Business Analytics Final Report

# LOLCODE

Yuhan Yao, Mengjie Shen, Xue Bai, Ling'er Zhang, Thomas Yee

Spring Semester of May 22nd, 2020

Link to the presentation: here

# Table of Contents

# Introduction

## Overview of Business Analytics

As the world is becoming more and more intellectual, business is not pure business anymore Instead, real business is the combination of data analysis and finance tradeoff and then it has the concept of "business analytics". It's a research discipline of identifying business needs and determining solutions to business problems through the analytics of data. The scope of business analytics is very wide, from pure software improvements to company's deal optimization. However, the ultimate goal of business analytics is to create solutions, give enough tools for robust project management, improve efficiency and reduce waste, and provide essential documentation.

Essentially, there are 7 steps in most business analytics cases, outlined below:

- Defining the business needs (The first step is always understanding what the business wants to improve as well as understanding the background of this business analytics case)
- Explore the data (It includes cleaning the data, removing the outliers, understanding the meaning of each data as well as identifying the inner relationship between each variable)
- Analyze the data (Using various method, including simple linear regression, KNN, multivariable regression logistic regression, etc. to derive insights from the data)
- Predict what is likely to happen (Use the model from the previous step wo predict what might happen afterwards, it is also the key step in business analytics)
- Optimize (Given the constraints and limitations, find the best solution model among all the statistic method we used)
- Make a decision (Based on the derived insights from the model, then the analyst needs to make a decision and take the action)
- Update the system with the results of the decision (Update all the results into the database since an evolving database that is continuously updated can provide new insights)

## Eleme Delivery Analytics Under COVID-19

As e-commerce booms and more and more people prefer to eat at home, the food delivery industry has been a high-growth industry which attracts both investors and competitors. To support the service of online food ordering, first we need the online platforms which enable convenient food ordering and delivery service for customers. While there are platforms such as Grubhub and UberEATS in the US, the main online ordering platforms in China are Elema and Meituan. Either case one of the key challenges to provide a better online delivery service is how to predict the delivery man's next action type and time so overall the platform can provide the customers with on-time delivery.

Within the context of Eleme delivery service, at every moment the platform sends orders from customers to the delivery man for instant delivery. The decisions being made by the delivery man are mainly two-kind: pick-up in store, and delivery to customers. At one specific moment, the delivery man may receive orders assigned to him, while he has unfinished orders which were previously assigned. He would need to make a decision about what to do next, based on his order status and geographical location. Therefore, the goal of this analytics is to build a model to predict the delivery-man's next action, based on his historical decision and current status.

## Basic Steps

Basically, we split this particular Elema delivery case into 3 stages: data exploration, data analysis, and key takeaways.

### Data Exploration

It is the combination of the above first two steps. To better understand the business needs and make the analysis as well as the prediction, the first thing we did is figuring out the meaning behind each variable and the potential connections among them. Also, we tried to combine all the data into one datasheet which is more convenient for the following analysis. Moreover, we did the real-word Elema ordering to help us deepen the understanding of relevant data and provide us with insights beyond the data.

### Data Analysis

This step includes the analyze, predict, optimize, and decide stage. Among regression techniques, we use linear regression and PCA, logistic regression and ridge regression to analyze the data. Besides, we also tried other methods like XGBoost to make predictions. And among all the tryouts, the winning formula which is the optimized decision is based on the Cocktail Method (combined Top-Down Approach and Bottom-Up Approach)

### Insights and Conclusions

This can also be seen as the last step of updating information. From all the exploration and analysis above, each group member comes up with their understanding of this project and some personal takeaways of "business analytics" subject. Also, we propose some considerations and questions that might contribute to the future prediction of this Elema delivery case.

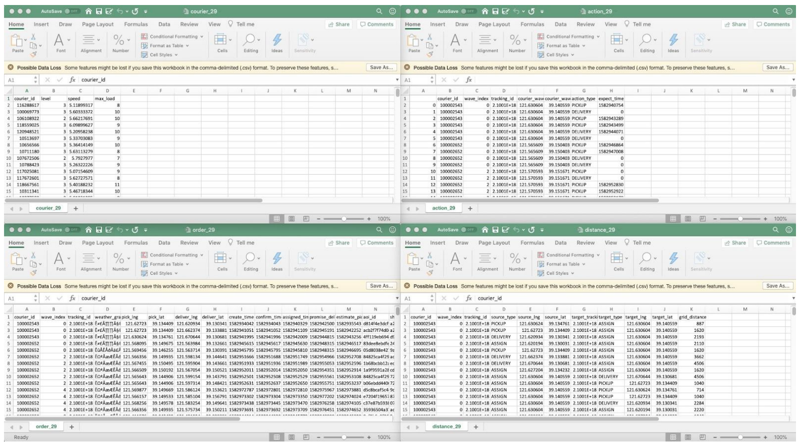
# Preliminary Data Exploration

## Parsing the Data

The first step to our analysis was combining and cleaning up the data to fit our needs and prepare for analysis. We had to understand the data before we could do anything with it. However, we also used this period of time effectively by asking extensive amounts of questions and quickly understanding the relationships between action, courier, distance, and order datasheets.

### Initial Combining of Datasheets

We started off by blindly using Pandas using Python to combine the data of all 28 days into one file. Obviously, this did not work for various reasons. Even my joining things together, we quickly realized that for every row in “order” meant 2 rows for “action”. For every “action” row mean row$^2$*2 rows in “distance”. And for every number for couriers in any of these data sheets meant 1 corresponding row in the “courier”. Thus, by combining all the days together without any reason, we were lost and had to start all over again, given that we actually made the data more messy rather than cleaning it up.



Possible Data Loss Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, s...

courier_29 — A1 courier_id

| | A | B | C | D |
|---|---|---|---|---|
| 1 | courier_id | level | speed | max_load |
| 2 | 116288617 | 3 | 5.11899317 | 8 |
| 3 | 100069773 | 3 | 5.60333372 | 10 |
| 4 | 106108922 | 2 | 5.66217691 | 10 |
| 5 | 118559025 | 3 | 6.09899627 | 9 |
| 6 | 120948521 | 3 | 5.20958238 | 10 |
| 7 | 10513697 | 3 | 5.33703083 | 9 |
| 8 | 10656566 | 3 | 5.36414149 | 10 |
| 9 | 10711180 | 3 | 5.63113279 | 8 |
| 10 | 107672506 | 2 | 5.7927977 | 7 |
| 11 | 10788423 | 3 | 5.26322226 | 9 |
| 12 | 117025081 | 3 | 5.07154609 | 9 |
| 13 | 117672601 | 3 | 5.62727571 | 8 |
| 14 | 118667561 | 3 | 5.40188232 | 11 |
| 15 | 10311341 | 3 | 5.46718344 | 10 |

action_29 — A1

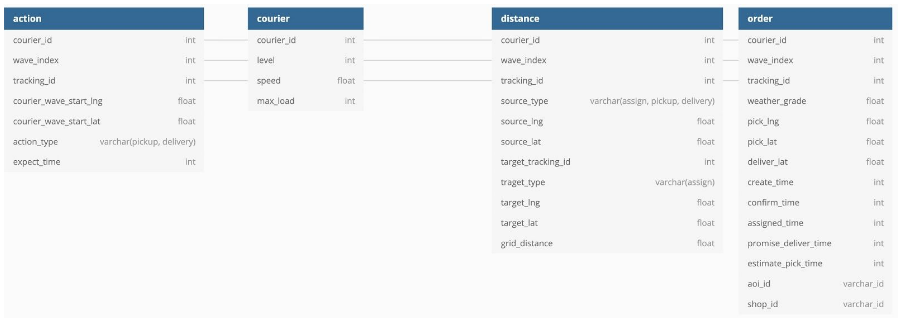
| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | | courier_id | wave_index | tracking_id | courier_wav | courier_wav | action_type | expect_time |
| 2 | 0 | 100002543 | 0 | 2.1001E+18 | 121.630604 | 39.140559 | PICKUP | 1582940754 |
| 3 | 1 | 100002543 | 0 | 2.1001E+18 | 121.630604 | 39.140559 | DELIVERY | 0 |
| 4 | 2 | 100002543 | 0 | 2.1001E+18 | 121.630604 | 39.140559 | PICKUP | 1582943289 |
| 5 | 3 | 100002543 | 0 | 2.1001E+18 | 121.630604 | 39.140559 | PICKUP | 1582943499 |
| 6 | 4 | 100002543 | 0 | 2.1001E+18 | 121.630604 | 39.140559 | DELIVERY | 1582944071 |
| 7 | 5 | 100002543 | 0 | 2.1001E+18 | 121.630604 | 39.140559 | DELIVERY | 0 |
| 8 | 6 | 100002652 | 0 | 2.1001E+18 | 121.565609 | 39.150403 | PICKUP | 1582946864 |
| 9 | 7 | 100002652 | 0 | 2.1001E+18 | 121.565609 | 39.150403 | PICKUP | 1582947008 |
| 10 | 8 | 100002652 | 0 | 2.1001E+18 | 121.565609 | 39.150403 | DELIVERY | 0 |
| 11 | 9 | 100002652 | 0 | 2.1001E+18 | 121.565609 | 39.150403 | DELIVERY | 0 |
| 12 | 10 | 100002652 | 2 | 2.1001E+18 | 121.570593 | 39.151671 | PICKUP | 0 |
| 13 | 11 | 100002652 | 2 | 2.1001E+18 | 121.570593 | 39.151671 | DELIVERY | 0 |
| 14 | 12 | 100002652 | 2 | 2.1001E+18 | 121.570593 | 39.151671 | PICKUP | 1582952830 |
| 15 | 13 | 100002652 | 2 | 2.1001E+18 | 121.570593 | 39.151671 | PICKUP | 1582952922 |

order_29 — A1 courier_id

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | courier_id | wave_index | tracking_id | weather_gra | pick_lng | pick_lat | deliver_lng | deliver_lat | create_time | confirm_tim | assigned_tin | promise_del | estimate_pic | aoi_id | sh |
| 2 | 100002543 | 0 | 2.1001E+18 | Ê≠£ÅΠΠÅ§( | 121.62723 | 39.134409 | 121.620934 | 39.130341 | 1582934042 | 1582934043 | 1582940329 | 1582942500 | 1582935543 | d814f4e3dcf | a2 |
| 3 | 100002543 | 0 | 2.1001E+18 | Ê≠£ÅΠΠÅ§( | 121.62723 | 39.134409 | 121.662374 | 39.133881 | 1582941051 | 1582941052 | 1582941109 | 1582945191 | 1582942252 | acb2f7f74d0 | a2 |
| 4 | 100002543 | 0 | 2.1001E+18 | Ê≠£ÅΠΠÅ§( | 121.630624 | 39.134761 | 121.670644 | 39.130681 | 1582941995 | 1582941996 | 1582942009 | 1582944815 | 1582943256 | 4ff119eb694 | d9 |
| 5 | 100002652 | 0 | 2.1001E+18 | Ê≠£ÅΠΠÅ§( | 121.568095 | 39.149675 | 121.563984 | 39.132661 | 1582945615 | 1582945617 | 1582945630 | 1582948315 | 1582946517 | 83dee8ebdfe | 24 |
| 6 | 100002652 | 0 | 2.1001E+18 | ÊûÅÊÅðÅä£Å | 121.569456 | 39.146255 | 121.563714 | 39.130391 | 1582945795 | 1582945795 | 1582945810 | 1582948315 | 1582946695 | 05d8038e42 | 9a |
| 7 | 100002652 | 2 | 2.1001E+18 | ÊΩ¤ÅæÆÊÅd | 121.566356 | 39.149935 | 121.598134 | 39.144641 | 1582951666 | 1582951688 | 1582951749 | 1582954966 | 1582952708 | 84825ca4f29 | ac |
| 8 | 100002652 | 2 | 2.1001E+18 | ÊΩ¤ÅæÆÊÅd | 121.567455 | 39.150495 | 121.595904 | 39.143661 | 1582951933 | 1582951936 | 1582951989 | 1582955053 | 1582952596 | 1b68bcbb12 | ec |
| 9 | 100002652 | 2 | 2.1001E+18 | ÊΩ¤ÅæÆÊÅd | 121.566509 | 39.150192 | 121.567054 | 39.150521 | 1582952011 | 1582952014 | 1582952050 | 1582954351 | 1582952914 | 1a9f9591a28 | cd |
| 10 | 100002652 | 2 | 2.1001E+18 | ÊΩ¤ÅæÆÊÅd | 121.565643 | 39.144906 | 121.599154 | 39.143791 | 1582952501 | 1582952508 | 1582952529 | 1582955561 | 1582953108 | 84825ca4f29 | 72 |
| 11 | 100002652 | 2 | 2.1001E+18 | ÊΩ¤ÅæÆÊÅd | 121.565643 | 39.144906 | 121.597314 | 39.148421 | 1582952631 | 1582952637 | 1582952650 | 1582955751 | 1582953237 | b06ebdd440 | 72 |
| 12 | 100002652 | 4 | 2.1001E+18 | ÊΩ¤ÅæÆÊÅd | 121.569877 | 39.149669 | 121.586124 | 39.153621 | 1582972787 | 1582972801 | 1582972810 | 1582975967 | 1582973881 | d5c8bcaf5c4 | 9c |
| 13 | 100002652 | 4 | 2.1001E+18 | ÊΩ¤ÅæÆÊÅd | 121.566157 | 39.149533 | 121.585104 | 39.156791 | 1582973302 | 1582973304 | 1582973350 | 1582977202 | 1582974024 | e7204f19657 | 87 |
| 14 | 100002652 | 4 | 2.1001E+18 | ÊΩ¤ÅæÆÊÅd | 121.568256 | 39.149578 | 121.583254 | 39.149641 | 1582973438 | 1582973445 | 1582973470 | 1582976258 | 1582974105 | c37e87b593 | 07 |
| 15 | 100002652 | 4 | 2.1001E+18 | ÊΩ¤ÅæÆÊÅd | 121.566356 | 39.149935 | 121.575734 | 39.150211 | 1582973691 | 1582973692 | 1582973709 | 1582976451 | 1582974652 | 35936504a3 | ac |

distance_29 — A1 courier_id

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | courier_id | wave_index | tracking_id | source_type | source_lng | source_lat | target_tracki | target_type | target_lng | target_lat | grid_distance |
| 2 | 100002543 | 0 | 2.1001E+18 | PICKUP | 121.630624 | 39.134761 | 2.1001E+18 | ASSIGN | 121.630604 | 39.140559 | 887 |
| 3 | 100002543 | 0 | 2.1001E+18 | PICKUP | 121.62723 | 39.134409 | 2.1001E+18 | ASSIGN | 121.630604 | 39.140559 | 1620 |
| 4 | 100002543 | 0 | 2.1001E+18 | DELIVERY | 121.620934 | 39.130341 | 2.1001E+18 | ASSIGN | 121.630604 | 39.140559 | 2193 |
| 5 | 100002543 | 0 | 2.1001E+18 | ASSIGN | 121.620194 | 39.130031 | 2.1001E+18 | ASSIGN | 121.630604 | 39.140559 | 2110 |
| 6 | 100002543 | 0 | 2.1001E+18 | PICKUP | 121.62723 | 39.134409 | 2.1001E+18 | ASSIGN | 121.630604 | 39.140559 | 1620 |
| 7 | 100002543 | 0 | 2.1001E+18 | DELIVERY | 121.662374 | 39.133881 | 2.1001E+18 | ASSIGN | 121.630604 | 39.140559 | 3662 |
| 8 | 100002543 | 0 | 2.1001E+18 | DELIVERY | 121.670644 | 39.130681 | 2.1001E+18 | ASSIGN | 121.630604 | 39.140559 | 4506 |
| 9 | 100002543 | 0 | 2.1001E+18 | ASSIGN | 121.627204 | 39.134232 | 2.1001E+18 | ASSIGN | 121.630604 | 39.140559 | 1620 |
| 10 | 100002543 | 0 | 2.1001E+18 | ASSIGN | 121.630604 | 39.140559 | 2.1001E+18 | DELIVERY | 121.670644 | 39.130681 | 4506 |
| 11 | 100002543 | 0 | 2.1001E+18 | ASSIGN | 121.630604 | 39.140559 | 2.1001E+18 | PICKUP | 121.62723 | 39.134409 | 1040 |
| 12 | 100002543 | 0 | 2.1001E+18 | ASSIGN | 121.630604 | 39.140559 | 2.1001E+18 | PICKUP | 121.630624 | 39.134761 | 714 |
| 13 | 100002543 | 0 | 2.1001E+18 | ASSIGN | 121.630604 | 39.140559 | 2.1001E+18 | PICKUP | 121.62723 | 39.134409 | 1040 |
| 14 | 100002543 | 0 | 2.1001E+18 | ASSIGN | 121.630604 | 39.140559 | 2.1001E+18 | DELIVERY | 121.620934 | 39.130341 | 2284 |
| 15 | 100002543 | 0 | 2.1001E+18 | ASSIGN | 121.630604 | 39.140559 | 2.1001E+18 | ASSIGN | 121.620194 | 39.130031 | 2220 |



| action | |
|---|---|
| courier_id | int |
| wave_index | int |
| tracking_id | int |
| courier_wave_start_lng | float |
| courier_wave_start_lat | float |
| action_type | varchar(pickup, delivery) |
| expect_time | int |

| courier | |
|---|---|
| courier_id | int |
| level | int |
| speed | float |
| max_load | int |

| distance | |
|---|---|
| courier_id | int |
| wave_index | int |
| tracking_id | int |
| source_type | varchar(assign, pickup, delivery) |
| source_lng | float |
| source_lat | float |
| target_tracking_id | int |
| traget_type | varchar(assign) |
| target_lng | float |
| target_lat | float |
| grid_distance | float |

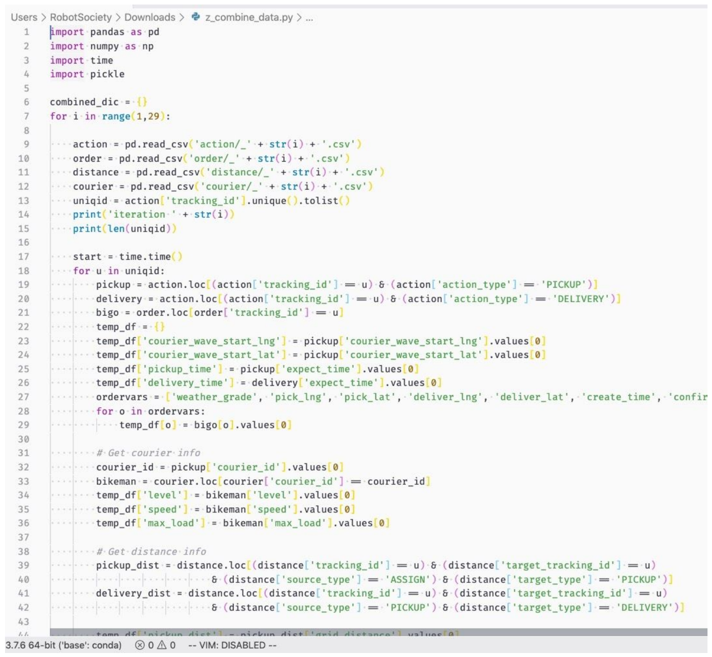
| order | |
|---|---|
| courier_id | int |
| wave_index | int |
| tracking_id | int |
| weather_grade | float |
| pick_lng | float |
| pick_lat | float |
| deliver_lat | float |
| create_time | int |
| confirm_time | int |
| assigned_time | int |
| promise_deliver_time | int |
| estimate_pick_time | int |
| aoi_id | varchar_id |
| shop_id | varchar_id |

## Errors in Merges

By this point, we were already frustrated with ourselves, because we didn't understand how we could Merge everything together. Initially, we thought that doing outer joins and concatenation would solve all our problems, and possibly transform the rows of each data set to be the same amount of rows. However, this proved to be difficult as we tried to duplicate information and make Pandas.merge work correctly. We made the mistake of continuously merging things together after duplicating, and this resulted in a finalized document with over 4 gigabytes of merged data, which wasn't even the combination of everything because the computer froze and died before completing. By this point, we realized we needed to understand the dataset more in order to parse the data more clearly.

## Finalized Way We Combined Everything Together

The solution to getting the data we wanted parsed was by manually taking all of the data and rewriting everything to our very own dataframe given a specific unique identifier (in this case it was Tracking ID). Afterward, we used a python script to combine all data given a specific datasheet, and then saved these datas as a list of dictionaries that we later saved as a pickle file (since this is a way to store data frames efficiently and would be easier to work with given that a pickle can loop more efficiently than a csv).



Users > RobotSociety > Downloads > z_combine_data.py > ...

```python
import pandas as pd
import numpy as np
import time
import pickle

combined_dic = {}
for i in range(1,29):

    action = pd.read_csv('action/_' + str(i) + '.csv')
    order = pd.read_csv('order/_' + str(i) + '.csv')
    distance = pd.read_csv('distance/_' + str(i) + '.csv')
    courier = pd.read_csv('courier/_' + str(i) + '.csv')
    uniqid = action['tracking_id'].unique().tolist()
    print('iteration ' + str(i))
    print(len(uniqid))

    start = time.time()
    for u in uniqid:
        pickup = action.loc[(action['tracking_id'] == u) & (action['action_type'] == 'PICKUP')]
        delivery = action.loc[(action['tracking_id'] == u) & (action['action_type'] == 'DELIVERY')]
        bigo = order.loc[order['tracking_id'] == u]
        temp_df = {}
        temp_df['courier_wave_start_lng'] = pickup['courier_wave_start_lng'].values[0]
        temp_df['courier_wave_start_lat'] = pickup['courier_wave_start_lat'].values[0]
        temp_df['pickup_time'] = pickup['expect_time'].values[0]
        temp_df['delivery_time'] = delivery['expect_time'].values[0]
        ordervars = ['weather_grade', 'pick_lng', 'pick_lat', 'deliver_lng', 'deliver_lat', 'create_time', 'confir
        for o in ordervars:
            temp_df[o] = bigo[o].values[0]

        # Get courier info
        courier_id = pickup['courier_id'].values[0]
        bikeman = courier.loc[courier['courier_id'] == courier_id]
        temp_df['level'] = bikeman['level'].values[0]
        temp_df['speed'] = bikeman['speed'].values[0]
        temp_df['max_load'] = bikeman['max_load'].values[0]

        # Get distance info
        pickup_dist = distance.loc[(distance['tracking_id'] == u) & (distance['target_tracking_id'] == u)
                                    & (distance['source_type'] == 'ASSIGN') & (distance['target_type'] == 'PICKUP')]
        delivery_dist = distance.loc[(distance['tracking_id'] == u) & (distance['target_tracking_id'] == u)
                                    & (distance['source_type'] == 'PICKUP') & (distance['target_type'] == 'DELIVERY')]

        temp_df['pickup_dist'] = pickup_dist['grid_distance'].values[0]
```

3.7.6 64-bit ('base': conda)   ⓧ 0 ⚠ 0   -- VIM: DISABLED --

data.p

data2.p

data28.p

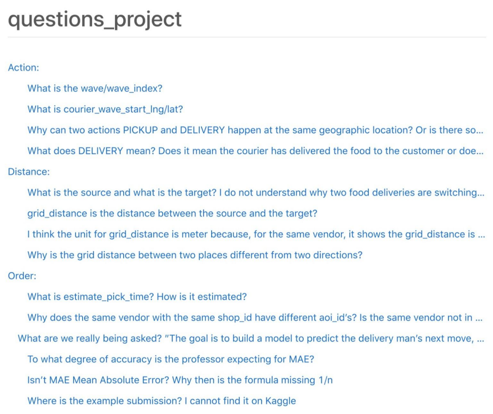
data29.p

## Struggles in Handling Large Data

By having an initial struggle in merges, pandas, and pickling the data, the next step was about understanding the large amount of data we had successfully parsed. This period of time took the longest, as we spent a whole month just asking questions to the professor and constantly having Office Hour Calls to dive deep into our understanding.

### Questions

The first step we needed to do was ask questions about the definitions of our variables. Even with the information available on Kaggle about the data, we felt this was still not enough and therefore compiled a whole list of questions. Some preliminary questions that helped guide our form of analysis later on were questions such as: “why is the grid distance between two places different from two directions?” or “what is the estimate_pick_time and how is it estimated?” These questions really helped in our understanding and getting prompt answers from the professor helped define the problem more clearly. For the full initial questions, visit here.



questions_project

Action:

- What is the wave/wave_index?
- What is courier_wave_start_lng/lat?
- Why can two actions PICKUP and DELIVERY happen at the same geographic location? Or is there so...
- What does DELIVERY mean? Does it mean the courier has delivered the food to the customer or doe...

Distance:

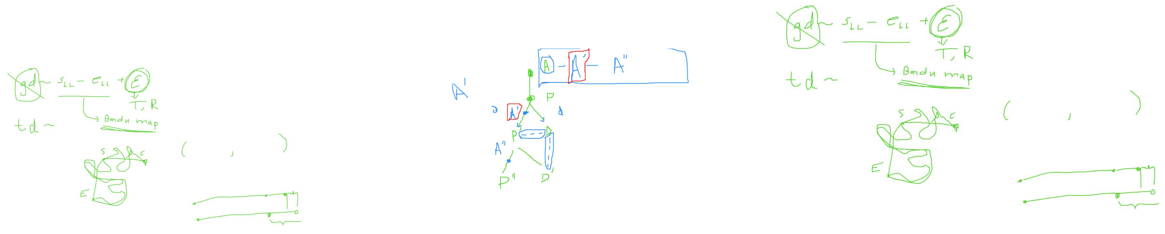
- What is the source and what is the target? I do not understand why two food deliveries are switching...
- grid_distance is the distance between the source and the target?
- I think the unit for grid_distance is meter because, for the same vendor, it shows the grid_distance is ...
- Why is the grid distance between two places different from two directions?

Order:

- What is estimate_pick_time? How is it estimated?
- Why does the same vendor with the same shop_id have different aoi_id's? Is the same vendor not in ...
- What are we really being asked? "The goal is to build a model to predict the delivery man's next move, ...
- To what degree of accuracy is the professor expecting for MAE?
- Isn't MAE Mean Absolute Error? Why then is the formula missing 1/n
- Where is the example submission? I cannot find it on Kaggle

### Office Hours and Answers

The preliminary questions were really useful, however we wanted to dive further and ask the professor more so we could understand the data in real time, and not go through a back and forth pdf. Therefore, we would meet most weeks in the last month to ask questions about how action types were decided upon, what the Eleme driver sees, and potential problems with the data (given we might not have every single variable that makes the driver decide his actions). Our questions were met with interesting answers, in the form that firstly, wave index was not as important a factor as we initially understood it to be. Secondly, we found a potential problem in the way Eleme drivers might decide their delivery orders given that the data showed there were instances for every wave in which a pickup driver could be assigned a value prior to them finishing the pickup of the previous order. This meant that assignments were not instantaneous at t=0 (aka when the wave starts). This confusion on their options made it hard to move forward given that it seemed like the Eleme Driver could be given direction from the app to do random assignments which would be hard for us to predict. For this analysis, we tried to ignore this problem, but the professor gave us comfort that we shouldn't give up and ask questions maybe to real life Eleme Drivers and position our analysis to not overthink the problem given that we don't have all the data in the world and that there could be statistical discrepancies. Therefore, this gave us the idea to research past the datasets to understand the more contextual relevance of variables and how our data was constructed in the first place.



## Eleme Exploration

### Ark System in Eleme

We believe it is essential to see from a bigger picture and find out where our course project actually fits in Eleme's broader operation strategy. From our research, we found out Eleme has been upgrading their AI delivery system called "Ark" for a couple of years. Their system has been through at least 5 iterations.

**Iteration 1: Optimizing Order Details**

This stage took delivery remaining time, number of deliveries couriers are carrying and detour distance into consideration. However, couriers do not always follow the Ark system-assigned optimal routes (scooter malfunction for instance), this iteration did not meet the expectations.

**Iteration 2: Introducing the Concept of "Buffer"**

By introducing the concept of buffer, this stage gets rid of individuality and does not see every delivery as a separated task. Instead, this stage reaches optimality from a higher level. However, this stage did less well during peak hours.

**Iteration 3: Using Wave Index to Combine Different Orders**

This stage combined several deliveries into waves and highlighted the deliveries that were marked urgent. However, the traditional model could no longer handle the complexity with this stage.

**Iteration 4: Start to Use Extensive Machine Learning**

This stage built five machine learning models. This stage works perfectly well in theory, but when applied in real life, human calibrations are needed extensively.

**Iteration 5: Use Deep Neural Networks**

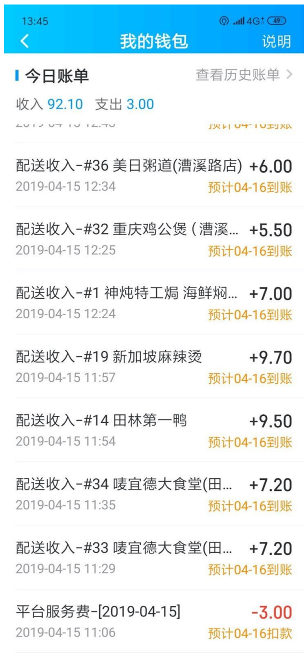
In this stage, the deep learning model takes care of almost all predictions, including route estimation and delivery information in the next 15-30 minutes. Also, their algorithm team fine-tuned the model to accommodate different occasions.

Generally speaking, the Ark system focuses on couriers' ability profiles (which is similar to our Courier datasheet with courier level, speed and max load). For example, the Ark system is prone to assigning orders to higher-level couriers during peak hours. The system balances the number of orders for each courier and balances the proportion of time-consuming orders and quick orders. The Ark system also exploits the time period between the order confirmed time and the ready for pickup time at each vendor (similar to Order datasheet). The system takes the weather into consideration and assigns couriers more time to deliver if it is raining to ensure the safety of couriers. The concept of grid (see Order datasheet) is also introduced to solve the problem of cross-grid delivery. As for our project, with all the information we are given, our project seems to be going from iteration 3 to iteration 4 in Eleme's operation strategy. We are seeking ways to build machine learning models to optimize the entire system.

(source: https://cloud.tencent.com/developer/article/1165938)

## Eleme driver exploration

To see how the Eleme drivers act in real word and have a better understanding of the data, we decided to order something via Eleme and asked the delivery man some questions in person regarding their actions of picking up and delivering the food. From the screenshot of the application the delivery man used, we found out that in the page of deciding to accept the order, the delivery man can see the distance between the current location and the pickup site as well as the distance between the current location and the delivery site. Moreover, the delivery man can see the reward he or she can receive after successfully delivering this order. After accepting, these orders will appear in the "in delivery" page which has additional information that is the required delivery time. Overall, the majority of the data can be understood and interpreted based on our previous understanding.

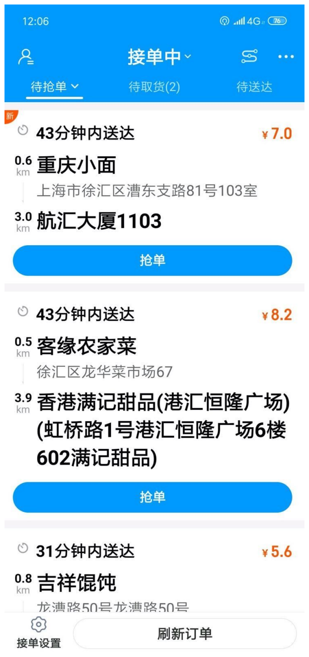

13:45

我的钱包 说明

今日账单 查看历史账单

收入 92.10 支出 3.00

| | |
|---|---|
| 配送收入-#36 美日粥道(漕溪路店) | +6.00 |
| 2019-04-15 12:34 | 预计04-16到账 |
| 配送收入-#32 重庆鸡公煲（漕溪... | +5.50 |
| 2019-04-15 12:25 | 预计04-16到账 |
| 配送收入-#1 神炖特工焗 海鲜焖... | +7.00 |
| 2019-04-15 12:24 | 预计04-16到账 |
| 配送收入-#19 新加坡麻辣烫 | +9.70 |
| 2019-04-15 11:57 | 预计04-16到账 |
| 配送收入-#14 田林第一鸭 | +9.50 |
| 2019-04-15 11:54 | 预计04-16到账 |
| 配送收入-#34 唛宜德大食堂(田... | +7.20 |
| 2019-04-15 11:35 | 预计04-16到账 |
| 配送收入-#33 唛宜德大食堂(田... | +7.20 |
| 2019-04-15 11:29 | 预计04-16到账 |
| 平台服务费-[2019-04-15] | -3.00 |
| 2019-04-15 11:06 | 预计04-16扣款 |


12:06

接单中

新

43分钟内送达 ¥7.0

0.6 km 重庆小面

上海市徐汇区漕东支路81号103室

3.0 km 航汇大厦1103

43分钟内送达 ¥8.2

0.5 km 客缘农家菜

徐汇区龙华菜市场67

3.9 km 香港满记甜品(港汇恒隆广场)(虹桥路1号港汇恒隆广场6楼602满记甜品)

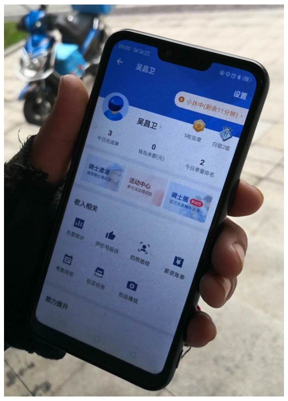
31分钟内送达 ¥5.6

0.8 km 吉祥馄饨

Moreover, there are two additional pieces of information we gathered from the real orders : the reward system and the information from the customer side. From the previous description we can see that the reward system incentives the deliveryman to pick up the high-reward order first which might affect their actions of delivering and picking up. However, this factor is not included in our dataset so further research about the impact of the reward system is needed to improve the prediction and help to have a better understanding of the real action decision. Moreover, in the in-delivery page the delivery man can receive the chat the customer sent. In some cases, the customers might call the delivery man for an early delivery request or ask them whether they haven't successfully delivered the food, all this information from the customer side affects the delivery man's action as well since they need a good rating score afterwards. Therefore, the customers' attitudes and the reward system have an impact on the delivery man's action but are not in the dataset which might affect our accuracy of prediction.

One additional thing we found through this exploration that might affect the delivery man's decision is called the insurance of in-time delivery. It can also be called "zhun shi bao'(准时宝)'in Chinese, which would compensate the customer after the delivery man has a delay. The more times the delivery man is delayed the higher compensation it is. Some stores might provide this kind of insurance to attract the customers' order. However, where the composition comes from is unsure. If the compensation comes from the reward of the order then this insurance can also affect the delivery man's incentives towards particular orders.

We also interviewed an Eleme driver to know more about which factors he values when he makes decisions about the next action. Originally our assumption was that they have the opportunity to choose which order they want to accept. He might take his current geographical locations, previous unfinished orders and the reward he can get into consideration. This will greatly help us in predicting his next action. However, after talking with one driver, we found that the system actually automatically assigns the order to couriers based on their geographical location. Couriers don't have the opportunities to decide which order he can take.



# Analysis of the Data

## Tools

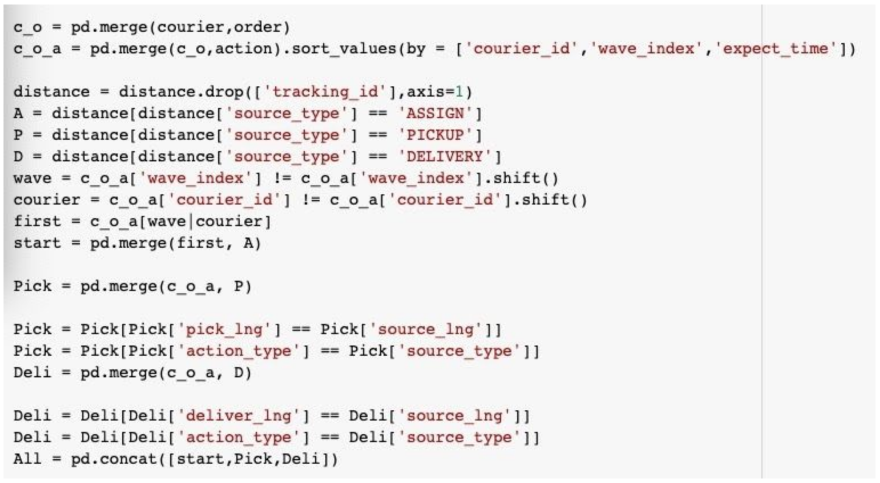
We extensively used excel and python to test out the results. The group firstly had to find ways to express our newfound ways to view data whenever we had a conversation, and therefore we used excel along with a lot of Zoom whiteboards to draw out explorative understandings of the material. After we got a good understanding of the data, we decided to use python more extensively to do all of the different types of regression and methodologies that would later form the analysis.

## Regression Techniques

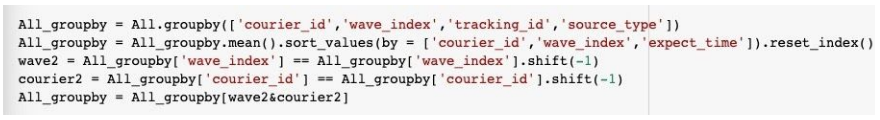
The first methodology we used was to test out proper regression techniques, given that they would help show as an explanatory means to jumpstart our investigations and pinpoint what we would want to look into when constructing an accurate prediction model.

### Linear Regression and PCA

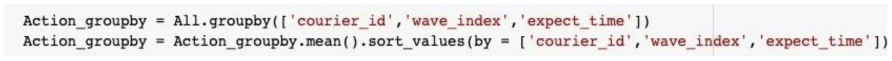
Our group first tried the simplest method, which is the linear regression. Since a courier can have many orders at the same time, it means that there are several possibilities for the courier to choose, each has a different task and location. This makes it more difficult to decide the relationship between dependent variable and independent variables because each final outcome y will have several possibilities x, with each x having several features including starting location, distance, assigned time, etc. Therefore, to integrate the useful information and reduce the number of data, we decide to take the average of all the possibilities of a single outcome. For example, if the courier finished "Pickup" at location B at time T, we can first find the courier's starting location A based on the previous action. Then, we can extract all the possible choices that can be made from location A and calculate the mean distance, action type, assign time and other given information. We then use this mean of all features to predict the real location B and time T.



```
c_o = pd.merge(courier,order)
c_o_a = pd.merge(c_o,action).sort_values(by = ['courier_id','wave_index','expect_time'])

distance = distance.drop(['tracking_id'],axis=1)
A = distance[distance['source_type'] == 'ASSIGN']
P = distance[distance['source_type'] == 'PICKUP']
D = distance[distance['source_type'] == 'DELIVERY']
wave = c_o_a['wave_index'] != c_o_a['wave_index'].shift()
courier = c_o_a['courier_id'] != c_o_a['courier_id'].shift()
first = c_o_a[wave|courier]
start = pd.merge(first, A)

Pick = pd.merge(c_o_a, P)

Pick = Pick[Pick['pick_lng'] == Pick['source_lng']]
Pick = Pick[Pick['action_type'] == Pick['source_type']]
Deli = pd.merge(c_o_a, D)

Deli = Deli[Deli['deliver_lng'] == Deli['source_lng']]
Deli = Deli[Deli['action_type'] == Deli['source_type']]
All = pd.concat([start,Pick,Deli])
```

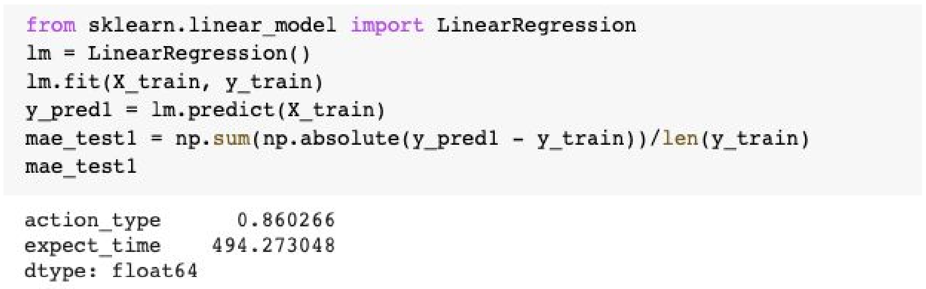


```
All_groupby = All.groupby(['courier_id','wave_index','tracking_id','source_type'])
All_groupby = All_groupby.mean().sort_values(by = ['courier_id','wave_index','expect_time']).reset_index()
wave2 = All_groupby['wave_index'] == All_groupby['wave_index'].shift(-1)
courier2 = All_groupby['courier_id'] == All_groupby['courier_id'].shift(-1)
All_groupby = All_groupby[wave2&courier2]
```



```
Action_groupby = All.groupby(['courier_id','wave_index','expect_time'])
Action_groupby = Action_groupby.mean().sort_values(by = ['courier_id','wave_index','expect_time'])
```

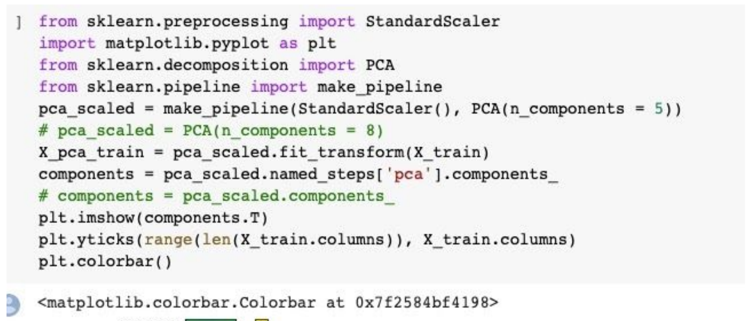
However, by feeding everything to the linear regression model and letting it return a prediction, the outcome was not satisfying. Using the MAE function, we got a loss of 494 for expected time and almost no accuracy for the action type prediction.



```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X_train, y_train)
y_pred1 = lm.predict(X_train)
mae_test1 = np.sum(np.absolute(y_pred1 - y_train))/len(y_train)
mae_test1
```

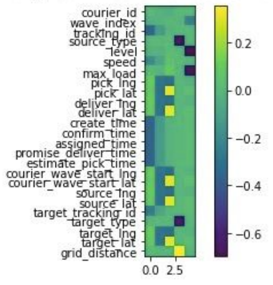
```
action_type      0.860266
expect_time    494.273048
dtype: float64
```

We then realized that it might be because all features were included as independent variables and there may be unnecessary information which is affecting the outcome. Therefore, we used the PCA and tried to find the features that matter more.

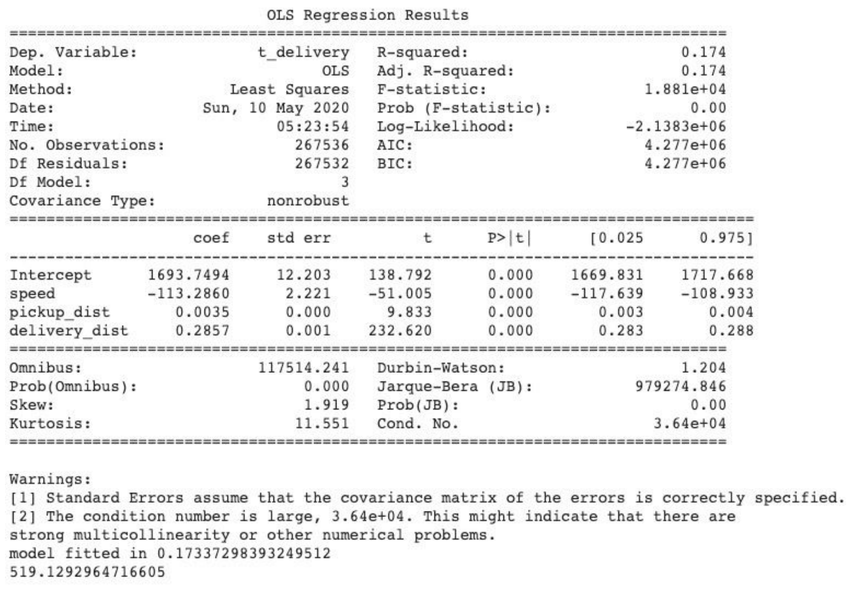


```
] from sklearn.preprocessing import StandardScaler
  import matplotlib.pyplot as plt
  from sklearn.decomposition import PCA
  from sklearn.pipeline import make_pipeline
  pca_scaled = make_pipeline(StandardScaler(), PCA(n_components = 5))
  # pca_scaled = PCA(n_components = 8)
  X_pca_train = pca_scaled.fit_transform(X_train)
  components = pca_scaled.named_steps['pca'].components_
  # components = pca_scaled.components_
  plt.imshow(components.T)
  plt.yticks(range(len(X_train.columns)), X_train.columns)
  plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x7f2584bf4198>
```



After continuing to filter out for Important variables, we went back and decided to take the initial results we found from PCA to do dimensionality reduction and solely focus on results we found as relevant. The summary of our results are shown below

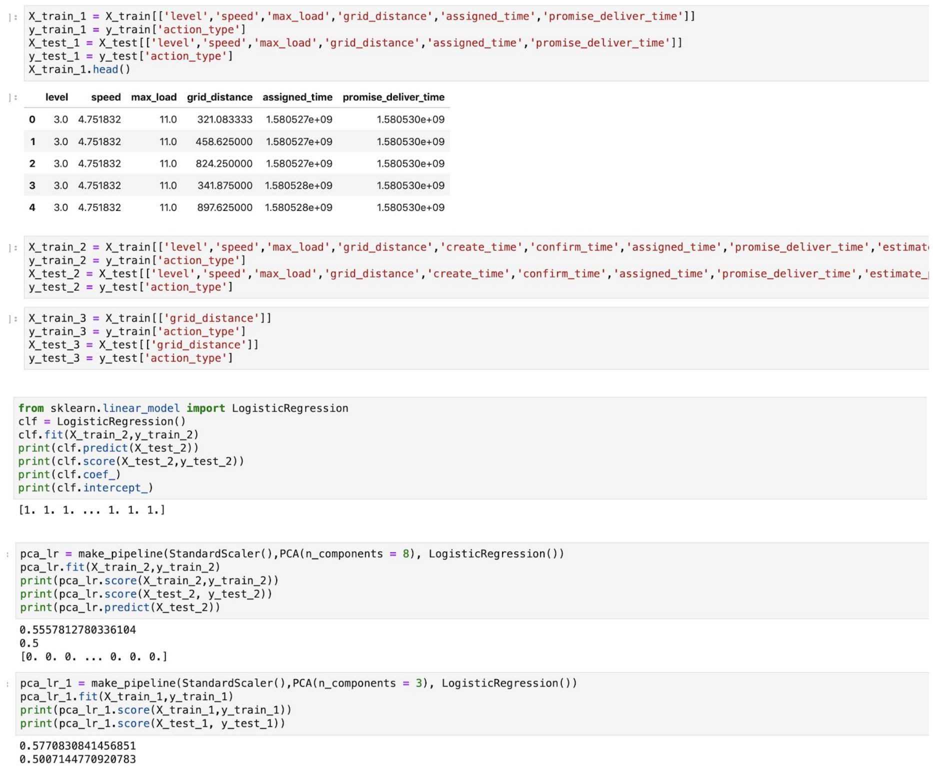


```
                            OLS Regression Results
==============================================================================
Dep. Variable:             t_delivery   R-squared:                       0.174
Model:                            OLS   Adj. R-squared:                  0.174
Method:                 Least Squares   F-statistic:                 1.881e+04
Date:                Sun, 10 May 2020   Prob (F-statistic):               0.00
Time:                        05:23:54   Log-Likelihood:            -2.1383e+06
No. Observations:              267536   AIC:                         4.277e+06
Df Residuals:                  267532   BIC:                         4.277e+06
Df Model:                           3
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept     1693.7494     12.203    138.792      0.000    1669.831    1717.668
speed         -113.2860      2.221    -51.005      0.000    -117.639    -108.933
pickup_dist      0.0035      0.000      9.833      0.000       0.003       0.004
delivery_dist    0.2857      0.001    232.620      0.000       0.283       0.288
==============================================================================
Omnibus:                   117514.241   Durbin-Watson:                   1.204
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           979274.846
Skew:                           1.919   Prob(JB):                         0.00
Kurtosis:                      11.551   Cond. No.                     3.64e+04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.64e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
model fitted in 0.17337298393249512
519.1292964716605
```

We found that having speed and distances could be pretty substantial variables to predicting the model. Even though the R-Squared is off, we suspected that the distribution has a lot of noise given that each day has many different variations. Therefore, we went forward with our analysis.

## Logistic Regression

Our group members tried to use logistic regression to predict the next action of the courier. The first time we randomly input several features. However, the result is exactly 50% correct and all of the actions were predicted to be "delivery". We tried to find the reason for that. After consulting the professor, we found that for each order, no matter the action is "delivery" or "Pickup", lots of the features are the same, such as "create time", "confirm time" and etc. are the same, because they belong to the same order. Therefore, the second time, we tried to input several features that we think might have correlation with the drivers' decision such as level, speed, max_load, grid_distance, assigned time etc. We first solely used the logistic regression, then, we tried first standardizing the data and implementing PCA to do logistic regression. However, the problem of only predicting one action type still exists. We were suggested to add some noises to the covariates by the professor. However, due to the time limitation and our findings on a better approach for prediction, we abandon this method of using the logistic regression.



```
X_train_1 = X_train[['level','speed','max_load','grid_distance','assigned_time','promise_deliver_time']]
y_train_1 = y_train['action_type']
X_test_1 = X_test[['level','speed','max_load','grid_distance','assigned_time','promise_deliver_time']]
y_test_1 = y_test['action_type']
X_train_1.head()
```

| | level | speed | max_load | grid_distance | assigned_time | promise_deliver_time |
|---|---|---|---|---|---|---|
| **0** | 3.0 | 4.751832 | 11.0 | 321.083333 | 1.580527e+09 | 1.580530e+09 |
| **1** | 3.0 | 4.751832 | 11.0 | 458.625000 | 1.580527e+09 | 1.580530e+09 |
| **2** | 3.0 | 4.751832 | 11.0 | 824.250000 | 1.580527e+09 | 1.580530e+09 |
| **3** | 3.0 | 4.751832 | 11.0 | 341.875000 | 1.580528e+09 | 1.580530e+09 |
| **4** | 3.0 | 4.751832 | 11.0 | 897.625000 | 1.580528e+09 | 1.580530e+09 |

```
X_train_2 = X_train[['level','speed','max_load','grid_distance','create_time','confirm_time','assigned_time','promise_deliver_time','estimat
y_train_2 = y_train['action_type']
X_test_2 = X_test[['level','speed','max_load','grid_distance','create_time','confirm_time','assigned_time','promise_deliver_time','estimate_
y_test_2 = y_test['action_type']
```

```
X_train_3 = X_train[['grid_distance']]
y_train_3 = y_train['action_type']
X_test_3 = X_test[['grid_distance']]
y_test_3 = y_test['action_type']
```

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression()
clf.fit(X_train_2,y_train_2)
print(clf.predict(X_test_2))
print(clf.score(X_test_2,y_test_2))
print(clf.coef_)
print(clf.intercept_)
```

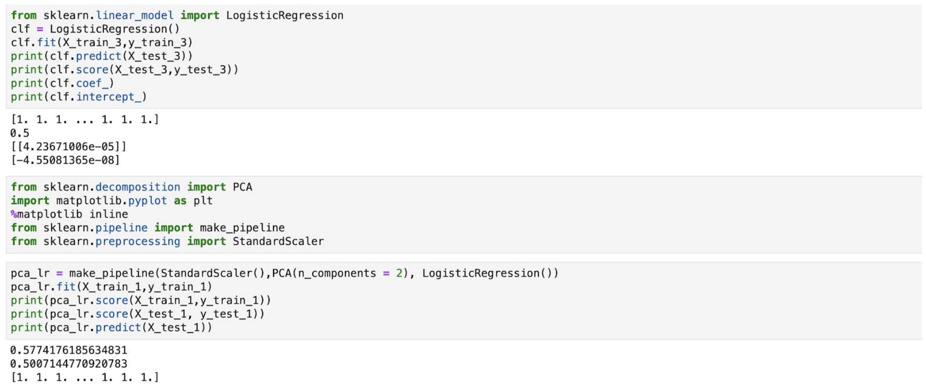
```
[1. 1. 1. ... 1. 1. 1.]
```

```
pca_lr = make_pipeline(StandardScaler(),PCA(n_components = 8), LogisticRegression())
pca_lr.fit(X_train_2,y_train_2)
print(pca_lr.score(X_train_2,y_train_2))
print(pca_lr.score(X_test_2, y_test_2))
print(pca_lr.predict(X_test_2))
```

```
0.5557812780336104
0.5
[0. 0. 0. ... 0. 0. 0.]
```

```
pca_lr_1 = make_pipeline(StandardScaler(),PCA(n_components = 3), LogisticRegression())
pca_lr_1.fit(X_train_1,y_train_1)
print(pca_lr_1.score(X_train_1,y_train_1))
print(pca_lr_1.score(X_test_1, y_test_1))
```

```
0.5770830841456851
0.5007144770920783
```



```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression()
clf.fit(X_train_3,y_train_3)
print(clf.predict(X_test_3))
print(clf.score(X_test_3,y_test_3))
print(clf.coef_)
print(clf.intercept_)
```

```
[1. 1. 1. ... 1. 1. 1.]
0.5
[[4.23671006e-05]]
[-4.55081365e-08]
```

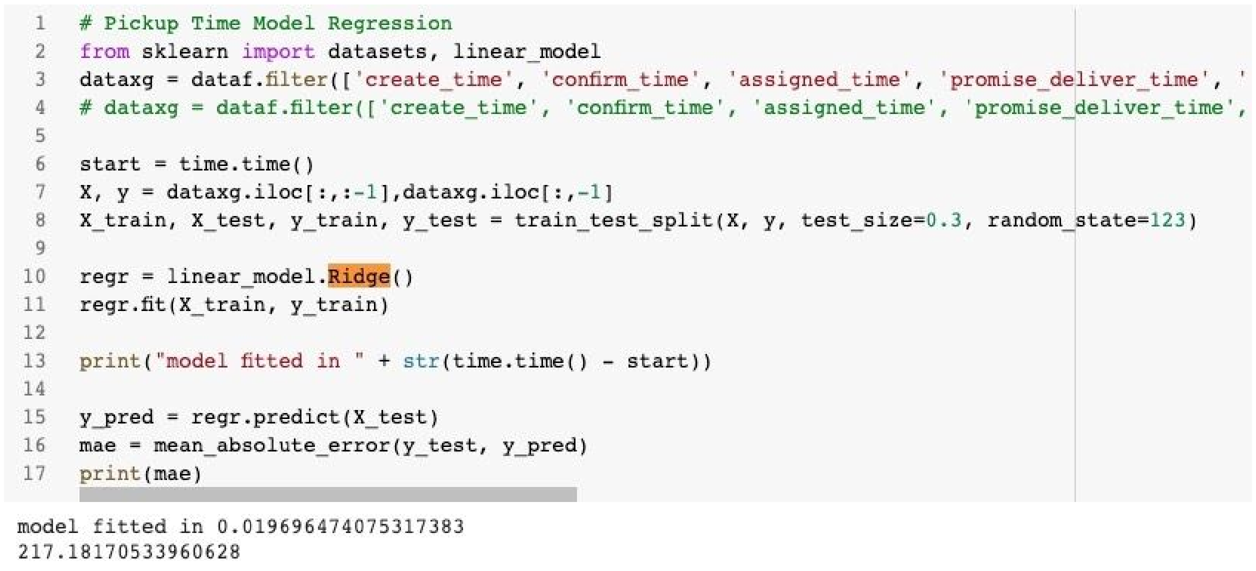
```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
```

```
pca_lr = make_pipeline(StandardScaler(),PCA(n_components = 2), LogisticRegression())
pca_lr.fit(X_train_1,y_train_1)
print(pca_lr.score(X_train_1,y_train_1))
print(pca_lr.score(X_test_1, y_test_1))
print(pca_lr.predict(X_test_1))
```

```
0.5774176185634831
0.5007144770920783
[1. 1. 1. ... 1. 1. 1.]
```

## Ridge Regression

Ridge Regression penalizes the L2 norm of the coefficients. This ensures that unimportant and or multicollinear variables are assigned a coefficient close to 0. Given that ridge regression is a more robust version of OLS, we expected a better test mean average error. However, the mean average error of ridge regression vs OLS is negligible. Therefore, we decided not to proceed further with ridge regression.



```
# Pickup Time Model Regression
from sklearn import datasets, linear_model
dataxg = dataf.filter(['create_time', 'confirm_time', 'assigned_time', 'promise_deliver_time', '
# dataxg = dataf.filter(['create_time', 'confirm_time', 'assigned_time', 'promise_deliver_time',

start = time.time()
X, y = dataxg.iloc[:,:-1],dataxg.iloc[:,-1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=123)

regr = linear_model.Ridge()
regr.fit(X_train, y_train)

print("model fitted in " + str(time.time() - start))

y_pred = regr.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
print(mae)
```

```
model fitted in 0.019696474075317383
217.18170533960628
```

## Other Methods

After our preliminary way we did analysis, we wanted to be a bit more creative and devised extended ways we could see how to make a prediction model. Given that the results from all regression were not giving us good MAE scores, we decided to be more creative and use the results from Regression, showing the importance of "time", "distance", and "speed" as our main variables with our next methodologies.

## XGBoost

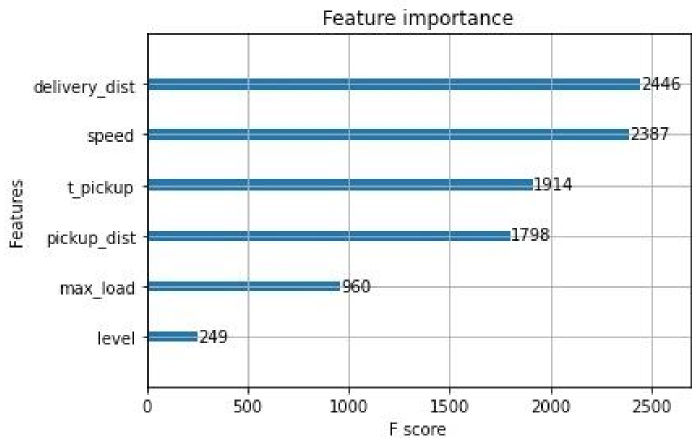
Since our data contains many numerical and categorical variables, we decided to use a gradient boosting algorithm, XGBoost, as a model. XGBoost is a machine learning library initially created by Tianqi Chen. Since then, the library has received numerous rave reviews and won multiple Kaggle competitions. Our hypothesis is that we can rely on the gradient boosting regression algorithm to learn a model that can give us the best performance.

We used all numerical explanatory variables, including one-hot-encoding of weather, as our explanatory variables. We then fit a XGBoost model using least squares loss. Since we want to fine-tune hyperparameters such as n, the number of predictors, d, the maximum depth of a tree, and p, percentage of data to sample for each predictor. We then performed grid search on the hyperparameters using k-fold cross validation. We arrived at optimal parameters, n=500, d=5, and p=.4.

We were able to achieve a test mean average error of 417 when predicting both the pickup time and delivery time. The error is reduced to 235 when we know one of pickup time or delivery time and are predicting the other time. However, our public leaderboard results are a lot worse than our top-down and bottom-up approaches, so we did not use XGBoost in our final model.

## Feature Importance

With our predicted model, we plotted the top 6 important features. Delivery distance and speed were the most important features. This makes sense as our "winning formula" model uses only delivery distance and speed for most of our predictions. Furthermore, this makes sense intuitively as the longer a delivery is, the longer the time should be.



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)
xg_reg = xgb.XGBRegressor(objective ='reg:squarederror', colsample_bytree = 0.3, learning_rate = 0.1,
                max_depth = 5, alpha = 10, n_estimators = 50)
```

## “Winning Formula”

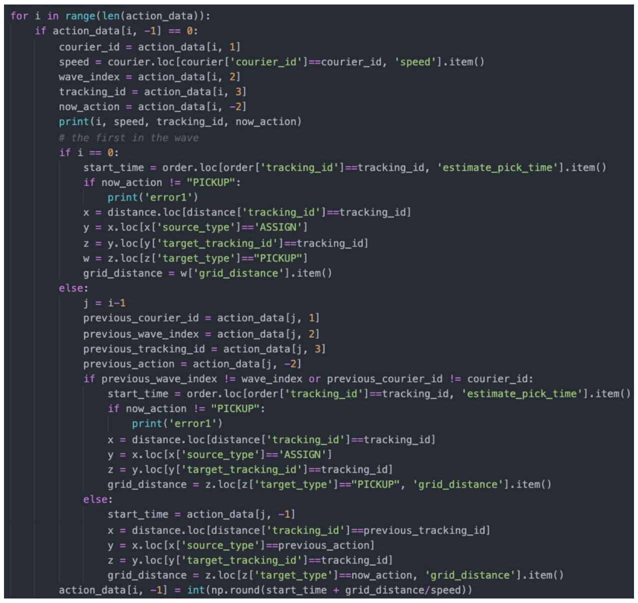
We noticed that 8910 entries were 0's (39.8%) and 13475 entries were given (60.2%). Therefore, we believe using given expect_time as known knowledge will increase our prediction accuracy significantly. Our winning formula was based on a very intuitive baseline (here we call it the Top-Down Approach). Our first attempt at the baseline gave us a surprisingly good score of 150 on Kaggle. After that, we continued to perfect the baseline model and tried a reverse baseline (here we call it Bottom-Up Approach). Although the reverse baseline did not perform as well as the original baseline, we analyzed the reasons and combined the two to obtain our winning model (here we call it the Cocktail Method).

### Top-Down Approach

The Top-Down Approach is our intuitive baseline that scored 150 at first try. In short, we calculated the expect_time of one row using the expect_time from the row above. In cases where the expect_time of the row above is given or predicted already, we simply predict the expect_time by adding the time of action. The time of action is estimated by using grid_distance/speed. We obtained the grid_distance by looking up the Distance datasheet as a dictionary. As long as we know the tracking_id, the target_id and the two consecutive actions, we are able to “read” the grid_distance accordingly. As for the speed, since the speed of couriers changes every day, we believe that speed itself is correlated with multiple factors, such as weather condition and so on. More complicated cases are those where the first expect_time of a wave is not given, which means the following rows will not have anything to base on. To tackle this problem, we used the estimate_pick_time to get the first expect_time of a wave.

However, the estimate_pick_time is not very accurate. According to the data of Feb 1st, only 53% of the pickups are actually done in time. So, we tweaked the data of estimate_pick_time to fine_tune our model and managed to get better results.

Since we are going from the first row to the last, we call this approach the Top-Down Approach.



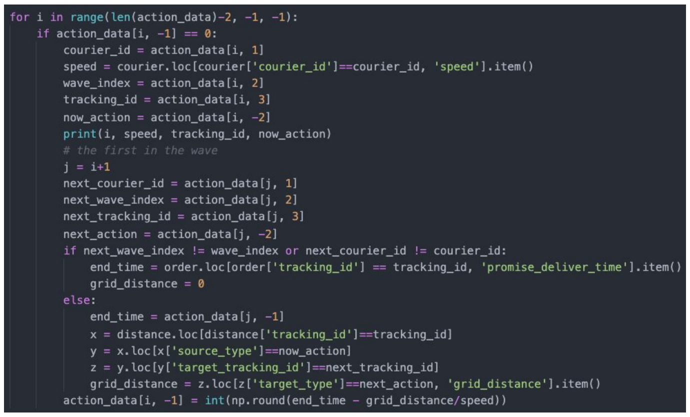
```python
for i in range(len(action_data)):
    if action_data[i, -1] == 0:
        courier_id = action_data[i, 1]
        speed = courier.loc[courier['courier_id']==courier_id, 'speed'].item()
        wave_index = action_data[i, 2]
        tracking_id = action_data[i, 3]
        now_action = action_data[i, -2]
        print(i, speed, tracking_id, now_action)
        # the first in the wave
        if i == 0:
            start_time = order.loc[order['tracking_id']==tracking_id, 'estimate_pick_time'].item()
            if now_action != "PICKUP":
                print('error1')
            x = distance.loc[distance['tracking_id']==tracking_id]
            y = x.loc[x['source_type']=='ASSIGN']
            z = y.loc[y['target_tracking_id']==tracking_id]
            w = z.loc[z['target_type']=="PICKUP"]
            grid_distance = w['grid_distance'].item()
        else:
            j = i-1
            previous_courier_id = action_data[j, 1]
            previous_wave_index = action_data[j, 2]
            previous_tracking_id = action_data[j, 3]
            previous_action = action_data[j, -2]
            if previous_wave_index != wave_index or previous_courier_id != courier_id:
                start_time = order.loc[order['tracking_id']==tracking_id, 'estimate_pick_time'].item()
                if now_action != "PICKUP":
                    print('error1')
                x = distance.loc[distance['tracking_id']==tracking_id]
                y = x.loc[x['source_type']=='ASSIGN']
                z = y.loc[y['target_tracking_id']==tracking_id]
                grid_distance = z.loc[z['target_type']=="PICKUP", 'grid_distance'].item()
            else:
                start_time = action_data[j, -1]
                x = distance.loc[distance['tracking_id']==previous_tracking_id]
                y = x.loc[x['source_type']==previous_action]
                z = y.loc[y['target_tracking_id']==tracking_id]
                grid_distance = z.loc[z['target_type']==now_action, 'grid_distance'].item()
        action_data[i, -1] = int(np.round(start_time + grid_distance/speed))
```

## Bottom-Up Approach

The Bottom-Up Approach, or the reverse baseline as we also call it, has a similar but reversed logic compared to the Top-Down Approach. We scored a 170 for our first try. Here, we estimate the predict_time using the row below. We are subtracting the time elapsed for an action from the expect_time of the row below insteading of adding from the row above (Top-Down). We get the grid_distance and speed the exact same way, but for cases where the last entry of a wave is not given, we used promise_deliver_time to estimate.

Similarly, promise_deliver_time has a 99.7% accuracy but the actual delivery time can be much earlier. That is probably why the Bottom-Up Approach did worse than the Top-Down Approach as the promise_deliver_time is more inaccurate in general. Therefore, we tweaked the data for promise_deliver_time to fine-tune our model again.

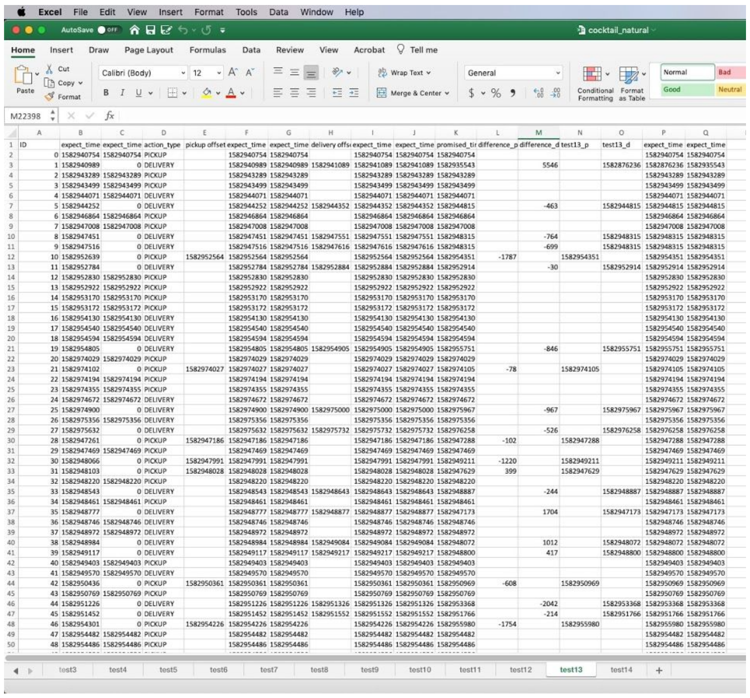
Since we are going from the last row to the first, we call this approach the Bottom-Up Approach.



```python
for i in range(len(action_data)-2, -1, -1):
    if action_data[i, -1] == 0:
        courier_id = action_data[i, 1]
        speed = courier.loc[courier['courier_id']==courier_id, 'speed'].item()
        wave_index = action_data[i, 2]
        tracking_id = action_data[i, 3]
        now_action = action_data[i, -2]
        print(i, speed, tracking_id, now_action)
        # the first in the wave
        j = i+1
        next_courier_id = action_data[j, 1]
        next_wave_index = action_data[j, 2]
        next_tracking_id = action_data[j, 3]
        next_action = action_data[j, -2]
        if next_wave_index != wave_index or next_courier_id != courier_id:
            end_time = order.loc[order['tracking_id'] == tracking_id, 'promise_deliver_time'].item()
            grid_distance = 0
        else:
            end_time = action_data[j, -1]
            x = distance.loc[distance['tracking_id']==tracking_id]
            y = x.loc[x['source_type']==now_action]
            z = y.loc[y['target_tracking_id']==next_tracking_id]
            grid_distance = z.loc[z['target_type']==next_action, 'grid_distance'].item()
        action_data[i, -1] = int(np.round(end_time - grid_distance/speed))
```

## Cocktail Method

The Cocktail Method is a way to combine the Top-Down Approach with the Bottom-Up Approach. After realizing that the skewed factors that showed Pickup and Delivery showed a different distribution than we initially hypothesized (which is covered in the "Contextualized Realizations" section), we knew that the methodologies of Top-Down emphasized a more likely prediction of Delivery time and Bottom-Up emphasized a more likely prediction of Pickup time (given that the skewness pushed Pickups skewed more early and Delivery to be later). With this knowledge and a blank excel sheet opened, we then began to transform the results and intermeshing the results together where the results of Top Down were filtered for only Delivery and the results of Bottom Up were filtered for only Pickup. When we initially submitted, the results were better than just solely the Top-Down or Bottom-Up Approach, and we felt we were going in the right track.


Excel File Edit View Insert Format Tools Data Window Help
cocktail_natural
Home Insert Draw Page Layout Formulas Data Review View Acrobat Tell me
M22398
test3 test4 test5 test6 test7 test8 test9 test10 test11 test12 test13 test14

| ID | expect_time | expect_time | action_type | pickup offset | expect_time | expect_time | delivery offs | expect_time | expect_time | promised_tir | difference_p | difference_d | test13_p | test13_d | expect_time | expect_time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1582940754 | 1582940754 | PICKUP | | 1582940754 | 1582940754 | | 1582940754 | 1582940754 | 1582940754 | | | | | 1582940754 | 1582940754 |
| 1 | 1582940989 | 0 | DELIVERY | | 1582940989 | 1582940989 | 1582941089 | 1582941089 | 1582941089 | 1582935543 | | 5546 | | 1582876236 | 1582876236 | 1582935543 |
| 2 | 1582943289 | 1582943289 | PICKUP | | 1582943289 | 1582943289 | | 1582943289 | 1582943289 | 1582943289 | | | | | 1582943289 | 1582943289 |
| 3 | 1582943499 | 1582943499 | PICKUP | | 1582943499 | 1582943499 | | 1582943499 | 1582943499 | 1582943499 | | | | | 1582943499 | 1582943499 |
| 4 | 1582944071 | 1582944071 | DELIVERY | | 1582944071 | 1582944071 | | 1582944071 | 1582944071 | 1582944071 | | | | | 1582944071 | 1582944071 |
| 5 | 1582944252 | 0 | DELIVERY | | 1582944252 | 1582944252 | 1582944352 | 1582944352 | 1582944352 | 1582944815 | | -463 | | 1582944815 | 1582944815 | 1582944815 |
| 6 | 1582946864 | 1582946864 | PICKUP | | 1582946864 | 1582946864 | | 1582946864 | 1582946864 | 1582946864 | | | | | 1582946864 | 1582946864 |
| 7 | 1582947008 | 1582947008 | PICKUP | | 1582947008 | 1582947008 | | 1582947008 | 1582947008 | 1582947008 | | | | | 1582947008 | 1582947008 |
| 8 | 1582947451 | 0 | DELIVERY | | 1582947451 | 1582947451 | 1582947551 | 1582947551 | 1582947551 | 1582948315 | | -764 | | 1582948315 | 1582948315 | 1582948315 |
| 9 | 1582947516 | 0 | DELIVERY | | 1582947516 | 1582947516 | 1582947616 | 1582947616 | 1582947616 | 1582948315 | | -699 | | 1582948315 | 1582948315 | 1582948315 |
| 10 | 1582952639 | 0 | PICKUP | 1582952564 | 1582952564 | 1582952564 | | 1582952564 | 1582952564 | 1582954351 | -1787 | | 1582954351 | | 1582954351 | 1582954351 |
| 11 | 1582952784 | 0 | DELIVERY | | 1582952784 | 1582952784 | 1582952884 | 1582952884 | 1582952884 | 1582952914 | | -30 | | 1582952914 | 1582952914 | 1582952914 |
| 12 | 1582952830 | 1582952830 | PICKUP | | 1582952830 | 1582952830 | | 1582952830 | 1582952830 | 1582952830 | | | | | 1582952830 | 1582952830 |
| 13 | 1582952922 | 1582952922 | PICKUP | | 1582952922 | 1582952922 | | 1582952922 | 1582952922 | 1582952922 | | | | | 1582952922 | 1582952922 |
| 14 | 1582953170 | 1582953170 | PICKUP | | 1582953170 | 1582953170 | | 1582953170 | 1582953170 | 1582953170 | | | | | 1582953170 | 1582953170 |
| 15 | 1582953172 | 1582953172 | PICKUP | | 1582953172 | 1582953172 | | 1582953172 | 1582953172 | 1582953172 | | | | | 1582953172 | 1582953172 |
| 16 | 1582954130 | 1582954130 | DELIVERY | | 1582954130 | 1582954130 | | 1582954130 | 1582954130 | 1582954130 | | | | | 1582954130 | 1582954130 |
| 17 | 1582954540 | 1582954540 | DELIVERY | | 1582954540 | 1582954540 | | 1582954540 | 1582954540 | 1582954540 | | | | | 1582954540 | 1582954540 |
| 18 | 1582954594 | 1582954594 | DELIVERY | | 1582954594 | 1582954594 | | 1582954594 | 1582954594 | 1582954594 | | | | | 1582954594 | 1582954594 |
| 19 | 1582954805 | 0 | DELIVERY | | 1582954805 | 1582954805 | 1582954905 | 1582954905 | 1582954905 | 1582955751 | | -846 | | 1582955751 | 1582955751 | 1582955751 |
| 20 | 1582974029 | 1582974029 | PICKUP | | 1582974029 | 1582974029 | | 1582974029 | 1582974029 | 1582974029 | | | | | 1582974029 | 1582974029 |
| 21 | 1582974102 | 0 | PICKUP | 1582974027 | 1582974027 | 1582974027 | | 1582974027 | 1582974027 | 1582974105 | -78 | | 1582974105 | | 1582974105 | 1582974105 |
| 22 | 1582974194 | 1582974194 | PICKUP | | 1582974194 | 1582974194 | | 1582974194 | 1582974194 | 1582974194 | | | | | 1582974194 | 1582974194 |
| 23 | 1582974355 | 1582974355 | PICKUP | | 1582974355 | 1582974355 | | 1582974355 | 1582974355 | 1582974355 | | | | | 1582974355 | 1582974355 |
| 24 | 1582974672 | 1582974672 | DELIVERY | | 1582974672 | 1582974672 | | 1582974672 | 1582974672 | 1582974672 | | | | | 1582974672 | 1582974672 |
| 25 | 1582974900 | 0 | DELIVERY | | 1582974900 | 1582974900 | 1582975000 | 1582975000 | 1582975000 | 1582975967 | | -967 | | 1582975967 | 1582975967 | 1582975967 |
| 26 | 1582975356 | 1582975356 | DELIVERY | | 1582975356 | 1582975356 | | 1582975356 | 1582975356 | 1582975356 | | | | | 1582975356 | 1582975356 |
| 27 | 1582975632 | 0 | DELIVERY | | 1582975632 | 1582975632 | 1582975732 | 1582975732 | 1582975732 | 1582976258 | | -526 | | 1582976258 | 1582976258 | 1582976258 |
| 28 | 1582947261 | 0 | PICKUP | 1582947186 | 1582947186 | 1582947186 | | 1582947186 | 1582947186 | 1582947288 | -102 | | 1582947288 | | 1582947288 | 1582947288 |
| 29 | 1582947469 | 1582947469 | PICKUP | | 1582947469 | 1582947469 | | 1582947469 | 1582947469 | 1582947469 | | | | | 1582947469 | 1582947469 |
| 30 | 1582948066 | 0 | PICKUP | 1582947991 | 1582947991 | 1582947991 | | 1582947991 | 1582947991 | 1582949211 | -1220 | | 1582949211 | | 1582949211 | 1582949211 |
| 31 | 1582948103 | 0 | PICKUP | 1582948028 | 1582948028 | 1582948028 | | 1582948028 | 1582948028 | 1582947629 | 399 | | 1582947629 | | 1582947629 | 1582947629 |
| 32 | 1582948220 | 1582948220 | PICKUP | | 1582948220 | 1582948220 | | 1582948220 | 1582948220 | 1582948220 | | | | | 1582948220 | 1582948220 |
| 33 | 1582948543 | 0 | DELIVERY | | 1582948543 | 1582948543 | 1582948643 | 1582948643 | 1582948643 | 1582948887 | | -244 | | 1582948887 | 1582948887 | 1582948887 |
| 34 | 1582948461 | 1582948461 | PICKUP | | 1582948461 | 1582948461 | | 1582948461 | 1582948461 | 1582948461 | | | | | 1582948461 | 1582948461 |
| 35 | 1582948777 | 0 | DELIVERY | | 1582948777 | 1582948777 | 1582948877 | 1582948877 | 1582948877 | 1582947173 | | 1704 | | 1582947173 | 1582947173 | 1582947173 |
| 36 | 1582948746 | 1582948746 | DELIVERY | | 1582948746 | 1582948746 | | 1582948746 | 1582948746 | 1582948746 | | | | | 1582948746 | 1582948746 |
| 37 | 1582948972 | 1582948972 | DELIVERY | | 1582948972 | 1582948972 | | 1582948972 | 1582948972 | 1582948972 | | | | | 1582948972 | 1582948972 |
| 38 | 1582948984 | 0 | DELIVERY | | 1582948984 | 1582948984 | 1582949084 | 1582949084 | 1582949084 | 1582948072 | | 1012 | | 1582948072 | 1582948072 | 1582948072 |
| 39 | 1582949117 | 0 | DELIVERY | | 1582949117 | 1582949117 | 1582949217 | 1582949217 | 1582949217 | 1582948800 | | 417 | | 1582948800 | 1582948800 | 1582948800 |
| 40 | 1582949403 | 1582949403 | PICKUP | | 1582949403 | 1582949403 | | 1582949403 | 1582949403 | 1582949403 | | | | | 1582949403 | 1582949403 |
| 41 | 1582949570 | 1582949570 | DELIVERY | | 1582949570 | 1582949570 | | 1582949570 | 1582949570 | 1582949570 | | | | | 1582949570 | 1582949570 |
| 42 | 1582950436 | 0 | PICKUP | 1582950361 | 1582950361 | 1582950361 | | 1582950361 | 1582950361 | 1582950969 | -608 | | 1582950969 | | 1582950969 | 1582950969 |
| 43 | 1582950769 | 1582950769 | PICKUP | | 1582950769 | 1582950769 | | 1582950769 | 1582950769 | 1582950769 | | | | | 1582950769 | 1582950769 |
| 44 | 1582951226 | 0 | DELIVERY | | 1582951226 | 1582951226 | 1582951326 | 1582951326 | 1582951326 | 1582953368 | | -2042 | | 1582953368 | 1582953368 | 1582953368 |
| 45 | 1582951452 | 0 | DELIVERY | | 1582951452 | 1582951452 | 1582951552 | 1582951552 | 1582951552 | 1582951766 | | -214 | | 1582951766 | 1582951766 | 1582951766 |
| 46 | 1582954301 | 0 | PICKUP | 1582954226 | 1582954226 | 1582954226 | | 1582954226 | 1582954226 | 1582955980 | -1754 | | 1582955980 | | 1582955980 | 1582955980 |
| 47 | 1582954482 | 1582954482 | PICKUP | | 1582954482 | 1582954482 | | 1582954482 | 1582954482 | 1582954482 | | | | | 1582954482 | 1582954482 |
| 48 | 1582954486 | 1582954486 | PICKUP | | 1582954486 | 1582954486 | | 1582954486 | 1582954486 | 1582954486 | | | | | 1582954486 | 1582954486 |

## Contextual Realizations

### The “Timeliness” Factor

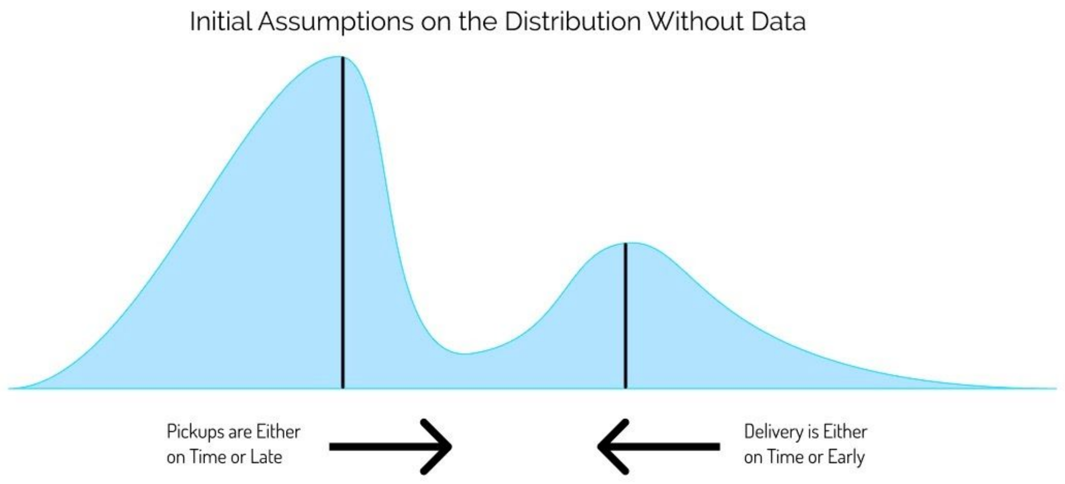
This was the first analysis that we did alongside the data solely using excel and some fanciful cell manipulation. After finding a proper way to create an outer join between the Order and Action Datasheets together, we decided we wanted to see on average the “Timeliness” Factor that Eleme Drivers took into account. Therefore, we took the first datasheets of all the different types to view the first day. We found the difference of actual times from the action datasheet to predicted pickup/delivery times showed a vast discrepancy of data, whereby there was around a 55% lateness in pickups whereas deliveries were 99% on time. This created an assumption that most days were like this, and that the “Timeliness” Factor was that Eleme drivers were willing to be late to pickups because there was possibly no penalty in doing so if they found a way to deliver on time.

### Levels

After the “Timeliness” Factor Exploration, we looked into Levels and how important they were relative to other variables. Surprisingly, through our OLS Summary Chart (not shown in this report), we found that the p-value of the Level variable showed it to be relatively insignificant compared to other variables, and this was later confirmed using PCA.  We suspect that the level is suffering from potentially an instrumental variable (such as how the actual reward system operates)  that is confounding the results. Logically, we believe that the levels would make a big difference in how an Elema driver is rated or is performing. However, we found that we would need much more time to analyze, and decided to focus on the main topic of predicting based on Kaggle's submission.

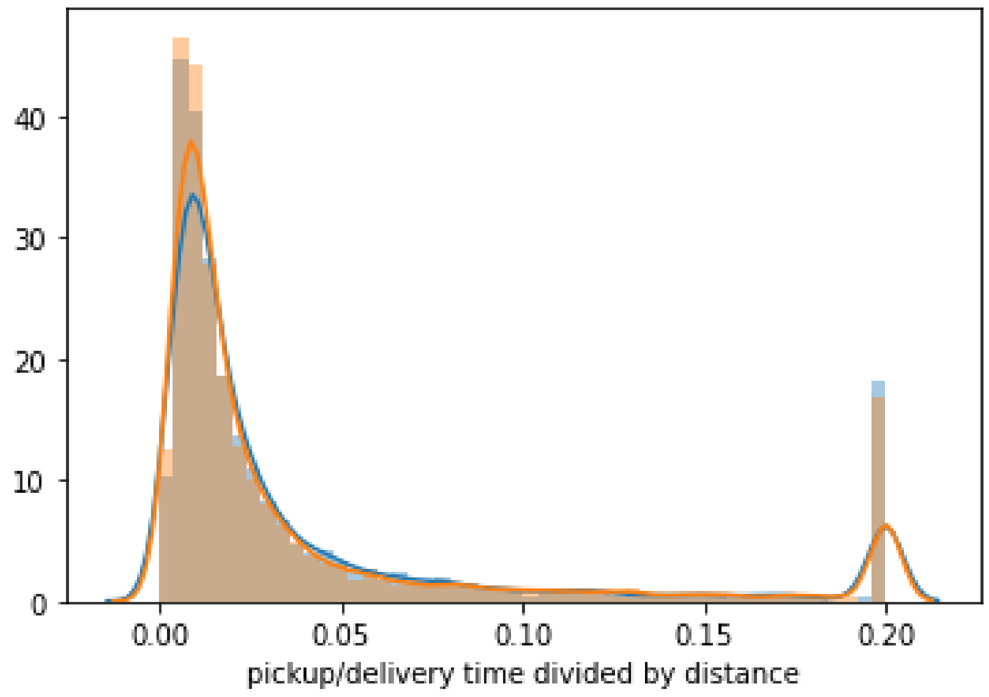
### Distributions

One of the many things we found interesting was the distribution of the data. Initially, we made very erroneous assumptions that were later fixed from viewing the distribution. One assumption we made that was wrong came from extending the “timeliness” factor to generally believe that couriers were willing to forgo their ability to pick up on time if it meant delivery could be delivered on time. After all, with one day’s worth of data, it seemed that the relative time that the delivery man decided to pick up food (if actual pickup time - expected pickup time was greater than 0, this constituted lateness) was on average later than the projected pickup time. This lead us to believe that our distribution should have looked hypothetically like this:


Initial Assumptions on the Distribution Without Data
Pickups are Either on Time or Late
Delivery is Either on Time or Early

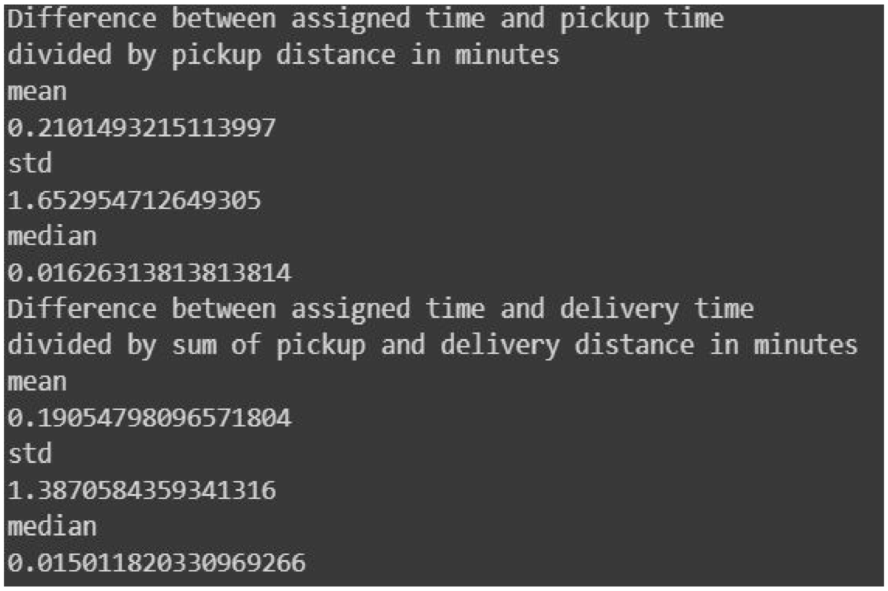
## Skewness

However, after compiling the data for all 28 days and breaking down the actual distribution of the historical data, we soon found out that our initial assumptions were wrong, and this was expressed when we mapped out the distribution in python. A big realization can when we realized the skewness was the exact opposite to what we thought it to be. (Note: the graph shows data clipped at a max of .20 distance/time. The actual distribution is not bimodal, but unimodal and highly skewed. Everything beyond .20 is represented on .20.)



## Mean and Standard Deviation/Volatility

While programming for the above mentioned graph, we wanted to ensure that our understanding of the data was not wrong, given the bias-variance tradeoff. We wanted to ensure that the variance was low such that we could depend on the graph to make our next predictions possible. Therefore, when checking for the mean and standard deviation, we found that our results were stable enough, with standard deviation being low. However, the median and the mean were completely different.

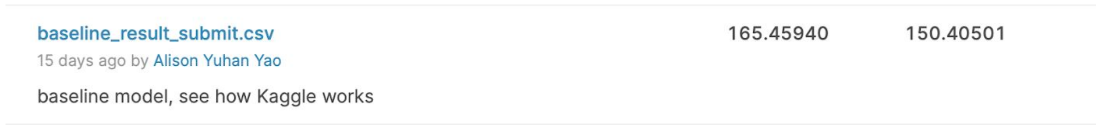


```
Difference between assigned time and pickup time
divided by pickup distance in minutes
mean
0.2101493215113997
std
1.652954712649305
median
0.01626313813813814
Difference between assigned time and delivery time
divided by sum of pickup and delivery distance in minutes
mean
0.19054798096571804
std
1.3870584359341316
median
0.015011820330969266
```

# Conclusion

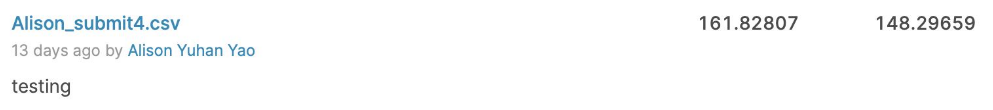
## Kaggle Submissions

As mentioned above, to our surprise, our baseline model, or the Top-Down Approach, gave us a pretty good score of 150 and helped us become the No. 1 for a couple of hours. Therefore, we kept working on modifying this model by adding to and subtracting from the estimate_pick_time.



baseline_result_submit.csv 165.45940 150.40501
15 days ago by Alison Yuhan Yao
baseline model, see how Kaggle works

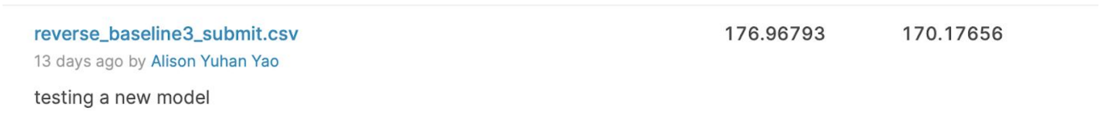
Subtracting 200 from the estimate_pick_time became the top score for the Top-Down Approach..



Alison_submit4.csv 161.82807 148.29659
13 days ago by Alison Yuhan Yao
testing

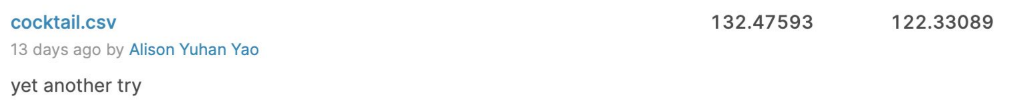
However, we understood that to achieve substantial improvement, we need to find a better model. Fine-tuning existing models can only get us this far.

Therefore, we came up with the Bottom-Up Approach, but the first try was worse than our expectation.



reverse_baseline3_submit.csv 176.96793 170.17656
13 days ago by Alison Yuhan Yao
testing a new model

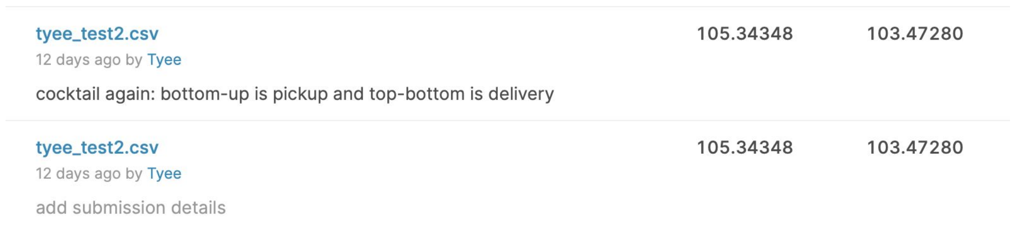
But we had no reasons to believe that the Bottom-Up Approach is flawed. As mentioned above, we suspected it was because promise_deliver_time is even more inaccurate than estimate_pick time, so we tried to combine the two models (Cocktail Method). The first try of combining the models was simply taking the average of the best prediction of the Top-Down Approach and the Bottom-Up Approach. To our surprise again, we had a much better score of 122. That is when we knew that the cocktail method is the “winning method” to go.



cocktail.csv 132.47593 122.33089
13 days ago by Alison Yuhan Yao
yet another try

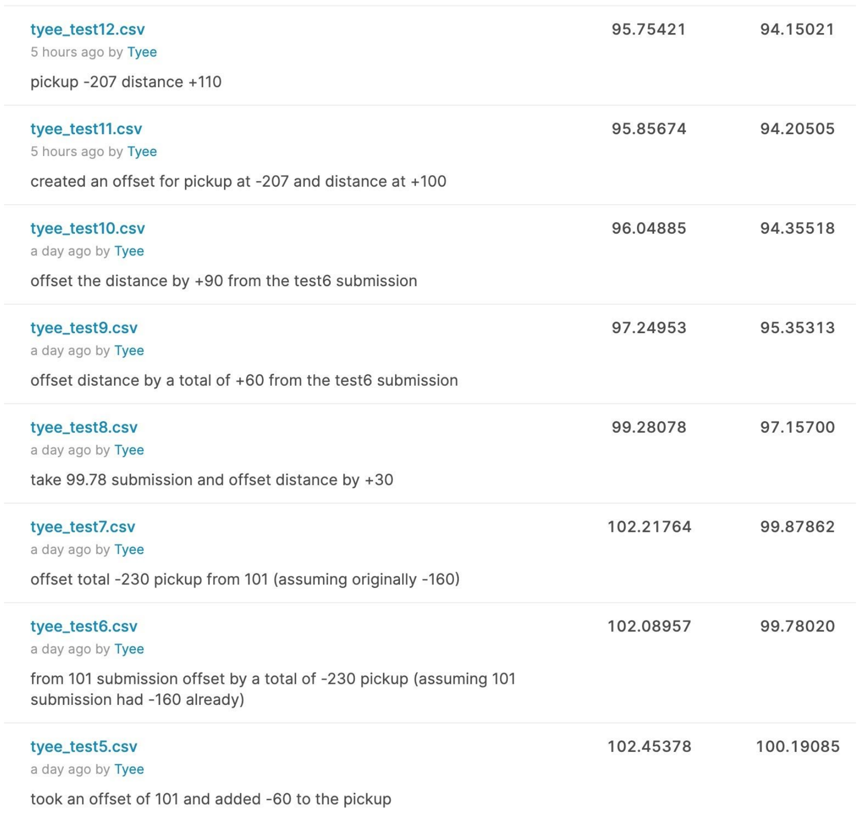
## Optimizations

We looked for smarter and more reasonable ways to combine the Top-Down Approach and the Bottom-Up Approach to formulate the Cocktail Method. Through trials and errors, we found out that the Top-Down Approach is more accurate for estimating deliveries while the Bottom-Up Approach is more accurate for estimating pickups. This model gave us a score around 103.



| | | |
|---|---|---|
| tyee_test2.csv<br>12 days ago by Tyee<br>cocktail again: bottom-up is pickup and top-bottom is delivery | 105.34348 | 103.47280 |
| tyee_test2.csv<br>12 days ago by Tyee<br>add submission details | 105.34348 | 103.47280 |

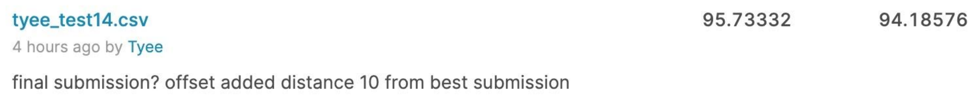
We fine-tuned the model multiple times by moving our distribution and skewing pickups rightward and deliveries leftward through offsetting the initial results by certain amounts of constants . Eventually, we managed to move beyond the 100 threshold.



| | | |
|---|---|---|
| tyee_test12.csv<br>5 hours ago by Tyee<br>pickup -207 distance +110 | 95.75421 | 94.15021 |
| tyee_test11.csv<br>5 hours ago by Tyee<br>created an offset for pickup at -207 and distance at +100 | 95.85674 | 94.20505 |
| tyee_test10.csv<br>a day ago by Tyee<br>offset the distance by +90 from the test6 submission | 96.04885 | 94.35518 |
| tyee_test9.csv<br>a day ago by Tyee<br>offset distance by a total of +60 from the test6 submission | 97.24953 | 95.35313 |
| tyee_test8.csv<br>a day ago by Tyee<br>take 99.78 submission and offset distance by +30 | 99.28078 | 97.15700 |
| tyee_test7.csv<br>a day ago by Tyee<br>offset total -230 pickup from 101 (assuming originally -160) | 102.21764 | 99.87862 |
| tyee_test6.csv<br>a day ago by Tyee<br>from 101 submission offset by a total of -230 pickup (assuming 101 submission had -160 already) | 102.08957 | 99.78020 |
| tyee_test5.csv<br>a day ago by Tyee<br>took an offset of 101 and added -60 to the pickup | 102.45378 | 100.19085 |

## Finale

The final submission continued our efforts to fine-tune the model and continues to have a relatively good score.



tyee_test14.csv 95.73332 94.18576
4 hours ago by Tyee
final submission? offset added distance 10 from best submission

## Key Takeaways

Each group member came away with a newfound way to appreciate how data in conjunction with business works in our everyday lives. In this section, we would like to devote testimonials and final notes we as individuals took away from the project

### Thomas Yee

I found working with the team to produce what we did really informative. It is rather mind blowing that with the "winning formula" it effectively represents that we were on average 94 seconds off from the actual time of an Eleme driver's action status. I would have never thought that the results could be simplified using only a few variables if it wasn't for my team doing many forms of analysis alongside me to see what types of features were important or not. I also loved having our weekly talks on our continued analysis on the data, and felt comforted by how expansionary this project really was. The professor was also very receptive to answering my questions, and just like real life, if no one knew answers, we had to go out and explore! It was overall a very fun experience and I found that my key takeaway was that Business Analytics is extremely powerful because it answers the what, why, and hows of data.

### Yuhan Yao (Alison)

Business Analytics is my very first Business class at NYU Shanghai and this is my very first time doing a Kaggle competition. Doing this project with my supporting and hardworking teammates has been a wonderful time. What I have learned is that sometimes we just need to trust our intuition. We tried many machine learning models but none of them did as well as our super intuitive baseline model. It was a gut-feeling that grid_distance and speed, together with the given expected_time are the most important components in our formula. And look at how our baseline model has taken us this far. It is amazing to look back and see how everything started as a simple idea. We asked so many questions and went to OH for many many weeks. Understanding data is the most important. .I think that is why Business Analytics is different from Data Science. Business acumen is cultivated through research, through experience and most importantly, through time.

## Mengjie Shen (Jennie)

It's really inspiring and fun as well to use what we've learnt in class to explore the real life problem. What I enjoyed the most in this process was that besides predicting the result (expected time), we had a lot of unexpected findings when we tried to understand the data and explore the model. For example, it's really mind-blowing when my team-mate found that sometimes some orders were assigned after he picked up the previous order, but other times, several orders might be assigned at the same time. I gradually understood that the difference between the Business Analytics and other data science courses is that it requires more understanding towards the data instead of just simply running the algorithm. Anyway I really enjoyed this challenging but interesting project!

## Ling'er Zhang (Lynn)

For me this project is really challenging initially since I had no background of coding and data analysis. However, all of my group members are so supportive and friendly that they gave me lots of advice and suggestions for working with large data. Though I might not be good at using codes to analyze the data, I could use Excel to find out the inner relationship from a smaller scale and could support the team in the aspect of understanding the data.

At first I thought business analytics highlights the business side more and was shocked after a few lectures that so many data science skills are needed in this class. However, throughout this semester and this Elema delivery case study, I found out that it is impossible to separate the coding and business parts: we need the business part to understand the background as well as the inner relationship of the data & we use the coding skill to analyze the data and apply them to business. Therefore, after this semester, for me business analytics is the updated version of business in the fast developing world!

## Xue Bai (Cheryl)

What I have learned from both this project and the course is the difference between business analytics and pure data science. At the beginning of the project, I was only looking at the relationship between given data and tried to come up with better models to do the prediction. However, it didn't go well. After having meetings with my team members and hearing their thoughts of the project, I realized that it is more important to look at the task from reality and collect necessary information by ourselves to better reflect the real world situation. I was impressed by their suggestions to interview the couriers and ask what they are thinking when making decisions. It thus occurred to me that however good the algorithm or model will be, the analytics should always be based on our understanding of the information.

## Future Considerations

Obviously, with the programmer's paradox, we realize that this project was a race against time and perfection. In a year's time, if we were to look back and see what we accomplished here today, we would obviously know better methods, more creative solutions, and cleaner ways to accomplish much better results. This knowledge will only come from time and experience after working with data and trying out new things. However, this entire project was a great way to jumpstart our journey into business analytics.. Leaving this report open ended is a great segway into how we could come back to this problem with fresh eyes and approach the problem in a different way later on as we grow along. Therefore, some future considerations we still do not have answers to would be great to explore someday.

### Future Questions

- A More In-Depth Analysis of the Reward System - how much are Eleme driver's paid? How does this drive their motivation to choose a specific order? How does Eleme decide the payment amounts?
- What are the Endogenous Variables of Grid Distance - how did Eleme use Longitude and Latitude? Does the Grid Distance really represent the routes that Eleme Drivers are using?
- What are the technical specifications of the Bikes Eleme Drivers use? What is the type and age of bikes used and how does this affect speed?
- Do variables like duration and when Eleme drivers took breaks affect the results and separation of waves?
- Do the types and genre of foods ordered matter (example: liquids might need to be driven more slowly to not make a mess)?
- How does the Alibaba ARK system AI work? What sort of Deep Learning is being used?
- Does the weight of the parcels affect the speed and max load?
- How were promised pickup time and promised delivery time calculated?
- Would traffic/road+sidewalk conditions per day based on a heat map of long/lat coordinates greatly affect the "timeliness" factor?
- What is the usual age an Eleme driver starts working? Does having experience and knowledge of roads make them "better" drivers with higher levels?
- What are the updates on the consumer side and how do they affect the Driver's decisions(ex: maybe stress and calling the Eleme driver means faster driving)?

- Do Eleme Drivers add in the time needed to get customer confirmation to get food (sometimes Eleme drivers have to drive back because no answer)?

- Are there any instances where the pickup location is late in making the food, affecting the driver's profits and therefore his/her decisions?

- What is the courier's health (healthier and more alert means faster maybe)?

- What is the corresponding city Eleme driver is in and what is the popularity of Eleme there over meituan? What does having competitors do to Eleme Driver's demand and amount of assignments?

- How does Eleme's system of shop availability affect the distances Eleme drivers need to cover (ex: if the shop is closed, then would there be fewer pickup points on the map)?

- Do customer satisfaction levels go into dictating level? Or is solely the system's rating of "timeliness"?

## Future Analysis

- Study the effect of seasonality and time series analysis - from other presentations we found that the 14 day was an outlier to the rest because it was valentines day
- more correlations given answers to the "future questions"
- Using one hot encoding of new categorical variables to broaden the scope of XGBoost and have more accuracy
- Search for instrumental variables in customer data
- Adding the Reward System and relevant variables into linear regression for more rooted prediction
- Analysis using Dynamic Programming of what is the most appropriate route an Eleme driver should take from point A to point B (having new assignments midway between pickup to delivery of another order)
- Causal inference on customer satisfaction in conjunction with courier data