Data Science Project

PORTFOLIO

Jing Hou

Jinghou8@gmail.com

647-968-8696

Table of content

Uber Ridership Data in New York City

1. Introduction

Uber is affecting the way people move in cities. While have had a big impact on traditional taxi services. The popular ride sharing service Uber has undoubtedly affected the taxi industry by offering lower prices, faster and more quality service, as well as a higher degree of transparency in terms of choosing drivers and determining fares. It also helps reduce the number of private cars on road and improve the efficiency of public transportation.

However, there are times when customers are left unsatisfied because of shortage of vehicles which ultimately led to Uber adopting surge pricing. it is important to get information from historical trip data to estimate the order demands to make sure there is no shortage of Uber drivers in specific area during specific time and weather. If the suddenly appeared demands for Uber cars are not satisfied, then customers' experience is likely to be ruined and the churn rate could rise.

The analysis is utilizing the existing resources and could provide the guidance for marketing teams to take campaign strategies and attract more drivers to reach supply-demand balance in specific areas.

2. Objective

The purpose of the project is to analyze the ridership and effect of weather on ridership. Since many customer voices their concern about Uber adopting surge pricing to deal with the increasing demand.

3. Data collection & wrangling

Raw data set:

1) Four CSV file with the monthly Uber pick-up records in New York City from April to July 2014
2) One Excel file with the weather information for these four months

A	B	C	D
Date/Time	Lat	Lon	Base
4/1/2014 0:11:00	40.769	-73.9549	B02512
4/1/2014 0:17:00	40.7267	-74.0345	B02512
4/1/2014 0:21:00	40.7316	-73.9873	B02512
4/1/2014 0:28:00	40.7588	-73.9776	B02512
4/1/2014 0:33:00	40.7594	-73.9722	B02512
4/1/2014 0:33:00	40.7383	-74.0403	B02512
4/1/2014 0:39:00	40.7223	-73.9887	B02512

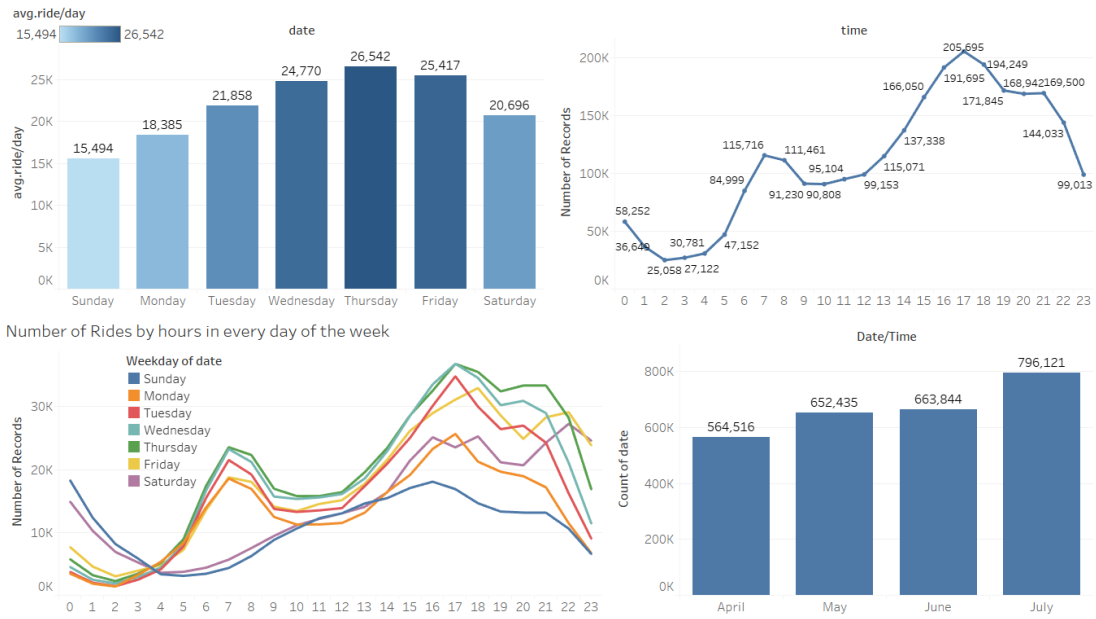
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
2014	Temp. (°C)		Dew Point (°C)		Humidity (%)		Sea Level Press. (hPa)		Visibility (km)		Wind (km/h)		Precip. (mm)		Events						
Apr	high	avg	low	high	avg	low	high	avg	low	high	avg	low	high	avg	low	high	avg	high	sum	
1	14	8	3	2	-2	-4	76	56	35	1021	1019	1016	16	16	16	26	16	40	0	
2	9	6	3	5	3	1	92	81	70	1022	1020	1019	16	16	16	16	6	32	0.76	Rain
3	17	11	4	5	1	-6	92	57	22	1021	1019	1017	16	16	13	21	9	26	2.03	Rain

First, we use pandas to load the local data file and the above figures are the raw data. It contains the pickup data, time and locations (longitude and latitude). Base is the TLC base company code affiliated with the Uber pickup. The second dataset is the weather data in New York City from April to June, which is messy.

During the data wrangling process, I union the four CSV file with the monthly Uber pick-up records and modified the date/time column for the two datasets. I merge the two table by date and create new columns to indicate if there is rain or fog or thunderstorm or snow on that day or if it is a normal sunny day. The following figure shows what does modified table look like.

	Date/Time	Lat	Lon	Base	date	time	Month	Events	Normal	Rain	Fog	Thunderstorm	Snow
0	2014-04-01 00:11:00	40.7690	-73.9549	B02512	2014-04-01	00:11:00	2014-04-01	Normal	True	False	False	False	False
1	2014-04-01 00:17:00	40.7267	-74.0345	B02512	2014-04-01	00:17:00	2014-04-01	Normal	True	False	False	False	False
2	2014-04-01 00:21:00	40.7316	-73.9873	B02512	2014-04-01	00:21:00	2014-04-01	Normal	True	False	False	False	False
3	2014-04-01 00:28:00	40.7588	-73.9776	B02512	2014-04-01	00:28:00	2014-04-01	Normal	True	False	False	False	False

4. Visualization using Tableau

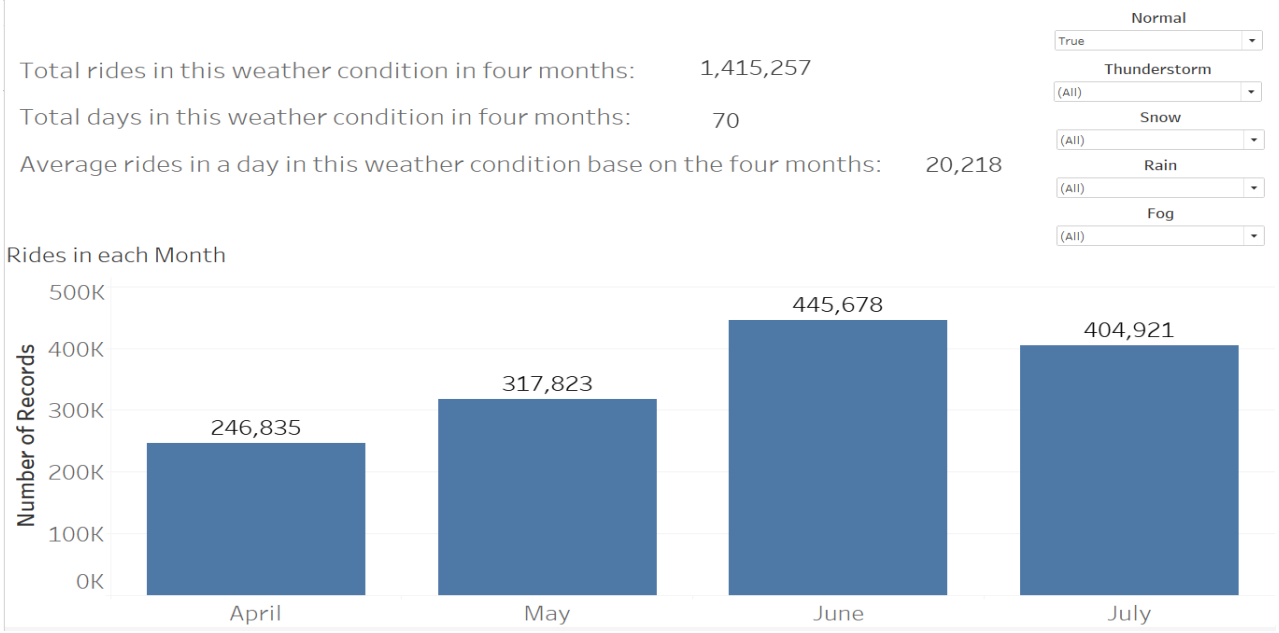


https://public.tableau.com/profile/jing.hou6869#!/vizhome/uber_1_15919871206210/Dashboard4?publish=yes

In Tableau, the Dashboard contains several views, which could demonstrate a variety of data simultaneously and interactively. The Tableau Dashboard in the current page shows how the number of orders vary among different periods of time (weekdays, months, hours). It is clearly to see that the total orders in the four month is steadily increasing. When we look at the day of the week. Wednesday, Thursday and Fridays have highest demand for rides while Sunday have the lowest. When we aggregate the data on the day level. It is obvious that three peaks appear during commuting hours and the evenings.

If we take a look at the bottom left corner on the above figure, The graph show number of rides distributed by hours in every day of the week. In the weekend, people have a different travel behavior than in the weekday.

On Saturday and Friday, the need for rides is increasing from 8pm – 10pm which is different than the other day of week. There are significantly higher demand for rides on Saturday and Sunday from midnight to 4am, and this period is usually considered as party time.

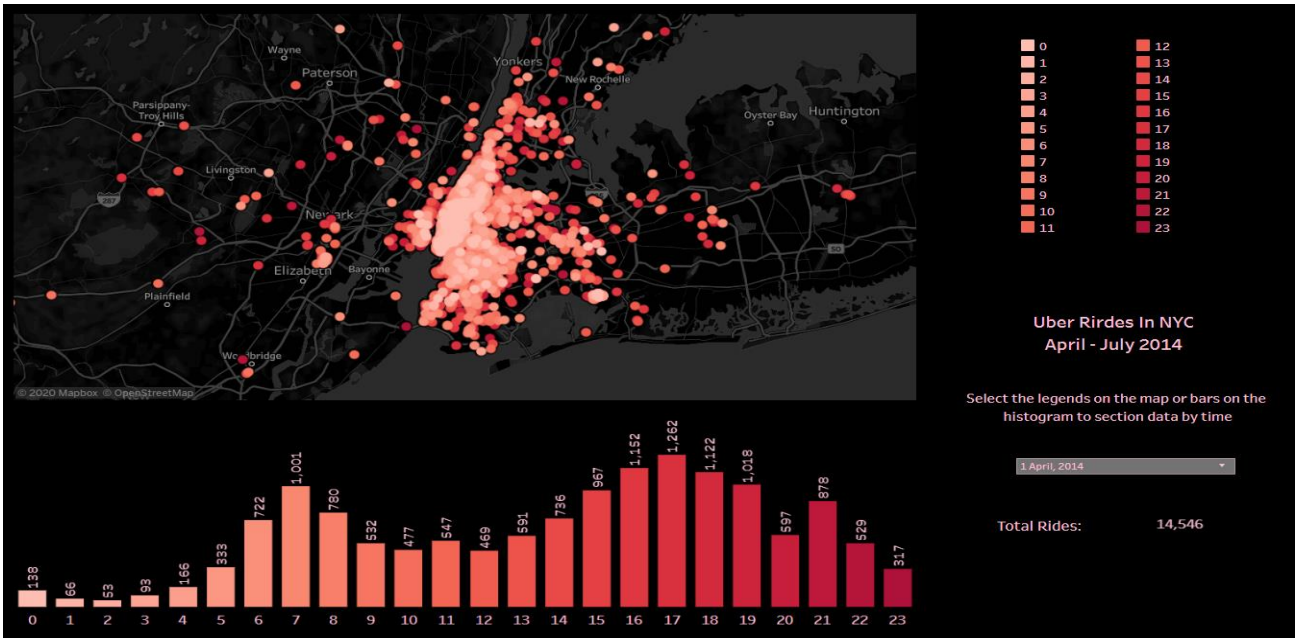


https://public.tableau.com/profile/jing.hou6869#!/vizhome/ridesindifferentweatherconditions/Dashboard1?publish=yes

Another factor that could highly affect the demand for rides is the weather condition. We confirm our assumption by setting parameters normal, thunderstorm, snow, rain and fog. The dashboard above demonstrates the difference of number of riders in different weather conditions by selecting the different weather in the top right corner. The figure above is showing the information of rides in normal/ sunny days. There are 70 normal/ sunny days occur in the four months, having 1415K total rides and around 20K rides in a day. Only 8 foggy days occur in the four months, having 179K total rides and around 22K rides in a day. 50 raining days occur in the 4 months, having 1223K total rides and around 24K rides in a day. only 2 snow days occur in the 4 months, having 38K total rides and around 19K rides in a day.13 thunderstorm days occur in the 4 months, having 348K total rides and around 27K rides in a day. The total number of trip records is about 2.7 million among four months (122 days) and around 22K rides in a day.

That is about 35% increasing by Comparing the average rides in a normal day and a thunderstorm day. 20% increasing by normal vs raining day.

The following graph shows the geographical and time distribution for four months. The dashboard illustrate most people traveled to the business area for work during the rush hours. What's more, the distribution appear if any single date is chosen, 1st April for example, this dashboard could be really helpful if the team wants to check the pickup points distribution on specific day if any important event occurred on that day.



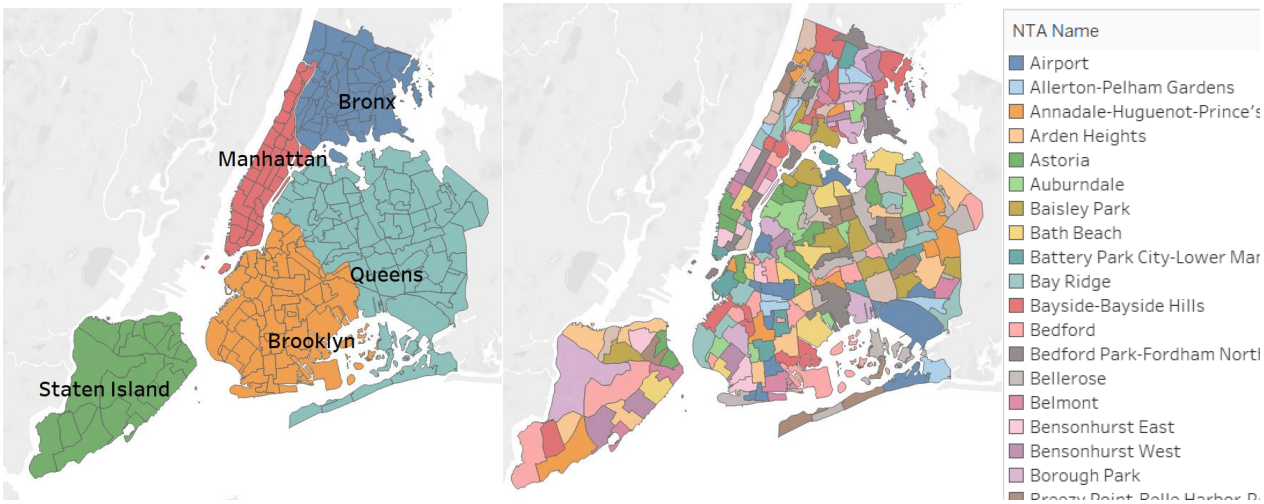
https://public.tableau.com/profile/jing.hou6869#!/vizhome/uber_1_15919871206210/1?publish=yes

The scatter plots can illustrate the geographical distribution of the pickup points. But sometimes analytics team would like to analyze the aggregation trip data in specific neighborhoods during specific periods. In addition, they would be interesting on knowing the demand and supply for the ridership in some areas. Whenever there is high demand in specific time. The Uber app can immediately offer some incentives to drivers who are driving around the area and push drivers to pick up riders to maintain the balance between supply and demand. Therefore, I aggregated data in different geographic location by connecting spatial files such as Shape-files, MapInfo tables, GepJSON files.

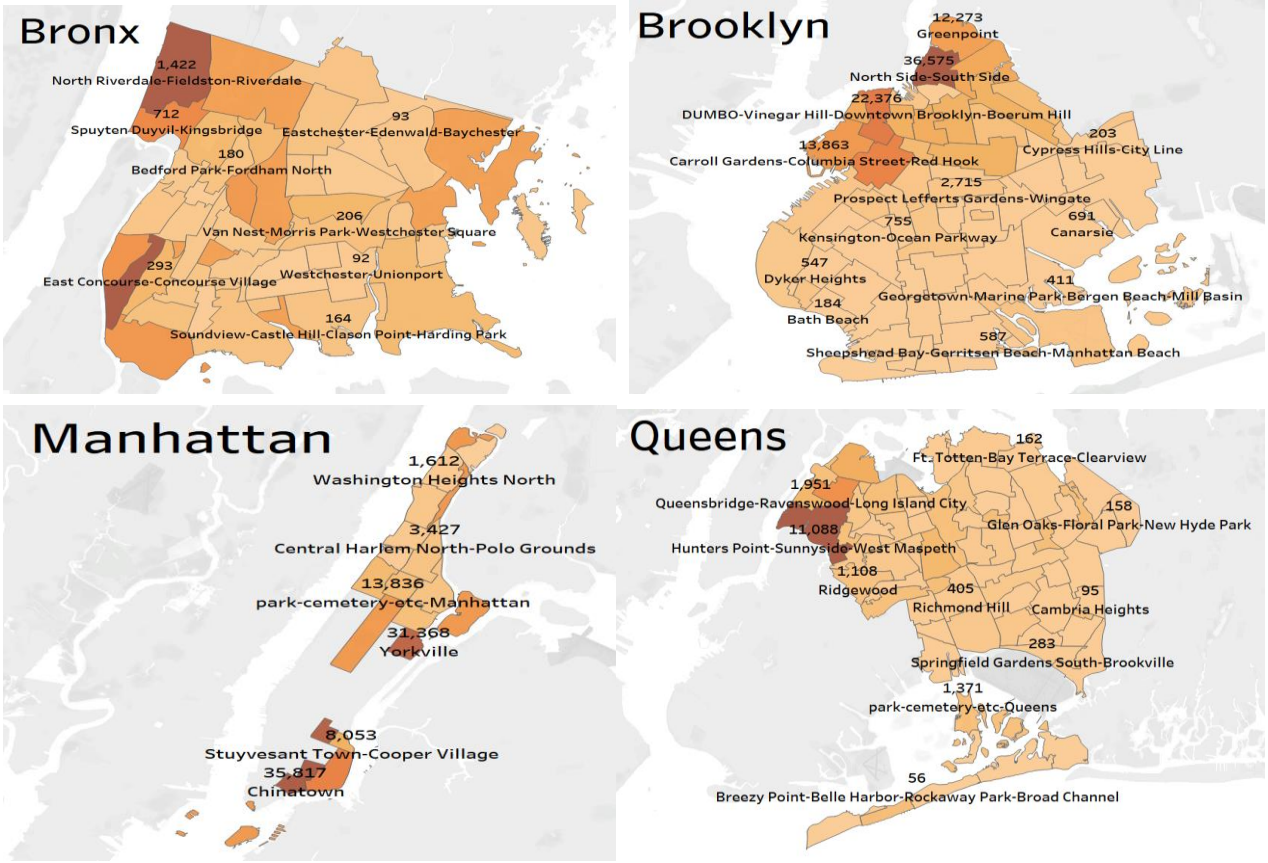
To build the ridership dashboard based on boroughs and areas, I download the New York city's Neighborhood Tabulation Areas shape-file and import it into the Tableau Desktop. The next figure shows the modified table merged with NTA shape-file on latitude and longitude

	Date/Time	Hour	NTACode	NTAName	daily-active-user	Geometry	Number of Records	date	Month	Events	Normal	Rain	Fog	Thunderstorm	Snow
0	2014-04-12 00:00:00	0	QN98	Airport	1	MultiPolygon	1	2014-04-12	2014-04-12	Normal	True	False	False	False	False
1	2014-04-12 00:07:00	0	QN98	Airport	1	MultiPolygon	1	2014-04-12	2014-04-12	Normal	True	False	False	False	False
2	2014-04-23 18:22:00	18	MN25	Battery Park City-Lower Manhattan	1	MultiPolygon	1	2014-04-23	2014-04-23	Normal	True	False	False	False	False
3	2014-04-23 18:22:00	18	MN25	Battery Park City-Lower Manhattan	1	MultiPolygon	1	2014-04-23	2014-04-23	Normal	True	False	False	False	False
4	2014-04-23 18:25:00	18	MN25	Battery Park City-Lower Manhattan	1	MultiPolygon	1	2014-04-23	2014-04-23	Normal	True	False	False	False	False

The following diagram display the shape-file by Tableau. Each borough or even smaller area is painted in different colors.

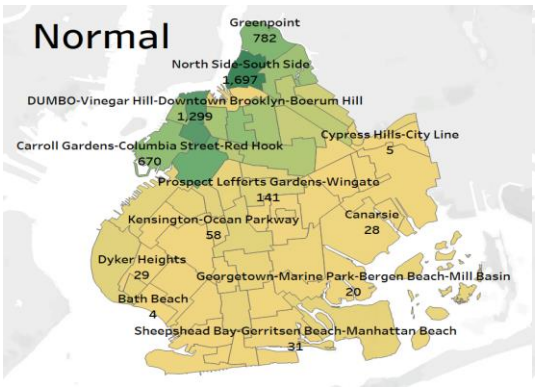


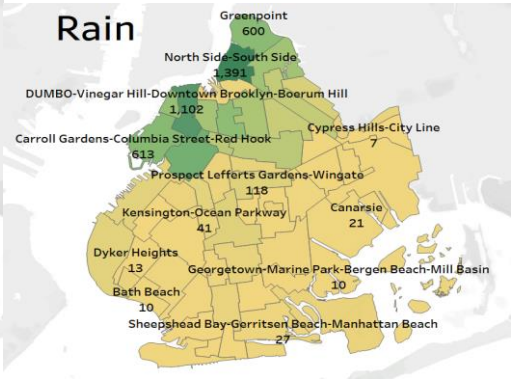
The four maps below display heat map information of total ridership in each borough, which is Bronx, Brooklyn, Manhattan and Queens. With the available options of setting specific date and time, it is very convenient to monitor how does total number of ridership change dynamically.

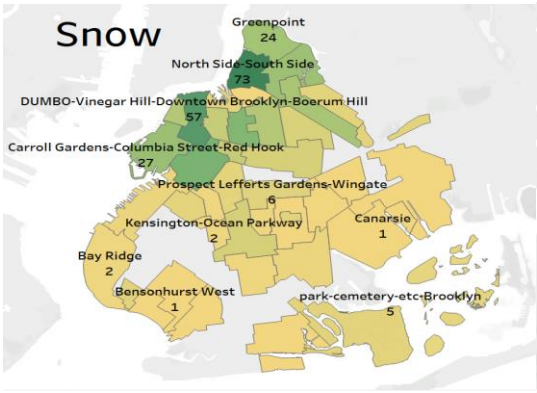


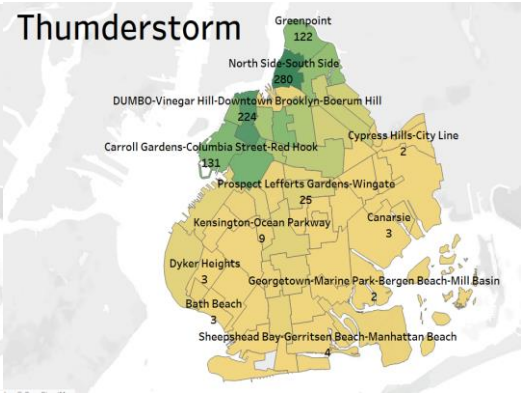
https://public.tableau.com/profile/jing.hou6869#!/vizhome/Heatmap--pickupsinboroughsindifferenthours/Dashboard1?publish=yes

These maps exhibit heat map information of number of pickups at Brooklyn on a normal/ raining/ snowing/ thunderstorm day. With the available options of setting specific time, weather and borough. So we can easily check the number of pickups in different location in NYC on different weather conditions.









https://public.tableau.com/profile/jing.hou6869#!/vizhome/Heatmap--pickupsinboroughsindifferentweather/Dashboard2?publish=yes

5. Conclusion

- ✧ Based on the data analysis, there are more rides on rainy, foggy and thunderstorm days than the normal day. Weather condition does influence Uber rides, the daily active user increase about 20% on rainy days.
- ✧ Total rides steadily increase from April to July.
- ✧ The uber rides demand in New York City reach the peak at afternoon rush hours between 5p.m to 9p.m on weekdays and in the evenings on weekends. Thursdays are the busiest days. There are always more rides in the evenings than in the mornings. 0:00am is the busiest hours in Sunday, but Sunday have the least ridership during a week
- ✧ According to the heatmap, Manhattan is the busiest and has max # of rides. Also, a high-density distribution around John F. Kennedy International Airport
- ✧ The map dashboard provides visualized pickup points across the New York City, which could help optimize the utilization of the existing resources in the future. For example, if there is any event to be hold in the future, the dynamic incentives programs could be provided to adjust the Uber drivers' distribution in advance so that the supply-demand balance could be reached to meet rider's need.
- ✧ Moreover, aggregating ridership data on a borough basis could be used to build a real-time ridership dashboard for analytics team with the help of the streaming data preprocessing techniques and Uber control center could make quick response to dynamic ridership data.

Loan Default Prediction for Bank

1. Introduction

The loan is one of the most important products of the banking. Banks loan money to companies in exchange for the promise of repayment. That means Bank only makes profit (interest) if the borrower pays off the loan. However, some people will default on the loans, being unable to repay them for some reason, then the bank loses money. To prevent this situation, banks need to find some methods to predict customers' behaviors. Machine learning algorithms have a pretty good performance on this purpose, which can predict risk by evaluating an individual's historical data.

2. Task Description

In this project, I used 8 dataset with plenty of information about the accounts of customers from a bank (dataset was collected from year of 1999) to train machine learning algorithms in order to predict customers who are more likely to default on loans. The goal of this project is to estimate different models' performance based on past behavioral data of customers and analyze the key features of default.

3. Data Description

Table	Row	Description
Account	4500	Static characteristics of an account
Client	5369	Characteristics of a client
Disposition	5369	Specify if the client is the owner of the account
Permanent order	6471	Debit-only orders such as payment of loan, household or leasing
Transaction	1056320	Each record describes one transaction on an account
Loan	682	A loan granted for a given account
Credit card	892	Credit card type: junior/classic/gold
Demographic data	77	Demographic characteristics of a district

4. Data preparations

Data are in asc files and I load data to database by using MySQL. Then we can transform and manipulate the data. Here is showing how I load data to database

```
#account
drop table if exists bank_loan.account;
create table bank_loan.account (
    account_id            int,
    district_id            int,
    frequency             varchar(20),
    date                  varchar(10)
);
truncate bank_loan.account;
-- load data into the movies table
load data local infile '../bank_loan/account.asc'
into table bank_loan.account
fields terminated by ';' ENCLOSED BY '"'
```

My main purpose is to predict customer behaviors about loan for each account. Thus, the most important table here is table “loan”. The columns “status” in table “loan” indicates whether the client (account_id) default or not. However, in the ‘’status’’ column, it divided customer’s loan status into 4 categories. In order to use binary classification, I encode the categorical variable into 1 or 0. So we count customer. The followering table illustrate how I encode transform the data.

Loan status (default or not default)	Loan status
1	B - contract finished, loan not payed
	D - running contract, client in debt
0	A - contract finished, no problems
	C – running contract, OK so far

5. Data Processing

I firstly select all useful features for the model by looking at the relativity with our target and some business sense. There are some columns describing which bank is each payment sent to, it is useless for us to predict loan default, so I drop them.

I generate some new features using aggregation functions to summarize transaction records for each account ‘s user. We gain a lot of meaningful insights by these new features. It tells us what kind of operations does clients, who is default on loan, like to do during data analysis process.

```sql
create table trans_agg
(select account_id, count(trans_id) as trans_sum, max(date)as trans_date,
        sum(case when operation='VYBER KARTOU' then 1 else 0 end)as num_credit_withdraw,
        sum(case when operation='VYBER' then 1 else 0 end)as num_withdraw_cash,
        sum(case when operation='PREVOD NA UCET' then 1 else 0 end)as num_remit_to_bank,
        sum(case when operation='VKLAD' then 1 else 0 end)as num_credit_cash,
        sum(case when operation='PREVOD Z UCTU' then 1 else 0 end)as num_collect_from_bank,
        round(avg(case when operation='VYBER KARTOU' then amount end))as amt_credit_withdraw,
        round(avg(case when operation='VYBER' then amount end))as amt_withdraw_cash,
        round(avg(case when operation='PREVOD NA UCET' then amount end))as amt_remit_to_bank,
        round(avg(case when operation='VKLAD' then amount end))as amt_credit_cash,
        round(avg(case when operation='PREVOD Z UCTU' then amount end))as amt_collect_from_bank,
        max(amount) as loan_amt
        from trans
    group by account_id
```

I am not only creating new features with MySQL but also perform feature engineering with Python. I also need Python to build and train models. Therefore, I use a Python SQL toolkit (SQLAlchemy) to connect Python to MySQL database. Then I use Pandas library to export the data as DataFrame, and preprocess the data using self-defined functions. I merge all tables together with valuable features and new features that generated.

```python
DB_HOST = 'localhost' # change to hostname of your se
DB_PORT = '3306' # change accordingly
DB_NAME = 'bank_loan' # name of your database
from sqlalchemy import create_engine
import pymysql
import pandas as pd

SQLALCHEMY_DATABASE_URI = f'{DB_TYPE}+{DB_DRIVER}://{
print(SQLALCHEMY_DATABASE_URI)

engine = create_engine(SQLALCHEMY_DATABASE_URI)
print(engine)
con1= engine.connect()

loan = pd.read_sql('''
    SELECT *
    FROM loan;
''',con=con1)
```

```python
def status (string):
    if string=="A" or string=="C":
        return '0'
    elif string=="B" or string=="D":
        return '1'
def gender (string):
    if string=="male":
        return '0'
    elif string=="female":
        return '1'
def age(d1, d2):
    d1 = datetime.strptime(d1, "%Y-%m-%d")
    d2 = datetime.strptime(d2, "%Y-%m-%d")
    return math.floor(abs((d1 - d2).days)/365)
def days_account(d1, d2):
    d1 = datetime.strptime(d1, "%Y-%m-%d")
    d2 = datetime.strptime(d2, "%Y-%m-%d")
    return abs((d1 - d2).days)
```

6. Exploratory Data Analysis (EDA)

I explore and do data analysis on the dataset to visualize, summarize and interpret the information that is hidden in rows and column format. It allows me to generate as many insights and statistical measure about the dataset as possible by performing EDA. Moreover, it also performs to define and refine our important features variable selection, that will be used in our model.

The following dashboard demonstrate the demographic comparison between default and not default. The left top chart indicates there are less defaults falls in the age groups 20-30 and 40-50 compare with other age groups. Moreover, people who under 20 have higher probability to default their loan. The pie chart shows we are dealing with an imbalanced dataset, 11% of default records and 89% of non- default.

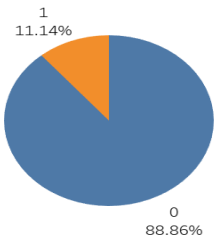
age

status	20-30	30-40	40-50	50-65	under 20
			Age on loan		
0	21.45%	24.75%	21.62%	23.10%	9.08%
1	18.42%	26.32%	19.74%	23.68%	11.84%

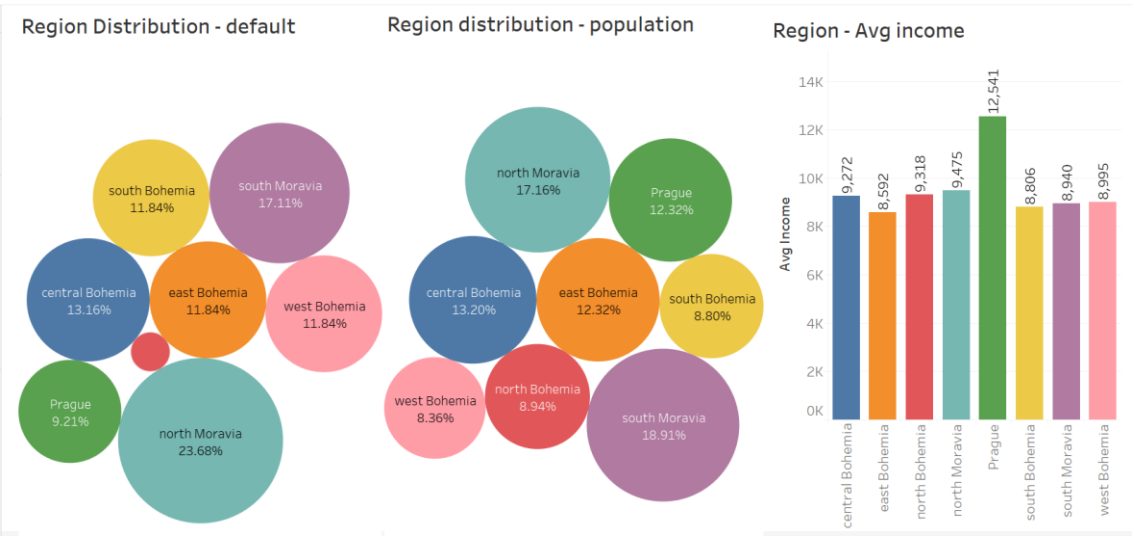
Gender

gender	status 0	1
male	49.340%	46.053%
female	50.660%	53.947%

Status



The dashboard below Illustrates geographical information. The first and second figures shows the distribution of defaults and population in regional perspective. The third diagram present the average income in each region. According to the diagram, we can see that people who live in Prague district have highest average income, which leads to the second least default cases compared with other regions. Therefore, people who living in richer area are less likely to default.



The following dashboard displays the summary of customers' average amount that dealing with their account in different activities/operations and the average frequency of using these activities. The figure also shows the comparison between default and non-default in the summary. Based on the charts, we find that account holder who are on default are tends to use a large amount of money and have less frequency on remit to bank, credit withdraw and collect from bank. However, they are more frequent on credit cash and withdraw cash, which means they may prefer to deal with cash more than people who pay loans on time. This can probably because clients who are on default have more possibilities of dealing with cash-flow crisis

Financial Operation - Amount

	status	
	1	0
collect_from_bank	27,935	32,767
credit_cash	17,928	13,695
credit_withdraw	2,944	2,642
remit_to_bank	4,713	4,585
withdraw_cash	7,280	6,965
loan_amt	51,041	43,341

Financial Operation - Times

	status	
	1	0
num_collect_from_bank	4.0	8.3
num_credit_cash	44.1	31.8
num_credit_withdraw	0.4	1.2
num_remit_to_bank	26.5	39.2
num_withdraw_cash	118.9	100.9
trans_sum	252.6	225.9

7. Avoid Data Leakage and feature engineer

To avoid data leakage, when I select features or generate new features, I put the loan issue date into considerations to decide whether to just use data right before the loan issue date. Based on the balance data, I also create the average balance of this account and ratio of balance before applying loan to loan amount as features to enhance the performance of the model.

```
trans_balance=[]
for account_id, group in trans_before_loan.groupby('account_id'):
    group.sort_values(by=['trans_id'])
    max_balance=group['balance'].max()
    avg_balance= group['balance'].mean()
    trans_count= group['trans_id'].count()
    ratio= (avg_balance/group['amount_loan']).head(1).values[0]

    balance_final=group['balance'].tail(1).values[0]
    sum_credit=group[group['type']=='1']['amount_trans'].sum()#useless

    count_credit= group[group['type']=='1'].count().tail(1).values[0]#useless
    avg_credit=sum_credit/count_credit


    trans_balance.append({'account_id': account_id,'trans_count':trans_count,'max_balance': max_balance,
                          'avg_balance':avg_balance,
                          'balance/loan': ratio, 'sum_credit':sum_credit,'balance_final':balance_final,
                         'count_credit':count_credit, 'avg_credit':avg_credit})
```

It would be better to clean all missing values in the dataset before apply ML algorithms. There are some missing values in this dataset. I would either drop them, or fill null as 0, or fill null as the average values etc.. depends on the features' function, quantity and type

I have engineered features in both MySQL and Python, create some new features to summarize the clients' transactional behavior from raw data. These features helps me to present the underlying problem to the predictive models because they provide some useful insights in EDA process.

Since our dataset is imbalanced, we can see from the pie chart in EDA session. There are only 11% of account users are on default. The model might not learn enough from the minor class and the model can be really biased. We fix this problem by using oversampling technique. It is the process of generating synthetic data that tries to randomly generate a sample of the attributes form observations in the minority class. I use RandomOverSampler() method in Imblearn package which randomly samples data points in the minority group until there are same number of points from both groups.

In order to prevent our dataset, contain some features that highly varying in magnitudes, unit and range. I perform standardization scaling method. It let many machine learning algorithms perform better or converge

faster when features are on a relatively similar scale and close to normally distributed. Standardizing can help features arrive in more digestible form.

8. Modelling and Validation

I firstly hold out the test dataset, then I use K-fold validation to split the data into train and validation dataset. I used Logistic Regression, KNeighbors Classifier, Decision Tree and Random Forest machine learning algorithm to train on our training dataset to build models. The following code shows how I use K-fold cross validation and build model method to train and evaluate models.

We mainly focus on recall score in confusion matrix instead of precision and accuracy scores to evaluation the performance of each model. The recall is the ratio of correct positive predictions to the total positives examples, which means how well the model be able to detect the true default. Banks will lose profit and take more risks if clients cannot pay back the loan, if our model could filter the clients that are more likely to default when they apply for loans. Banks can decline their applications and thus control the risk.

After briefly checking the performance of each model, although KNeighbors has the best F1 score, Logistic Regression has better recall score. But I still build pipeline and tune the hyperparameter on the four ML algorithms and I get the best result from the best hyperparameter of Random Forest.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from imblearn.over_sampling import RandomOverSampler
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from xgboost import XGBClassifier

from scikitplot.metrics import plot_roc
from sklearn.metrics import roc_auc_score


#data is too small
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

kf = KFold(n_splits=5, random_state=42, shuffle=True)

models=[LogisticRegression(random_state=1),
        KNeighborsClassifier(),
        DecisionTreeClassifier(),
        RandomForestClassifier()
        ]


for model in models:
    precision_scores = []
    recall_scores=[]
    f1_scores=[]
    accuracy_scores=[]
    roc_auc_scores=[]
    for train_index, test_index in kf.split(X_train):
        X_train1, X_test1 = X_train.iloc[train_index], X_train.iloc[test_index]
        y_train1, y_test1 = y_train.iloc[train_index], y_train.iloc[test_index]

        scaler = StandardScaler()
        X_train_scaled = scaler.fit_transform(X_train1)
        X_test_scaled = scaler.transform(X_test1)

        ros = RandomOverSampler(random_state=0)

        X_train_resampled, y_train_resampled = ros.fit_resample(X_train_scaled, y_train1)

        model.fit(X_train_resampled, y_train_resampled)

        y_pred = model.predict(X_test_scaled)
        y_proba = model.predict_proba(X_test_scaled)

        precision_scores.append( precision_score(y_test1, y_pred))
        recall_scores.append(recall_score(y_test1, y_pred))
        f1_scores.append(f1_score(y_test1, y_pred))
        accuracy_scores.append(accuracy_score(y_test1, y_pred))
        roc_auc_scores.append(roc_auc_score(y_test1, y_proba[:, 1]))
    print(model)
    print(np.mean(precision_scores))
    print(np.mean(recall_scores))
    print(np.mean(f1_scores))
    print(np.mean(accuracy_scores))
    print(np.mean(roc_auc_scores))
    print(classification_report(y_test1, y_pred))
```

```
LogisticRegression(random_state=1)
0.4492307692307692
0.7801298701298702
0.5696207018787665
0.855045871559633
0.9228271909394931
              precision    recall  f1-score   support

           0       0.95      0.85      0.90        93
           1       0.46      0.75      0.57        16

    accuracy                           0.83       109
   macro avg       0.71      0.80      0.73       109
weighted avg       0.88      0.83      0.85       109

KNeighborsClassifier()
0.3772138047138047
0.6760281385281385
0.47739771737532094
0.8165137614678899
0.8121888295387052
              precision    recall  f1-score   support

           0       0.94      0.90      0.92        93
           1       0.55      0.69      0.61        16

    accuracy                           0.87       109
   macro avg       0.75      0.80      0.77       109
weighted avg       0.89      0.87      0.88       109

DecisionTreeClassifier()
0.5434498834498834
0.49136363636363634
0.5058739091992466
0.8844036697247706
0.7164709122032724
              precision    recall  f1-score   support

           0       0.90      0.97      0.93        93
           1       0.67      0.38      0.48        16

    accuracy                           0.88       109
   macro avg       0.78      0.67      0.71       109
weighted avg       0.87      0.88      0.87       109

RandomForestClassifier()
0.8400000000000001
0.4971103896103896
0.616
0.9266055045871561
0.9328351581886822
              precision    recall  f1-score   support

           0       0.91      0.98      0.94        93
           1       0.78      0.44      0.56        16

    accuracy                           0.90       109
   macro avg       0.84      0.71      0.75       109
weighted avg       0.89      0.90      0.89       109
```

9. Conclusion

The performance of random forest model gives us the best result. The model gives fair recall and slightly low precision score. The model now detects more than half of clients who are more likely to default but it also includes some of other classes. This scenario indicates that models could misclassify some good clients. If the model is deployed in production, it could result in declining loan application from some clients who are not going to default loan. However, the model would catch the majority of potential customer who are going to default on loan, and it would only affect a small portion of good clients regrading to their loan application. Banks become more proactive in leveraging the risk by using the model.

From the figures above in the top right which is interpreting the feature importance of RF model, it tells what are the main factors that makes the model think they are going to default. People who had negative amount in their balance and their balance at the time they place the loan application plays a significant role in predicting loan default. These importance factors would help banks to perform profiling analysis on each account user, as well as have better understand of what kind of clients have high potential of not getting loan based on their account' condition