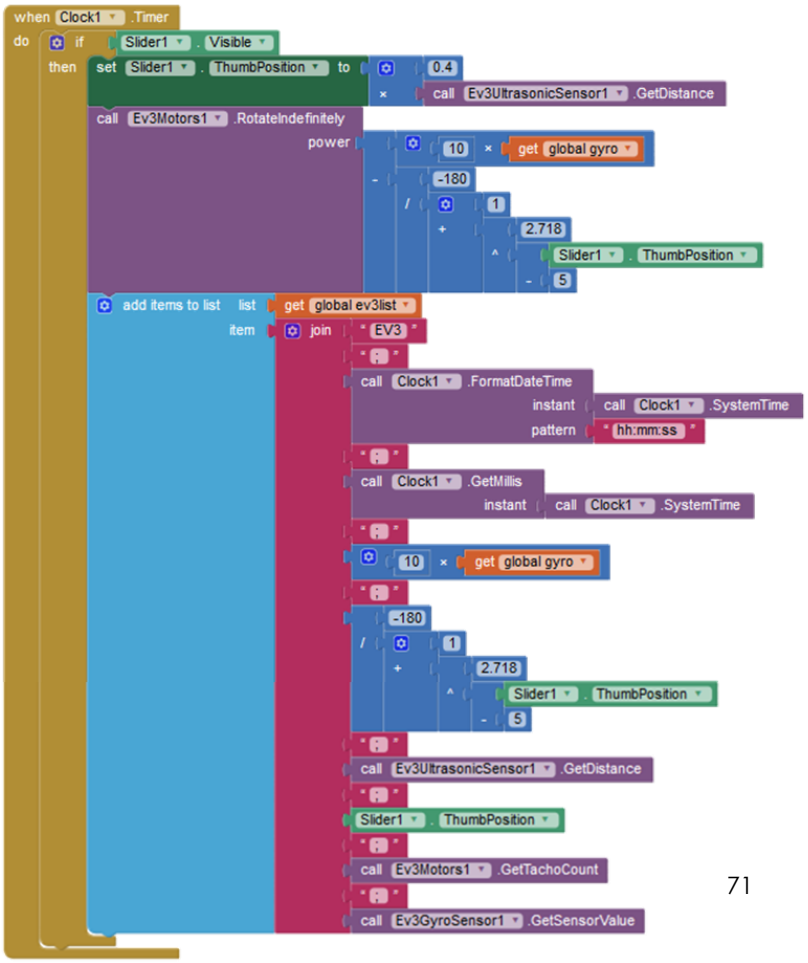
Unlike using jupyter notebook, databricks might take longer time to run few prediction especially paramgridbuilder. Simpler regressor such as decision tree or random random may have a faster time to complete while gradient descent or SVM could take longer time especially on the community accounts. Tweaking paramgridbuilder with step size (in scikit learn known as learning rate) could be slow, possibly overshooted.

Using redis database also brought its own challenges. It was one of the official cloud storage picked by MIT App Inventor beside Firebase and TinyWebDB. The rdb NoSQL database was normally stored in Redis cloud memory and could be backed into rdb file with paid subscription. This rdb file is a compressed binary file of personal database. It is impossible to convert it directly to csv file as first it is needed to load it to Redis and then read it and convert it through redis-cli through EC2 platform.

Another temporary solution to collect the data would be to download the sensor data locally on android phone and upload it manually to S3 as CSV format as below.

http://ai2.appinventor.mit.edu/#6370871918067712



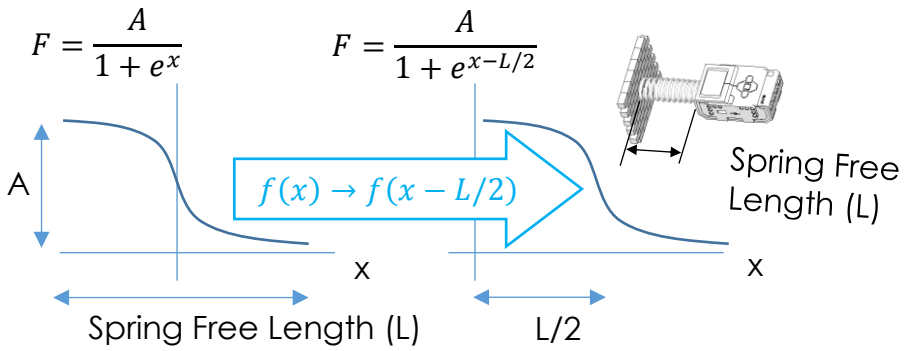


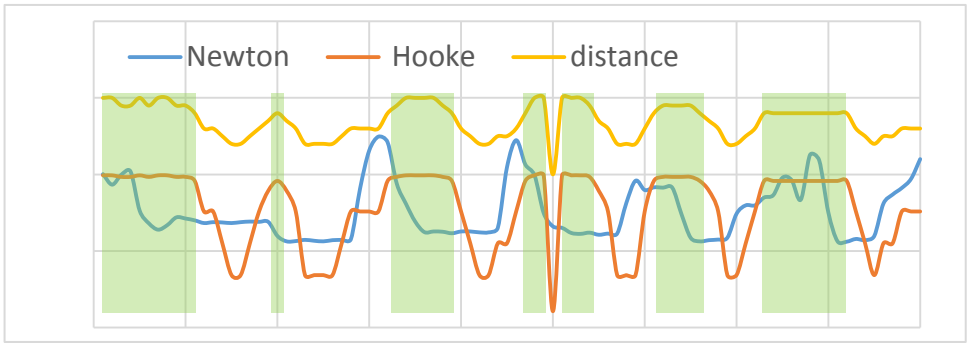
Spring Forces

(APPENDIX 1)

Based on overall idea described on page 56 herewith is a concept of ultrasonic sensor, the initial force passed to Lego (triggered by tilting the phone) and the spring force occurred where distance is equal/less than 20 cm (L).

Spring force uses sigmoid function, shifted by L/2 where L is the spring free length (no compression force applied).





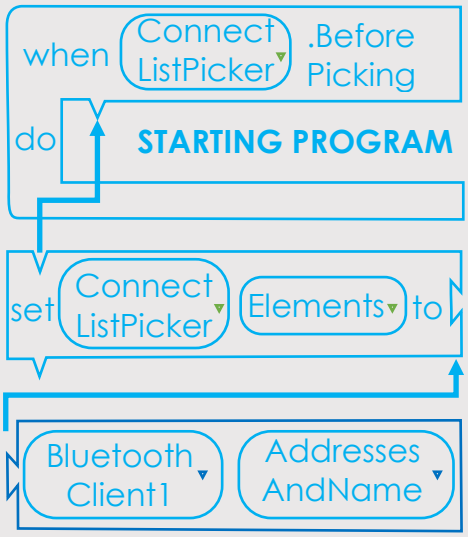
For fast response spring reaction force is multiplied higher says for this case given a factor of 1.8 from initial force.

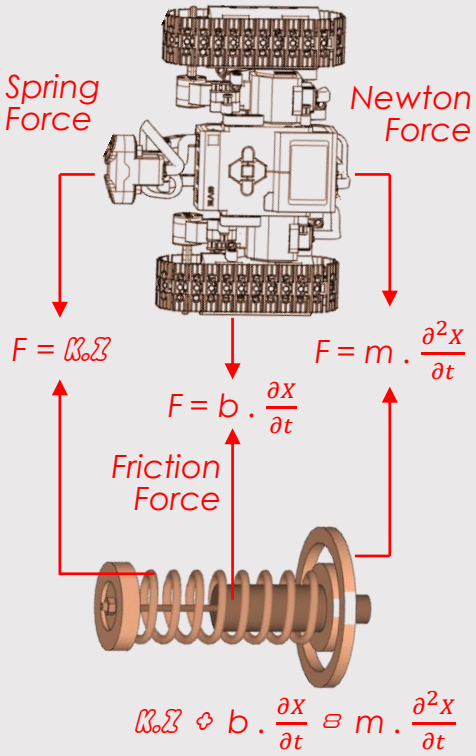
Above chart show typical curves over time (two minutes). When the spring force occurs (red line went below zero) it means the distance to obstacle is getting too close and Lego encounters with virtual spring force to avoid collision. Meaning to say those areas shaded in green represent safe distance and no intervention needed from Lego.

AUTONOMOUS DRIVING

Beyond Physic Project Scope Programming

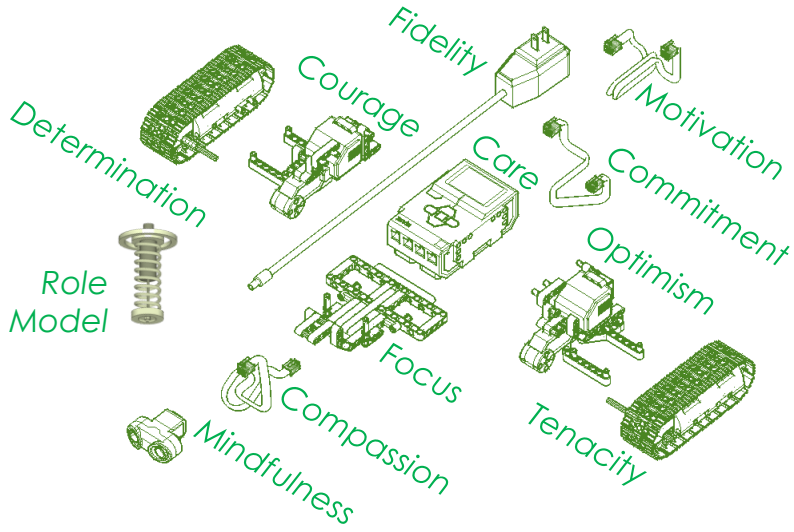
(APPENDIX 2)





Beside all technicality surrounds this project there are another take away such as **project management** and **leadership** that could be introduced to youth. I published this project as an amazon app at https://www.amazon.ca/dp/B01LM3G9FS/

For the first dozen leadership traits I relate them to the **Project Scope** that I applied for this portfolio. Use your imagination to understand each part that explicitly explains a quality of a trait. For example a charger represent a fidelity to tirelessly provide power whenever needed, ultrasonic sensor for mindfulness, etc.



Another part of leadership traits could be covered in *physics* and *programming* of this Lego. There are more than one dozen more leadership traits we could relate to. Use your imagination!

The **Physics** of the robot		**Programming** of the robot	
Math Equations	• Patience • Confidence	Lean EV3 UX Design	• Creativity • Innovation
Damping, Detection	• Perseverance • Humility	Bluetooth Phone Gyro	• Resilience • Passion
Hooke's Law spring force	• Honesty	EV3 Motors EV3 Sensors	• Authentic • Encouragement
Newton 2nd	• Integrity	Start Program	• Openminded
Friction Force	• Respect	AI Program	• Inspiration
Control loop	• Kindness	Drive program	• Moral

PAYMENTS CANADA

PAYMENTS CANADA

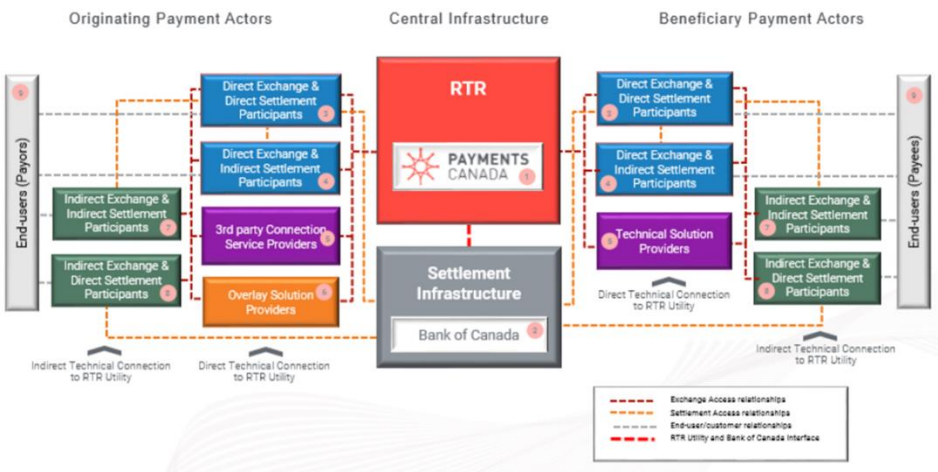
As part of Payments Canada's modernization plan, financial institutions are laying the foundation to support a new payments system known as the real-time rail (RTR).

The RTR is designed to facilitate the instant and irrevocable exchange and clearing of payment items. The instant and irrevocable nature of the RTR places a large importance on the requirement for a robust system of detecting fraudulent transactions in real-time.

A large bank wants to ensure their internal fraud detection systems are capable of meeting the needs of the RTR when it goes live in 2019-2020. In particular, the ever-changing tactics of fraudsters necessitates the ability to quickly adapt to new attack vectors.

While the banks can purchase off-the-shelf fraud solutions from third-party vendors, many of these products do not fully integrate with the wealth of information the bank has at their disposal across products and channels.

In this portfolio I would like to share my project where I was assigned to develop a flexible fraud solution that quickly learn to adapt to new attack vectors.



PAYMENTS CANADA

SAMPLE DATA

The sample data consist of 8000 labeled cases enriched by a hundred features was given. There are six columns needed to digitize to ease most models to digest numerical values. Some oversample also made and the length of rows (8000) consistently monitored during the transformation.

```
def f(x):
 return
  pd.concat([#TRANSFORMING x5 feature
             pd.DataFrame(((x['x5'].fillna('0.0%').str[:-].astype(float))\
             *100).astype(int)),

             #TRANSFORMING x6 feature
             pd.DataFrame(x['x6'].map({'Monday': 1,'Tuesday': 2,'Wednesday':\
             3,'Thurday': 4,'Friday': 5,np.nan: 0})),

             #TRANSFORMING x20 feature
             pd.DataFrame(x['x20'].map({'Jan': 1, 'Feb': 2,'Mar': 3,'Apr': \
             4,'May': 5, np.nan: 0,'Jun': 6,'Jul': 7,'Aug': 8,'Sept': 9,'Oct':\
             10,'Nov': 11,'Dec': 12})),

             #TRANSFORMING x27 feature
             pd.DataFrame(x['x27'].map({'Morning':1, 'Afternoon': 2,\
             'Evening':3, 'Night':4, np.nan:0})),

             #TRANSFORMING x49 feature
             pd.DataFrame(x['x49'].map({True:1, False: -1,np.nan: 0})),

             #TRANSFORMING x57 feature
             pd.DataFrame(x['x57'].fillna('$0.0'). str.strip('$').astype(float))

             #COMBINING WITH 94 other features
             x.drop(['x5','x6','x20','x27', 'x49','x57'], axis=1).fillna(0)

            ], axis=1)
```

Oversampling dashboard

```
length of X_train before ROS 6400
length of y_train before ROS 6400
0    6040
1     360
Name: y, dtype: int64
length of X_train after ROS 12080
length of y_train after ROS 12080
1    6040
0    6040
Name: y, dtype: int64
```

Newly Transformed Data Mapping

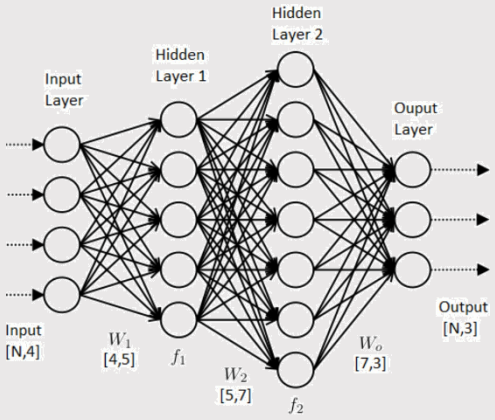
	-3687.1	-4.0	-3.0	-2.0	-1.0	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0	12.0	3647.74	total
x20		0	0	0	0	0	100	9	47	210	635	1289	1912	1823	1229	545	165	26	10	0	8000
x27		0	0	0	0	0	80	200	3553	3895	272	0	0	0	0	0	0	0	0	0	8000
x49		0	0	0	0	4330	92	3578	0	0	0	0	0	0	0	0	0	0	0	0	8000
x5		0	1	60	462	1881	3139	1929	479	48	1	0	0	0	0	0	0	0	0	0	8000
x57		1	0	0	0	0	85	0	0	0	0	1	0	0	0	0	0	0	1	0	8000
x6		0	0	0	0	0	91	199	2347	4135	1179	49	0	0	0	0	0	0	0	0	8000

Newly Transformed DataFrame

	x5	x6	x20	x27	x49	x57	x0	x1	x2	x3	...	x91	x92	x93	
0	3	2	8	2	-1	1578.34	-0.247824	6.425853	-6.485658	-8.241462	...	0.592543	14.967527	6.691044	-0.27
1	2	3	7	2	-1	767.76	2.623274	-2.808263	5.321746	-2.437653	...	-1.712788	-3.638637	-2.256492	2.70
2	0	2	8	3	-1	931.06	-5.460551	5.997701	-4.614194	-6.035128	...	2.206956	7.292558	10.095556	1.60
3	0	4	8	2	1	2124.52	1.566307	-0.080517	0.064754	6.063193	...	-3.841669	5.189840	10.291069	-3.64
4	0	3	7	3	-1	-3055.67	0.750715	15.656095	-6.778600	11.251037	...	8.933781	-7.295794	-11.246059	-4.87
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
7995	1	2	6	2	-1	-1664.04	-2.664967	-3.427965	-5.143886	0.289251	...	2.123462	6.139848	-4.323045	-6.45
7996	0	2	5	2	-1	-42.81	2.174302	-4.912113	-4.803975	-22.081911	...	-12.307131	-4.762671	10.574798	-3.95
7997	-1	3	8	2	1	220.59	2.247079	-3.377818	-1.567924	-7.445187	...	-8.084817	8.031024	7.282587	2.43
7998	0	2	8	3	1	467.47	-1.402385	-20.117766	3.705099	11.634801	...	4.920424	-0.954106	6.261179	0.62
7999	-2	2	6	2	-1	-598.35	-5.761582	8.638806	8.391700	-6.474205	...	-4.721215	5.300117	-13.373868	1.45

PAYMENTS CANADA

FRAUD MODEL



	0	1	pred	true
0	0.953507	0.046493	0	0
1	0.954878	0.045122	0	0
2	0.956652	0.043348	0	0
3	0.978931	0.021069	0	0
4	0.956652	0.043348	0	0
5	0.980535	0.019465	0	0
6	0.983018	0.016982	0	0
7	0.875673	0.124327	0	0
8	0.026100	0.973900	1	1
9	0.950784	0.049216	0	0
10	0.974157	0.025843	0	0

Many popular models including Random Forest and Keras will be discussed quickly as the objective of this portfolio is more than just fine tuning the model.

```
model = Sequential()
model.add(Dense(40, activation='relu'))
model.add(Dense(30, activation="tanh"))
model.add(Dense(50, activation="tanh"))
model.add(Dropout(rate=0.5))
model.add(Dense(10, activation='sigmoid'))
model.add(Dense(2, activation="softmax"))

model.compile(optimizer='adam', metrics= ['accuracy'],
loss = 'sparse_categorical_crossentropy')
```

Above sequential is just an example of one optimization effort which at least taking four hidden layers between that hundred 100 features input and 2 feature outputs (fraud, or not fraud).

Summarizing the model prediction we find both loss and validation loss went down and accuracy and validation accuracy went up that showed a good trend of hyper parameter tuning. The accuracy would be between 80-90%.

Looking at the summary dataframe made from below code, the prediction will tend to be on the majority percentage to win the vote.

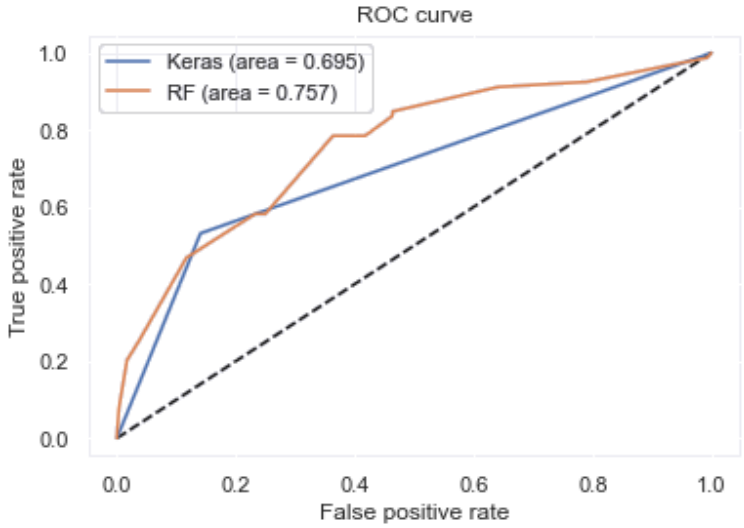
```
y_pred = model.predict(X_test)
summary=pd.DataFrame(y_pred)
summary['pred']=np.where(summary[0]>summary[1],0,1)
summary['true']=y_testing.values
```

There 1600 rows from testing data where true frauds were 79 and predicted fraud were 256 based on 20/80 test train split where the train data were later oversampled (grew from 6040 to 12080). Prior to sequential modeling the validation took the first thousand and left 11K for the model to learn.

PAYMENTS CANADA

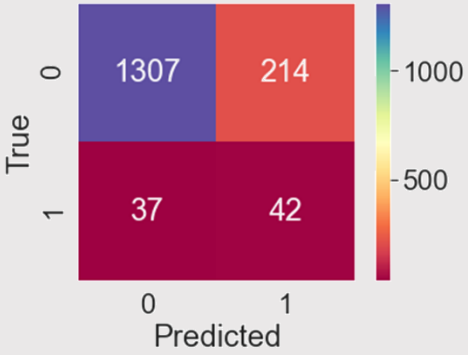
ROC and CONFUSION MATRIX

Based on the customer interest on ROC and Area under curve I took two models to compare. The second model was a random forest with 2 estimators and 2 max depth where Random Forest shows bigger AUC than Keras.



Another popular metric commonly used are confusion matrix and Classification report both taken from Scikit Learn libraries.

With Keras model it predicted correctly 1307 non-fraud cases and 42 fraud cases. 214 cases were predicted fraud where in fact they were not (false positive). 37 cases were predicted non-fraud but actually they were (false negative).



	precision	recall	f1-score	support
0	0.97	0.86	0.91	1521
1	0.16	0.53	0.25	79

METRIC AND LOSS FUNCTION

There would be no absolute right or wrong answer when picking a metric or a loss function as a benchmark to run a model. While in theory the loss function is used to optimize your model. This is the function that will get minimized by the optimizer where a metric is used to judge the performance of your model. It could give few clue to help with optimization process but most ineffective. I think as an experienced data science all metrics, functions help to improve model score.

One of the popular loss function widely brought through cost function and budgetary limit (s) is a lasso and ridge RSS curve originally introduced by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani from their book An Introduction to Statistical Learning.

They mentioned and explained how the size of budget reflect the ability of be risk-averse taking calculated risk to achieve few objective, which could be in real world related to marketing or research sector to new product or market.

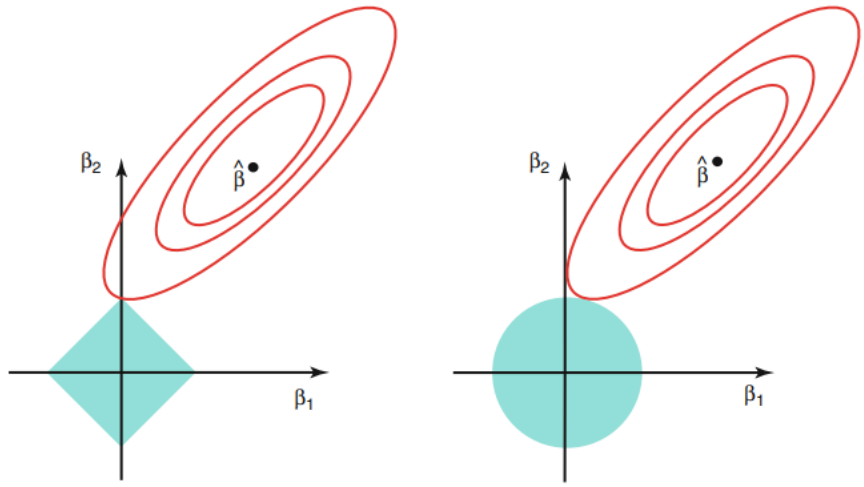


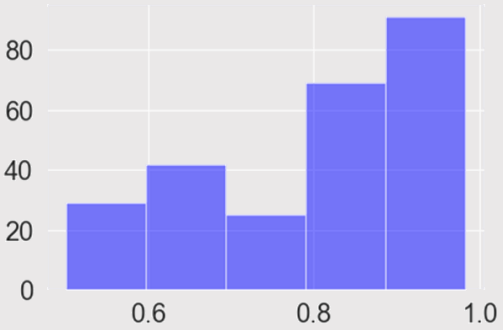
FIGURE 6.7. *Contours of the error and constraint functions for the lasso* (left) *and ridge regression* (right). *The solid blue areas are the constraint regions,* $|\beta_1| + |\beta_2| \leq s$ *and* $\beta_1^2 + \beta_2^2 \leq s$, *while the red ellipses are the contours of the RSS.*

When we perform the lasso we are trying to find the set of coefficient estimates that lead to the smallest RSS, subject to the constraint that there is a *budget* s for how large $\sum_{j=1}^{p} |\beta_j|$ can be. When s is extremely large, then this budget is not very restrictive, and so the coefficient estimates can be large. In fact, if s is large enough that the least squares solution falls within the budget

PAYMENTS CANADA

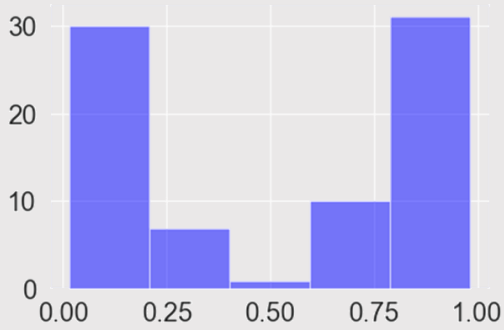
THRESHOLD

RECOMMENDATION



Keras Distribution of Fraud Probability for cases where Frauds were predicted

summary[1].loc[(summary ['pred']==1)]



Keras Distribution of Fraud Probability for cases where Frauds actually occured

summary[1].loc[(summary ['true']==1)]

Probably related to previous concern on metric to use of loss function we could also see the decision on classify as fraud from below segregation. As a whole there are 256 cases predicted as fraud but only 79 cases were fraud.

```
for i in [0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8]:
    print('Considered fraud as model'
          'probability <',i*100,'%',\
  summary['pred'].loc[(summary[1]>i)]\
          .value_counts().sum())
```

```
Considered fraud as modelprobability < 20.0 % 321
Considered fraud as modelprobability < 30.0 % 290
Considered fraud as modelprobability < 40.0 % 271
Considered fraud as modelprobability < 50.0 % 256
Considered fraud as modelprobability < 60.0 % 226
Considered fraud as modelprobability < 70.0 % 185
Considered fraud as modelprobability < 80.0 % 157
```

Running above code without looking at true and predicted values we know that the lower the threshold the more cases considered fraud and vice versa. Perhaps it is not a wise way to find the threshold.

The other way might be looking at the histogram of each the true or predicted chart on the left. The true chart showing more cases on the higher probability of fraud where the prediction chart show an opposite of bell curve. Probably the data itself was far away from perfect or the model could be further improved to get a closer normal distribution alike thus ease to propose the fraud threshold.

Most prediction model has a luxury for probability study. Random Forest use predict_proba and Keras has it built-in with its output neural network nodes as percentage of probability. Keras optimization would take time and could go even more complex than random forest for example.

I would suggest to keep the 50% threshold along with Keras recommendation as a whole for now. As the model learns there would be more information to consider. Perhaps also more information from the feature dictionary where feature engineering could be done broadly.

PAYMENTS CANADA

PIPELINE AND JOBLIB

Wrapping this project the objective is also to share the model in joblib format where I tested this feature quickly in the beginning since data cleaning process. Herewith is the simple code saving and loading back the joblib file to and from the current working directory.

```
#TESTING joblib file
pipe1 = Pipeline([('testing', FunctionTransformer(f))])
joblib.dump(pipe1, "pipe.joblib")
pipe2 = joblib.load("pipe.joblib")
data=pipe2.fit_transform(data1)
```

In term of bigger picture joblib vision is to provide tools to easily achieve better performance and reproducibility when working with long running jobs.

- Avoid computing the same thing twice: code is often rerun again and again, for instance when prototyping computational-heavy jobs (as in scientific development), but hand-crafted solutions to alleviate this issue are error-prone and often lead to unreproducible results.

- Persist to disk transparently: efficiently persisting arbitrary objects containing large data is hard. Using joblib's caching mechanism avoids hand-written persistence and implicitly links the file on disk to the execution context of the original Python object. As a result, joblib's persistence is good for resuming an application status or computational job, eg after a crash.

```
from joblib import Memory
cachedir = '...cache location'
mem = Memory(cachedir)
import numpy as np
a = np.vander(np.arange(3)).\
astype(np.float)
square = mem.cache(np.square)
square(a)
```

```
[Memory] Calling square...
square(array([[0., 0., 1.],
       [1., 1., 1.],
       [4., 2., 1.]]))
```

```
array([[ 0.,  0.,  1.],
       [ 1.,  1.,  1.],
       [16.,  4.,  1.]])
```

Main features

Transparent and fast disk-caching of output value: a memoize or make-like functionality for Python functions that works well for arbitrary Python objects, including very large numpy arrays. Separate persistence and flow-execution logic from domain logic or algorithmic code by writing the operations as a set of steps with well-defined inputs and outputs: Python functions. Joblib can save their computation to disk and rerun it only if necessary:

```
from joblib import \
    Parallel,\
    delayed
from math import sqrt
Parallel(n_jobs=1)\
(delayed(sqrt)(i**2)\
 for i in range(5))
```

```
[0.0, 1.0, 2.0, 3.0, 4.0]
```

Embarrassingly parallel helper: to make it easy to write readable parallel code and debug it quickly:

Fast compressed Persistence: a replacement for pickle to work efficiently on Python objects containing large data (joblib.dump & joblib.load).

https://joblib.readthedocs.io/en/latest/

CUSTOMER LOYALTY & OPERATION AGILITY

My last portfolio would be covering a telecom company with a dataset within two-year time covering net and gross revenues with each transaction of data cost and other cost associated from each call.

The main task is to document weekly and year-to-year net and gross revenue to ease the comparison by timeline prior to set the strategy ahead. I was also asked to give some opinions on the business by looking at the two years trend.

As I have years of experience in sales and supply chain I would like to take this opportunity from CRM and SCM point of view in term of customer loyalty and operational agility.

Customer Relationship Management
There are few definition about CRM, based on Wikipedia it was defined as the process of managing interactions with existing as well as past and potential customers. It is one of many different approaches that allow a company to manage and analyze its own interactions with its past, current and potential customers. It uses data analysis about customers' history with a company to improve business relationships with customers, specifically focusing on customer retention and ultimately driving sales growth.

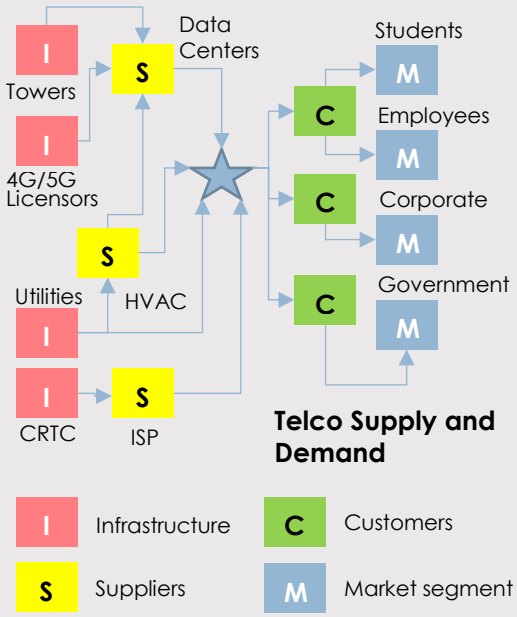
From data science point of view CRM application is perfect but not necessarily needs commercial tools such as Salesforce.

Due to limited and encrypted dataset we could consider only a few things such as the churn rate based on tenancy of certain customer overtime (customer loyalty).

Supply Chain Management
On the supply side which drive the cost and pricing could be illustrated on the left. Unlike an assembly industry (such as automotive) in Telco the supply chain network is way shorter. The star symbol represents the telco company in the middle of the supply chain where the cash flows from the right to the left.

In process improvement initiatives we may hear methodologies like CRISP DM, DMAIC, DFSS, PMBOK, PDCA, Agile, Lean along with supply chain optimization but we will focus only on a few such as agility.

There are best practices internally or from other organization that can be learnt, adopted and implemented as a strategy.



Data Exploratory (Azure SQL)

	yyww	wrevn	wcost	wdcst	wdays	wtran
0	1701	42,719.25	105341	12014.11	7	6393
1	1702	50,793.81	122713.5	14122.28	7	6746
2	1703	58,537.40	140303.6	16930.82	7	7033
3	1704	68,547.91	167916.1	19944.1	7	7246
4	1705	68,888.15	171464.6	20359.58	7	6598
5	1706	71,783.63	180521.4	25041.62	7	6881
6	1707	79,105.46	198262.3	27488.89	7	7456
7	1708	85,965.55	205904.8	27199.88	7	7811
8	1709	80,168.22	183152.6	21526.24	7	7370
9	1710	73,487.91	166767.6	19720.95	7	7727
10	1711	86,568.57	192688.4	23779.57	7	8471
11	1712	97,756.52	220010.4	24575.86	7	8572
12	1713	88,230.23	197775.4	22265.95	7	8247
13	1714	74,659.87	171442.1	19017.32	7	7834
14	1715	82,249.16	185755.8	20667.07	7	8780
15	1716	83,718.61	189281	22914.08	7	9027
16	1717	111,586.54	229197.4	29526.5	7	9109
17	1718	80,206.75	190000.7	22162.52	7	8386
18	1719	83,565.36	180735.9	24554.77	7	9347
19	1720	84,847.97	193393.5	28422.37	7	9753
20	1721	103,376.29	238541.9	32851.71	7	10087
21	1722	95,084.51	219645	30172.43	7	9406
22	1723	78,831.84	185363.2	25701.03	7	9669
23	1724	81,783.09	191248.3	27400.78	7	10231
24	1725	95,820.66	229065.5	32393.23	7	10157
25	1726	144,156.26	354188.4	50049.32	7	9859
26	1727	93,317.51	226899	31338.1	7	8770
27	1728	67,201.68	156963.7	21810.6	7	9242
28	1729	65,945.80	155656.8	21453.5	7	9417
29	1730	78,610.06	178271.2	24290.93	7	9278
30	1731	69,636.32	158280.6	22922.79	7	8572
31	1732	65,189.83	154113.1	23260.68	7	9301
32	1733	67,918.64	159892	23322.3	7	9480
33	1734	80,720.43	199714.4	25156.27	7	9223
34	1735	75,998.20	165812.6	25189.21	7	8689
35	1736	69,171.67	157988.9	24721.84	7	8479
36	1737	84,647.85	190472.9	27834.46	7	9287
37	1738	84,255.86	188395.9	28422.02	7	9279
38	1739	91,885.05	196731.8	27433.41	7	9184
39	1740	72,379.84	160684.6	23516.13	7	8507
40	1741	72,556.37	164341.4	25065.79	7	9346
41	1742	74,851.55	168190.6	27003.85	7	9294
42	1743	81,360.00	180360.3	28179.38	7	9216
43	1744	79,557.02	169037.2	26675	7	8689
44	1745	72,004.52	159620.8	23549.53	7	8939
45	1746	84,649.47	178111.2	25550.64	7	9402
46	1747	97,140.70	204794.5	29780.79	7	9393
47	1748	96,518.94	211758.4	32938.35	7	9171
48	1749	85,878.63	193884.1	27127.28	7	8826
49	1750	93,980.87	202917.8	29153.11	7	8858
50	1751	105,139.20	241918.4	39752.4	7	8938
51	1802	58,091.37	146364.5	21817.67	7	7379
52	1803	69,421.78	164654.2	23459.55	7	7603
53	1804	68,678.23	165374.5	21718.98	7	7320
54	1805	58,352.29	138952.6	20693.94	7	7206
55	1806	77,428.52	160730.9	26375.54	7	7743
56	1807	88,148.33	161067.8	31906.94	7	7979
57	1808	89,621.52	161403.7	28619.93	7	7656
58	1809	84,710.75	151493	22985.7	7	7775
59	1810	90,107.62	161860.2	24780.25	7	8471
60	1811	92,223.02	168932	28442.73	7	8844
61	1812	104,136.47	192856	36265.1	7	8749
62	1813	67,285.78	124774.1	24151.85	7	7936
63	1814	79,430.67	139631.4	27616.78	7	8598
64	1815	90,274.73	160129.6	31154.96	7	8914
65	1816	97,666.50	169318.8	34645.22	7	8776
66	1817	75,942.53	136192	25974.81	7	7979
67	1818	72,733.01	127078.7	25594	7	8212
68	1819	96,165.54	149370	31760.74	7	8580
69	1820	102,156.97	159016.7	36575.75	7	8439
70	1821	110,705.32	164596.6	36822.65	7	7972
71	1822	89,694.52	129792.4	31752.68	7	7386
72	1823	103,452.25	147501.5	31677.8	7	7810
73	1824	107,907.33	171426	36868.68	7	8071
74	1825	140,263.65	217930.3	44963.18	7	8483
75	1826	76,100.05	110807.5	21452.67	7	7118
76	1827	82,774.90	116998.8	24807.35	7	7869
77	1828	99,106.28	128212.6	31094	7	8735
78	1829	107,022.61	141356.8	35310.57	7	8872
79	1830	94,113.83	129786.8	27793.02	7	7882
80	1831	86,243.03	126646	28413.18	7	8222
81	1832	98,067.03	137421.9	32864.46	7	9010
82	1833	108,644.23	143255.1	32738.39	7	9134
83	1834	121,912.14	160208.9	33400.46	7	8870
84	1835	84,663.46	118399.7	24619.91	7	7744
85	1836	105,811.21	142764.8	30442.28	7	8570
86	1837	128,314.63	162323	34947.99	7	9387

The transactional table has a year and 9 months long spread across 781,652 rows and the other table which has 859 rows segregating that many customers into two segment A and B. I blended the two into dbo.data and added yyww column representing year and week of the year. In this analysis we will focus only on segment A.

Below is weekly report sample code

```
import pyodbc
import pandas as pd

server = 'mysqlserver12345-
  takehome.database.windows.net'
database = 'mySampleDatabase'
username = 'azureuser'
password = 'Thx22azure'
driver= '{ODBC Driver 17 for SQL Server}'

connection =
  pyodbc.connect('DRIVER='+driver+';SERVER='+server+';P
  ORT=1433;DATABASE='+database+';UID='+username+';PWD='
  + password)

sql = """

select
        yyww, sum(revenue) as wrevn,
        sum(cost) as wcost,
        sum(datacost) as wdcst,
        count(distinct date) as wdays,
        count(yyww) as wtran
from (
select segment, time as date, revenue, cost, datacost,
  yyww
from dbo.data
where segment = 'Segment A') as t
where yyww <> 1752 and yyww <> 1801 and yyww <> 1838
group by yyww
order by yyww;

"""

data = pd.read_sql(sql,connection)
display(pd.read_sql(sql, connection))

connection.close()
```

3 weeks are excluded as they are not comparable. See below

```
select yyww, count(distinct time) as num_days from data
where yyww in (1752, 1801, 1838) group by yyww order by
  yyww
```

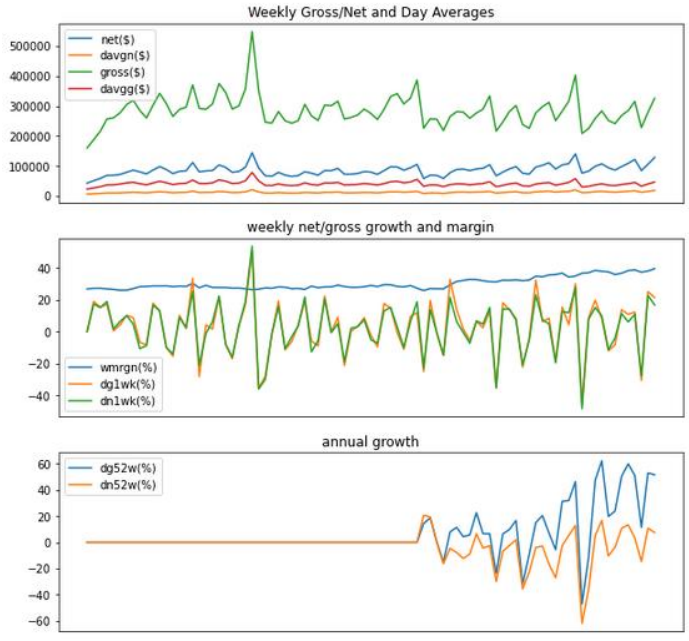
yyww	num_days
1752	8
1801	13
1838	1

Further Data Exploratory (Python)

There were few more queries needed from SQL to accomplish the same result below which I also made in Jupyter Notebook to share easily. Below link summarized the reports using Pandas https://colab.research.google.com/github/FariusGitHub/DataScience/blob/master/sql.ipynb

The report below summarized weekly net sales, daily net sales, weekly gross sales, daily gross sales, weekly margin, weekly margin changes and annual margin changes.

yyww	net($)	davgn($)	gross($)	davgg($)	wmrgn(%)	dg1wk(%)	dn1wk(%)	dg52w(%)	dn52w(%)
1701	42719	6102	160074	22867	26.69	0.00	0.00	0.00	0.00
1702	50793	7256	187628	26804	27.07	18.90	17.21	0.00	0.00
1703	58537	8362	215770	30824	27.13	15.25	15.00	0.00	0.00
1704	68547	9792	256407	36629	26.73	17.10	18.83	0.00	0.00
1705	68888	9841	260711	37244	26.42	0.50	1.68	0.00	0.00
...	...	...	...	...	...	...	...	...	...
1833	108644	15520	284637	40662	38.17	10.79	6.07	59.96	13.34
1834	121912	17416	315520	45074	38.64	12.21	10.85	51.03	3.25
1835	84663	12094	227681	32525	37.18	-30.55	-27.84	11.40	-14.73
1836	105811	15115	279017	39859	37.92	24.98	22.55	52.97	10.77
1837	128314	18330	325583	46511	39.41	21.27	16.69	51.59	7.47



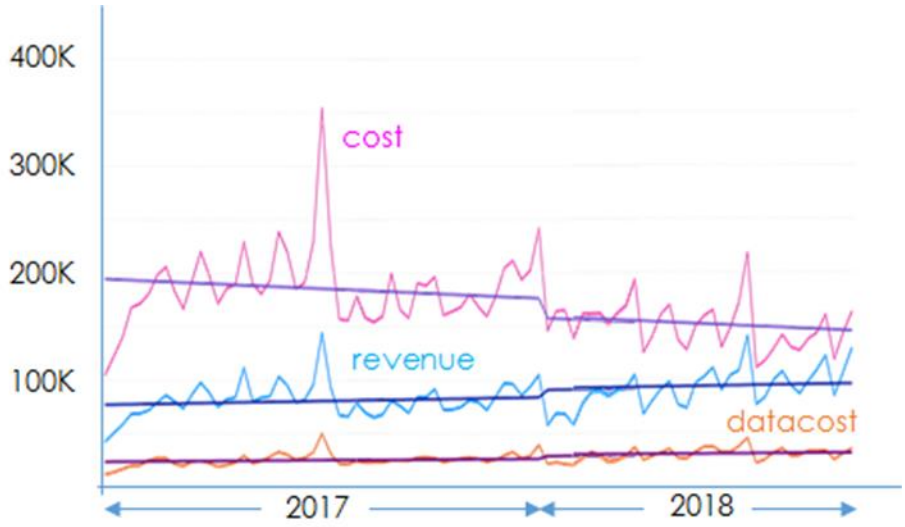
Agility Analysis

Example of Power BI DAX code to generate estimated revenue (net sales) using linear regression.

```
Estimated revenue =
VAR Known =
FILTER (
SELECTCOLUMNS (
ALLSELECTED ( S3[yyww] ),
"Known[X]", S3[yyww],
"Known[Y]", S3[Actual revenue]
),
AND (
NOT ( ISBLANK ( Known[X] ) ),
NOT ( ISBLANK ( Known[Y] ) )
)
)
VAR Count_Items =
COUNTROWS ( Known )
VAR Sum_X =
SUMX ( Known, Known[X] )
VAR Sum_X2 =
SUMX ( Known, Known[X] ^ 2 )
VAR Sum_Y =
SUMX ( Known, Known[Y] )
VAR Sum_XY =
SUMX ( Known, Known[X] *
Known[Y] )
VAR Average_X =
AVERAGEX ( Known, Known[X] )
VAR Average_Y =
AVERAGEX ( Known, Known[Y] )
VAR Slope =
DIVIDE (
Count_Items * Sum_XY - Sum_X *
Sum_Y,
Count_Items * Sum_X2 - Sum_X ^ 2
)
VAR Intercept =
Average_Y - Slope * Average_X
RETURN
SUMX (
DISTINCT ( S3[yyww] ),
Intercept + Slope * S3[yyww]
)
```

Credit: Daniil Maslyuk, 10 September 2017
xxlbi.com/blog/simple-linear-regression-in-dax

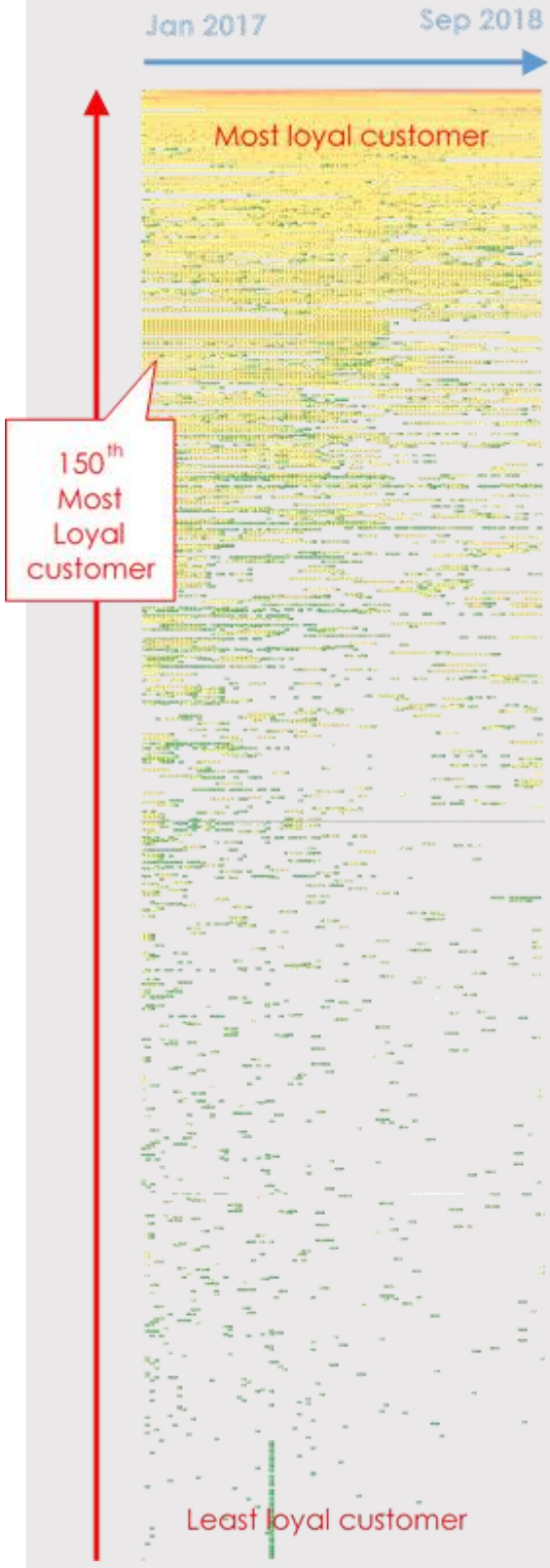
Using DAX code on the left we could drew regression lines on each curve as below. The revenue went up as the datacost went up too although the cost decreased.



Based on segment A analysis we could see the successful cost reduction of other cost (such as tower operation, salary, data centre leases, HVAC, etc) but not on the datacost regardless the increasing margin.

One way to reduce data cost could be negotiation power. As the cellular service company does not have any power to reinvent the technology (5G) the satellite and tower services companies and 5G licenses can be negotiated with higher volume of usage. The broadband prediction along with traditional usage could be a challenge especially when the consumer spending power is weakened. Other opportunities include waste elimination, variation reduction such as for an operational case in energy storage to reduce energy cost.

Loyalty Analysis



Traditionally the loyalty of the customer could be visualized using 2D spectrum based on the customer loyalty rank and timeline on the left which show how many time each customer did business. There are 87 months in total and the most loyal customer did transaction on all the months where the least may only did once.

At the left chart at after about the 150th customer (18% of total customer) we could expect the customer only stay half of the time (half of the whole 87 weeks). As the chart goes lower more blanks will be seen. This chart was drawn with Excel conditional formatting and screen shot at the zoom out of 10%.

Thanks to Sand Dance that could visualize above spectrum into 3D like below where we could orbit the skyscraper blocks and inspect how many loyal customer in 2017 compared to 2018. The 831 customers were grouped into 9 bins on id axis and the 87 months were grouped into 27 bins.

We could see less population for layer one and layer two of the most loyal customer from 2017 to 2018. Subsequent population from each layer also reduced that suggested weaker campaign although the cost declined (see purple line from the previous page) that might have correlation with less loyal customer.