Capstone 2: Milestone Report

Problem

Businesses need to be smart in order to survive in the competitive business world. Knowing what its clients need, being able to meet that demand, and potentially attracting new clients are all vital to keeping a business successful and running. TalkingData, founded in 2011, is China's largest data intelligence solution provider, and they have the capability to help enterprises expand their services.

This dataset contains millions of mobile user data. Based on the device, location, and most importantly app usage, I will attempt to predict the user's demographics. This will enable businesses to improve their marketing strategies to better meet their customers' demands and needs.

Target Audience

Since this dataset deals with mobile device usage, my target audience will be mobile phone companies. At a glance, the dataset contains some demographic and geographic information such as age, gender, longitude, and latitude. With location data, companies can identify the cities where their products are most popular. Demographic data help to determine the types of consumers businesses are attracting. These analysis and visualization techniques will be demonstrated in this project.

Data

The data was acquired via a past Kaggle competition. The data source is comprised of 8 separate CSV files:

1. app_events.csv
2. app_labels.csv
3. events.csv
4. gender_age_test.csv
5. gender_age_train.csv
6. label_categories.csv
7. phone_brand_device_model.csv
8. sample_submission.csv

The "sample_submission.csv" is irrelevant, so it is excluded from this project.

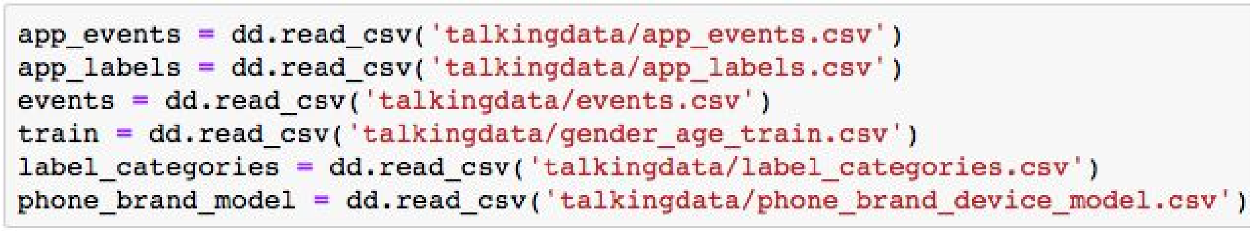
Data Wrangling

To start off, a couple of the CSV files contains millions of rows of data. These files, which were loaded into their own dataframe, required too much memory usage to process. As a solution, instead of using the common pandas package, Dask was used instead.

When datasets are too large for Pandas, Scikit-learn, and Numpy to process on a single machine, tools like Spark and Dask are utilized. The large datasets can be spread over clusters with multiple nodes to enable parallelism. Spark is written in Scala and unfortunately does not have full support for Python. Dask, on the other hand, is written in Python, so was chosen for this project.

Dask, as mentioned above, enables parallel computations. A single Dask dataframe is composed of many smaller Pandas dataframes. Luckily, Dask dataframes uses the Pandas API, so Python users who are comfortable with Pandas should not have difficulty using dask dataframes. Due to its parallel nature though, computations are only triggered by calling the ".compute()" method on dask dataframes.

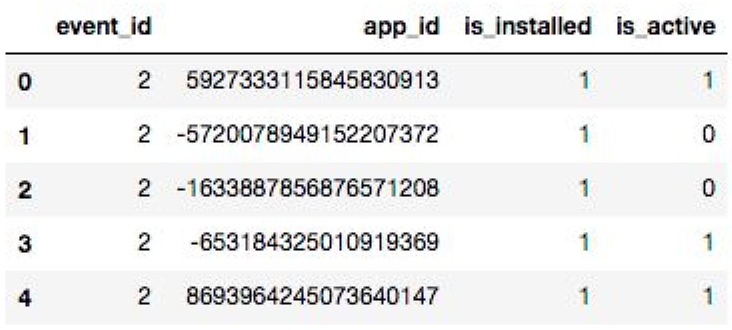
Shifting focus back to the project, each of the relevant CSV files are loaded into its own dask dataframe



```
app_events = dd.read_csv('talkingdata/app_events.csv')
app_labels = dd.read_csv('talkingdata/app_labels.csv')
events = dd.read_csv('talkingdata/events.csv')
train = dd.read_csv('talkingdata/gender_age_train.csv')
label_categories = dd.read_csv('talkingdata/label_categories.csv')
phone_brand_model = dd.read_csv('talkingdata/phone_brand_device_model.csv')
```

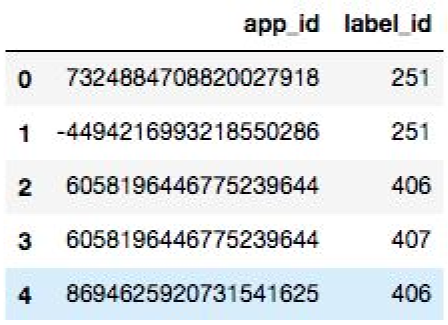
Here is a preview of the first 5 rows of each dataframe:

- **app_events**



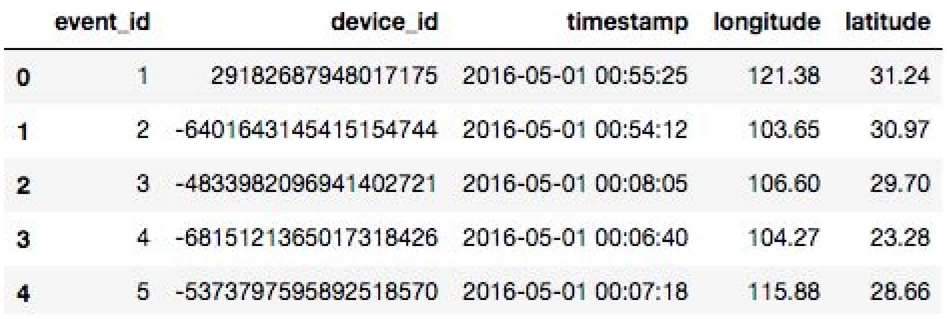
	event_id	app_id	is_installed	is_active
0	2	5927333115845830913	1	1
1	2	-5720078949152207372	1	0
2	2	-1633887856876571208	1	0
3	2	-653184325010919369	1	1
4	2	8693964245073640147	1	1

- **app_labels**



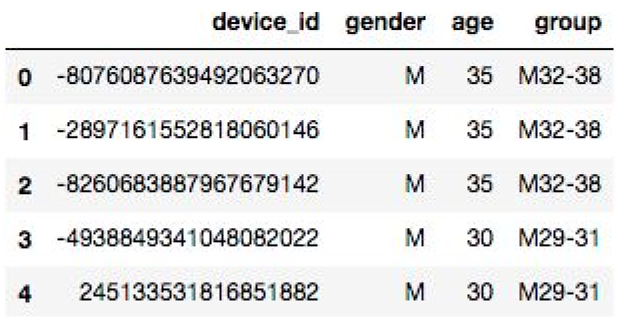
	app_id	label_id
0	7324884708820027918	251
1	-4494216993218550286	251
2	6058196446775239644	406
3	6058196446775239644	407
4	8694625920731541625	406

- **events**



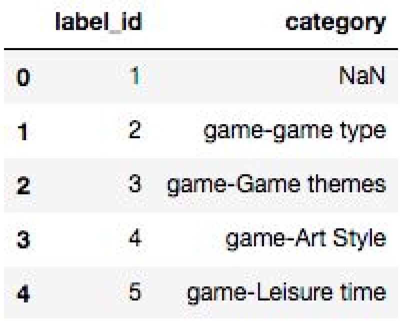
	event_id	device_id	timestamp	longitude	latitude
0	1	29182687948017175	2016-05-01 00:55:25	121.38	31.24
1	2	-6401643145415154744	2016-05-01 00:54:12	103.65	30.97
2	3	-4833982096941402721	2016-05-01 00:08:05	106.60	29.70
3	4	-6815121365017318426	2016-05-01 00:06:40	104.27	23.28
4	5	-5373797595892518570	2016-05-01 00:07:18	115.88	28.66

- **train**



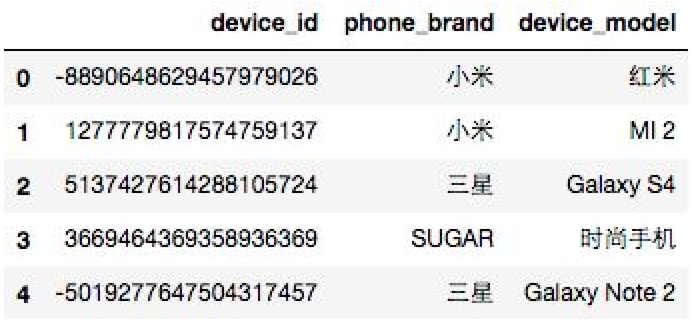
	device_id	gender	age	group
0	-8076087639492063270	M	35	M32-38
1	-2897161552818060146	M	35	M32-38
2	-8260683887967679142	M	35	M32-38
3	-4938849341048082022	M	30	M29-31
4	245133531816851882	M	30	M29-31

- **label_categories**



	label_id	category
0	1	NaN
1	2	game-game type
2	3	game-Game themes
3	4	game-Art Style
4	5	game-Leisure time

- **phone_brand_model**

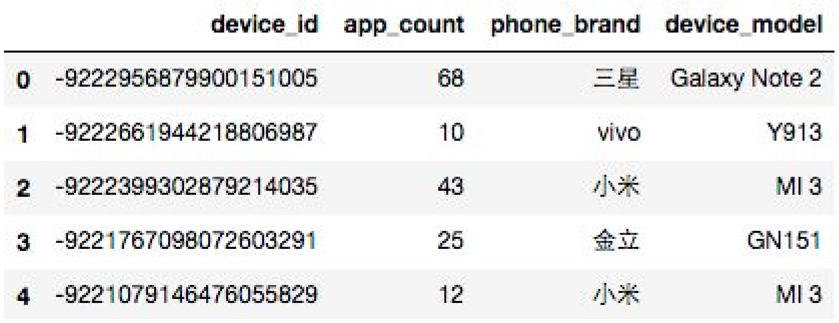


	device_id	phone_brand	device_model
0	-8890648629457979026	小米	红米
1	1277779817574759137	小米	MI 2
2	5137427614288105724	三星	Galaxy S4
3	3669464369358936369	SUGAR	时尚手机
4	-5019277647504317457	三星	Galaxy Note 2

One thing to note on the **app_events** dataframe is that it contains a “screenshot” of the apps that are on a user’s device whenever he/she opens an app that utilizes the TalkingData SDK. Each time a user opens an app, it gets recorded in the **events** dataframe. If a user opens two of such apps, two separate events will be recorded in **events** and two screenshots of the applications on the device will be recorded in **app_events**. Ultimately, this means there are duplicate applications recorded for a single device in **app_events**.

With that in mind, it might be interesting to look at the number of apps installed in each phone. In order to achieve that, two merges needed to be done, along with some aggregating to remove duplicates:

1. Grouping **events** by “device_id” and only getting the *first* occurrence of the ID. This way, we only get one screenshot of the mobile apps of each device
2. Merge the above dataframe with **app_events** by “event_id” to get the list of applications on each device
3. Group the above dataframe by “device_id” again and aggregate by count. This gives us the number of mobile apps
4. This step was optional, but to get the corresponding brand and model of each device, another merge was performed on the above dataframe to **phone_brand_model** (merging on “device_id”)



	device_id	app_count	phone_brand	device_model
0	-9222956879900151005	68	三星	Galaxy Note 2
1	-9222661944218806987	10	vivo	Y913
2	-9222399302879214035	43	小米	MI 3
3	-9221767098072603291	25	金立	GN151
4	-9221079146476055829	12	小米	MI 3

“*num_apps” dataframe*

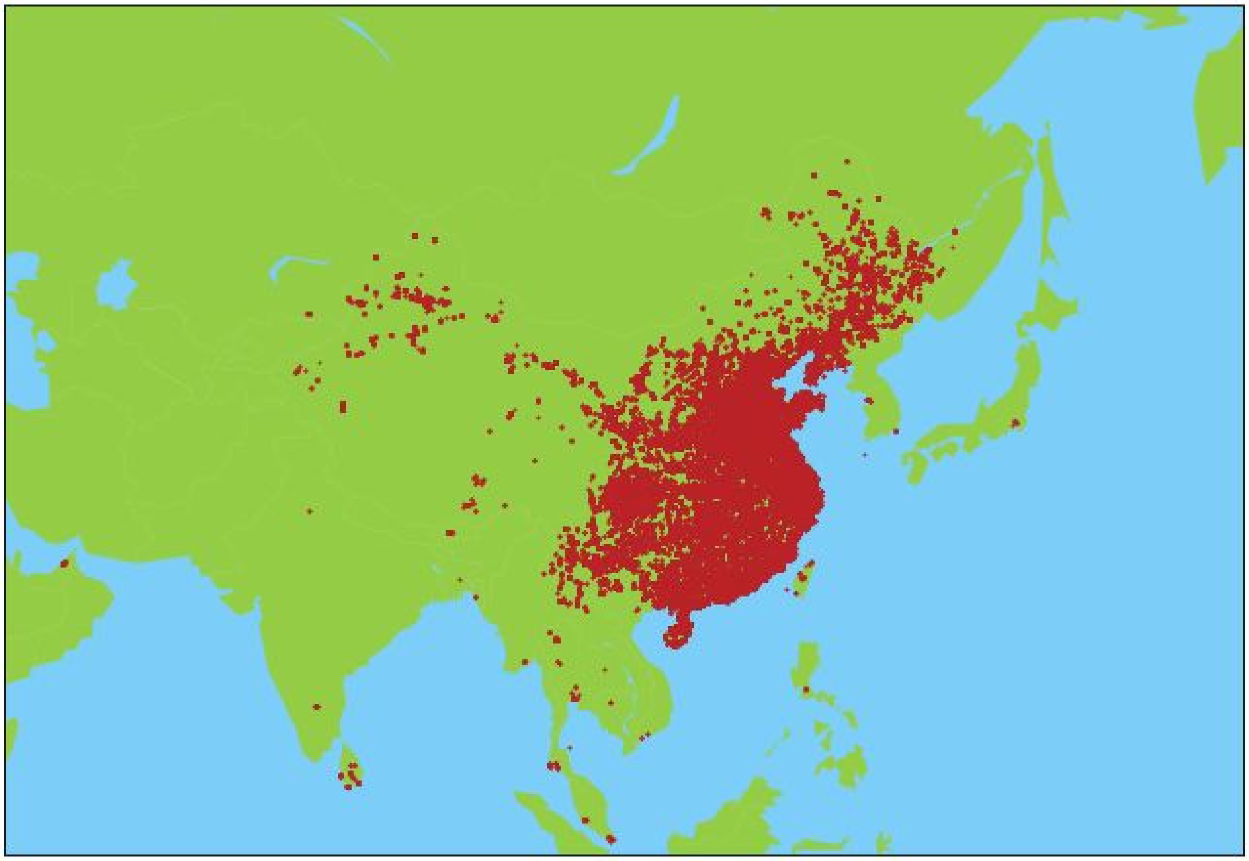
Another dataframe was created with demographic information. The demographic data could have been merged to the above dataframe, but will reduce the size of the dataframe tremendously (from 42,621 rows down to 16,951), thus the decision to keep the dataframes separate.

	device_id	gender	age	group	app_count	phone_brand	device_model
0	-8260683887967679142	M	35	M32-38	53	小米	MI 2
1	7477216237379271436	F	37	F33-42	26	华为	荣耀6 plus
2	6352067998666467520	M	32	M32-38	19	华为	荣耀畅玩4X
3	8026504930081700361	M	25	M23-26	31	小米	MI 4
4	-7271319853104672050	M	27	M27-28	34	三星	Galaxy Note 3

"gender_info" dataframe

Exploratory Data Analysis

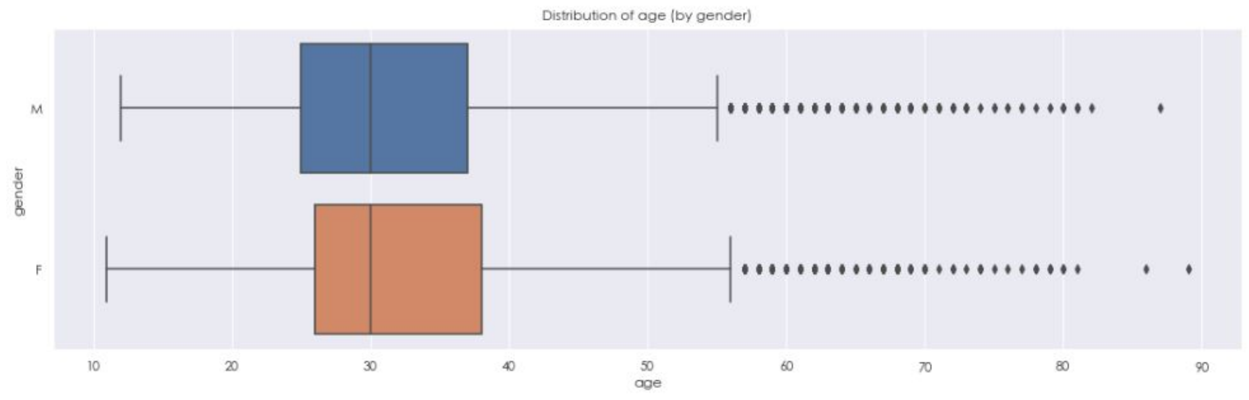
First, to get a visual of the location of users, the "longitude" and "latitude" were plotted with the Basemap tool.



Location of users (when opening an app with TalkingData SDK)

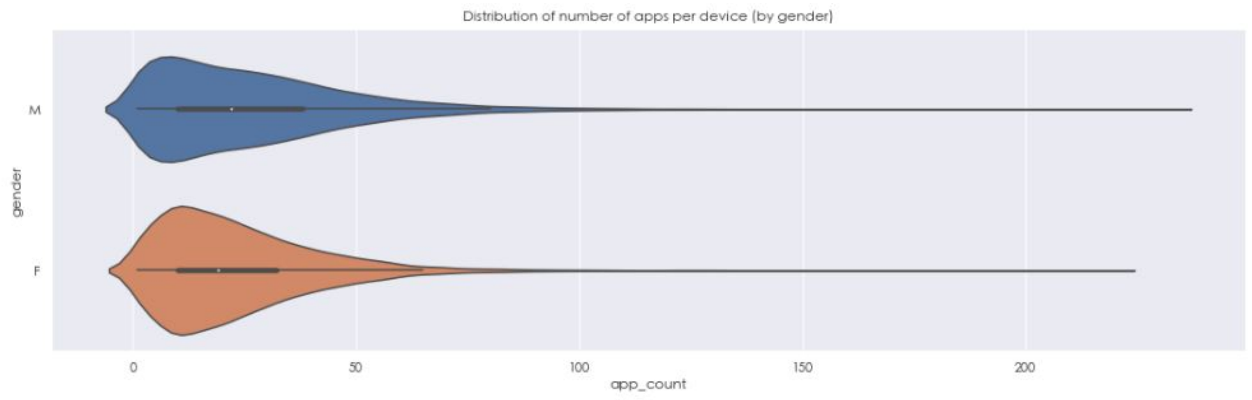
When plotted, the pairs of longitude and latitude points clearly form the shape of China. Quite evidently, most of the data gathered are from the eastern region of China. This makes sense as the 5 largest cities in China are all located in eastern China.

Now, I wonder if males and females have different behaviors when it comes to mobile app usage. To start off, let's use **gender_info** and take a look at the age distribution of this dataset.

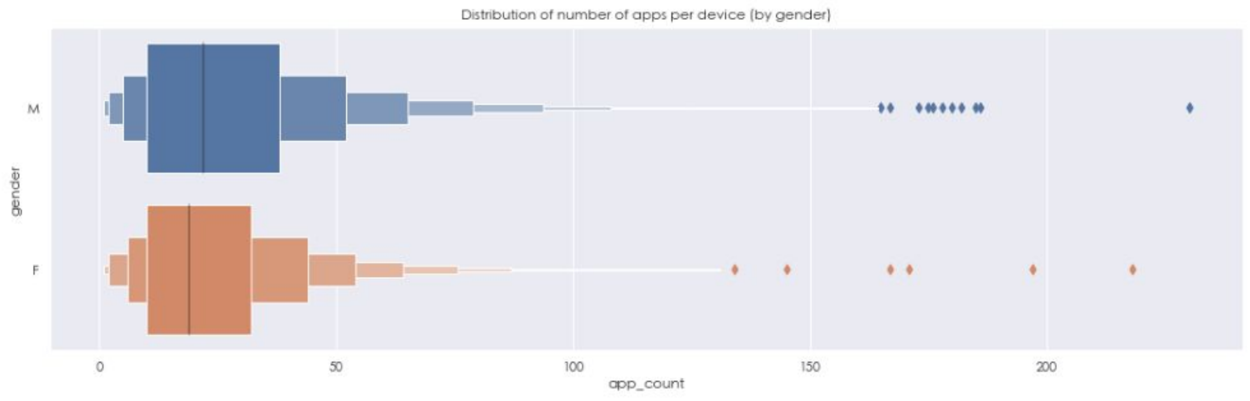


Distribution of age (by gender)

The age distribution between males and females seem to be fairly similar. The age range for females might be infinitesimally wider. Now, let's look at the distribution of total apps per device.



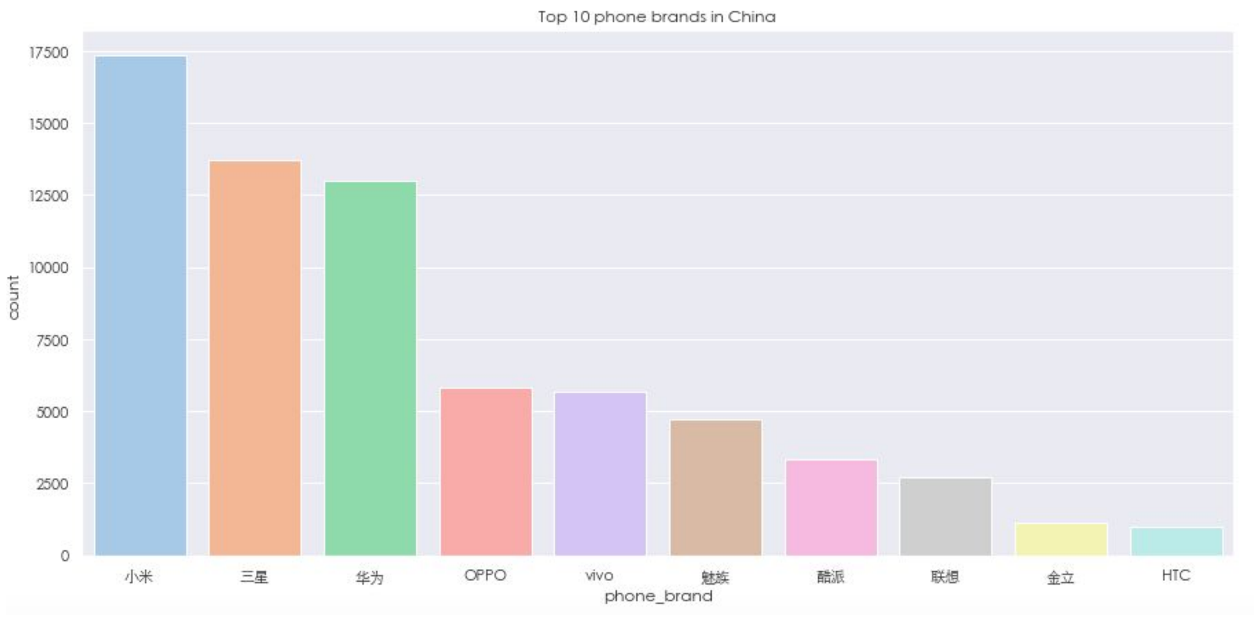
Violin plot of distribution of total app count (by gender)



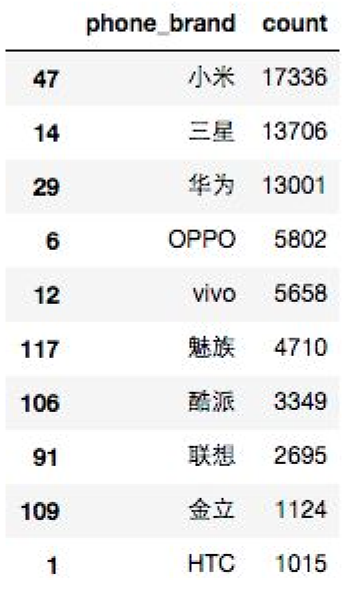
Boxen plot of distribution of total app count (by gender)

Looking at both categorical distribution plots, we can see that Chinese men, in this dataset, will generally have more mobile apps than Chinese women. A larger portion of the females have 10-20 apps. This applies to males too, but looking at the violin plot, the violin representing males is slightly more stretched out and the tip doesn't form as rapidly, so my assumption is most guys in this dataset are likely to have more apps than females.

Now, onto the actual device, I wanted to see which mobile phone companies dominate the industry.

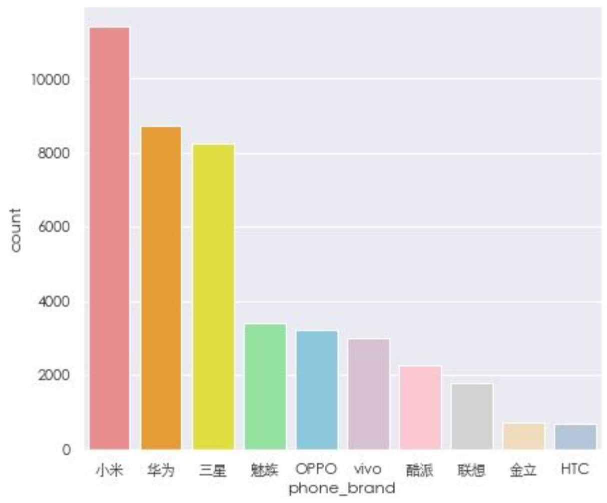


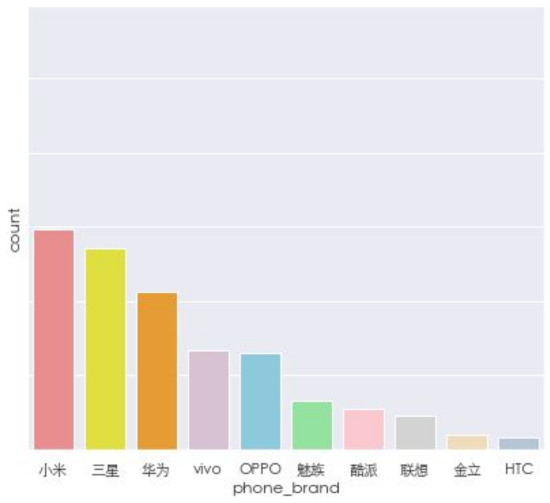
Top 10 mobile phone brands in China



	phone_brand	count
47	小米	17336
14	三星	13706
29	华为	13001
6	OPPO	5802
12	vivo	5658
117	魅族	4710
106	酷派	3349
91	联想	2695
109	金立	1124
1	HTC	1015

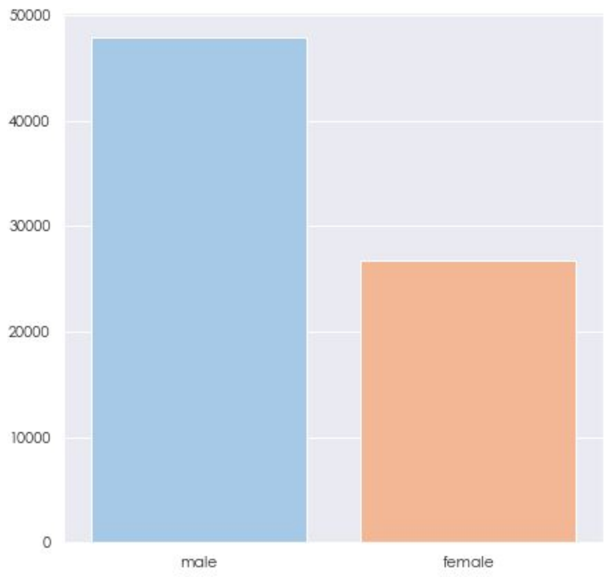
The 3 most popular brands are XiaoMi, Samsung, and Huawei. There are at least two times more Huawei (3rd) devices than there are OPPO (4th) devices, which is a good indication that the top 3 brands in this dataset are the “big three” to Chinese consumers. Now, do guys and girls have the same preferences when it comes to mobile devices?





XiaoMi devices (red) seem to hold first place for both genders. Huawei devices (orange) are more popular than Samsung devices (yellow) to the male population, while the opposite is true for females. Also, Meizu devices (green) attract more male consumers and are seemingly less popular to females.

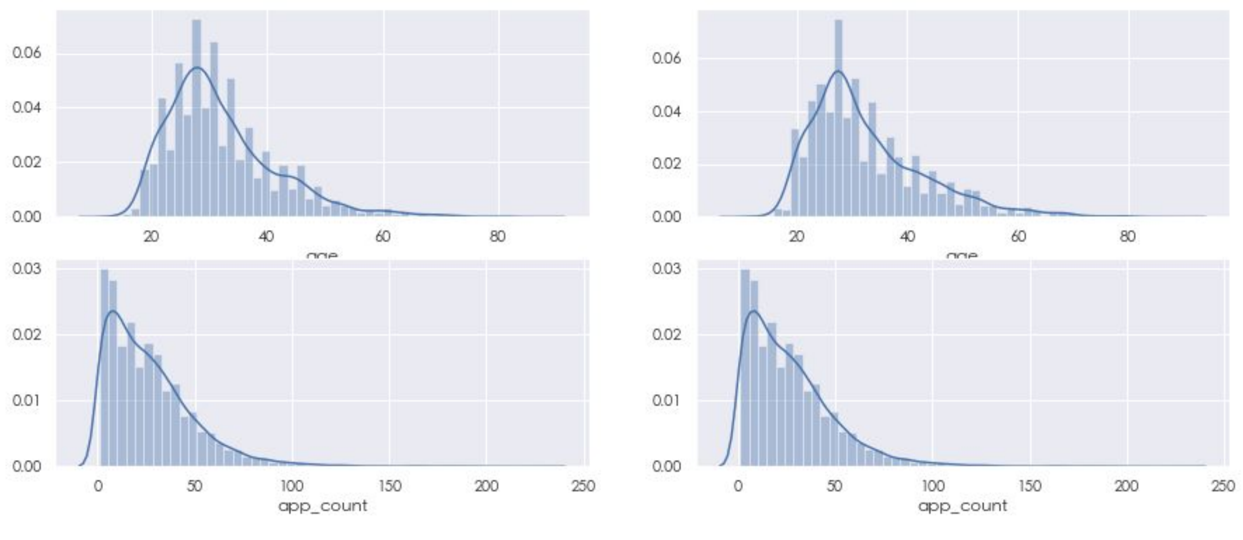
Both graphs above share the same y-axis. Just by comparison, it seems there are more data gathered from males than females. Let’s confirm by filtering the **train** dataframe (it is the only dataframe that contains demographic data) to create two separate dataframes based on gender.



Indeed, this dataset contains far more male mobile users -- nearly two times more than females. Should we be concerned that this dataset is not representative of the entire Chinese population? Possibly not due to China having a far larger male population; Chinese men outnumber women by more than 30 million. This is very likely due to the country's previous one child policy. Since China's gender imbalance is large and the majority of the data collected are of consumers who are located in the most populous cities of China, I am inclined to believe this dataset is representative of the Chinese population.

Inferential Statistics

To start off, I examined the distribution of various features. Most, if not all, have a skewed distribution. Below is the distribution of age (top) and number of mobile apps (bottom).

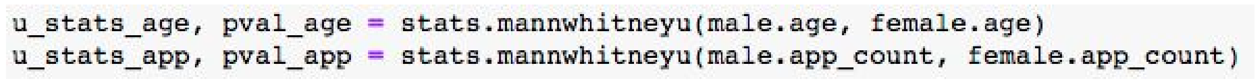


Skewed distribution of 'age' and 'app_count', filtered my gender (male on left, female on right)

Due to the absence of a normal distribution, nonparametric tests have to be performed instead. The Mann-Whitney U test is a nonparametric test that tests whether two samples are likely to derive from the same population. The distributions of “age” and “app_count” were previously visualized, so I chose to use them for this test. The null hypotheses are:

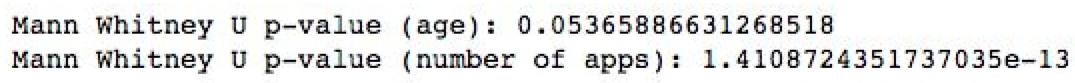
- The distributions of age between both genders are the same
- The distributions of number of apps per device by gender are the same

The chosen significance level is 0.05.



```
u_stats_age, pval_age = stats.mannwhitneyu(male.age, female.age)
u_stats_app, pval_app = stats.mannwhitneyu(male.app_count, female.app_count)
```

Mann Whitney U computation in Python



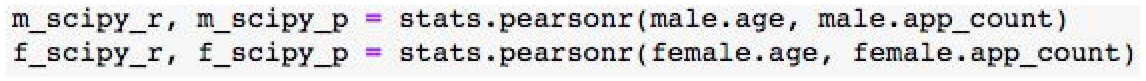
```
Mann Whitney U p-value (age): 0.05365886631268518
Mann Whitney U p-value (number of apps): 1.4108724351737035e-13
```

Mann Whitney U p-value

The p-value for the distributions of age is slightly bigger than the significance level, so the null hypothesis is not rejected. The distributions of age between genders are the same.

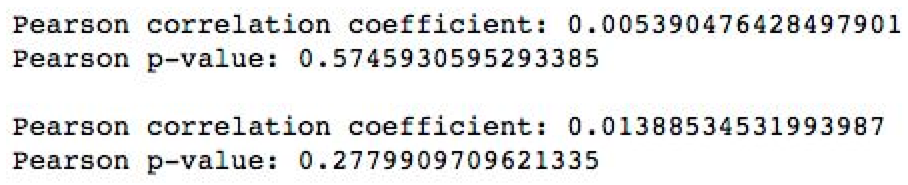
In contrast, the p-value for the distributions of total number of mobile apps is extremely small, smaller than the significance level, so the null hypothesis is rejected. The distributions are not the same. This aligns with the speculation previously made that Chinese men seem to generally have more apps installed in their mobile devices.

To my understanding, the pearson correlation coefficient doesn’t require variables to have a normal distribution, so out of curiosity, I tested the correlation between “age” and “app_count”. The null hypothesis is that there is no correlation between age and number of apps and the significance level is 0.05.



```
m_scipy_r, m_scipy_p = stats.pearsonr(male.age, male.app_count)
f_scipy_r, f_scipy_p = stats.pearsonr(female.age, female.app_count)
```

Pearson correlation coefficient computation in Python



```
Pearson correlation coefficient: 0.005390476428497901
Pearson p-value: 0.5745930595293385

Pearson correlation coefficient: 0.01388534531993987
Pearson p-value: 0.2779909709621335
```

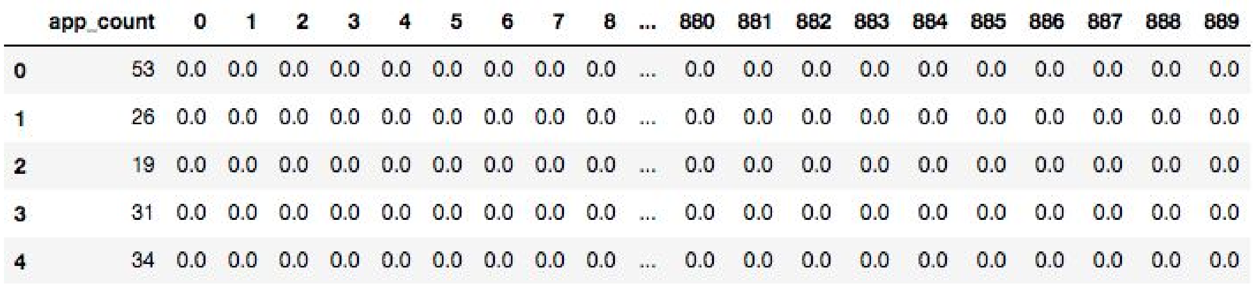
Pearson correlation coefficient and p-value

Both p-values are bigger than 0.05, so the null hypotheses are not rejected. Regardless of gender, age and number of apps installed have no correlation.

Machine Learning

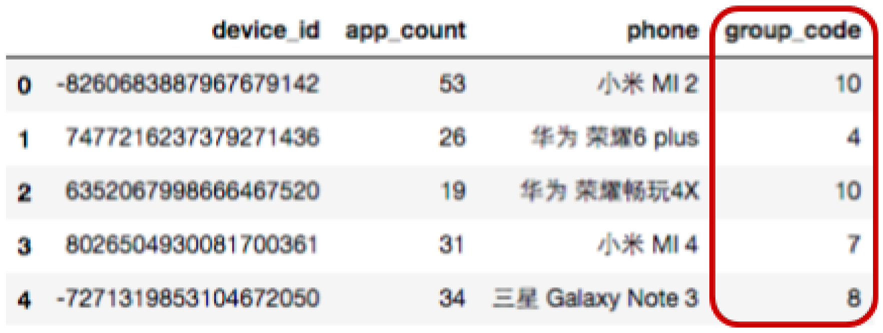
For this project, since I am attempting to answer a classification question, I chose the Random Forest Classifier to predict the gender/age group of a user.

Previously, an observation was made where it seems that Chinese men and women have slightly different preferences when it comes to brands of mobile phones. Device model is unique to each brand, so instead of having "phone_brand" and "device_model" in separate columns, they were combined into one column instead. This along with the number of apps per device were used as predictor variables.



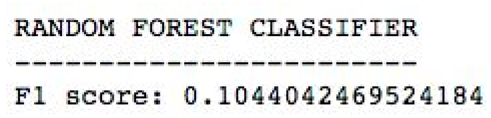
	app_count	0	1	2	3	4	5	6	7	8	...	880	881	882	883	884	885	886	887	888	889
0	53	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	26	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	31	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	34	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

The goal of this project is to predict the gender/age group. In the dataset, the "group" column is of string type. Since this is a classification problem, the column is converted into a category type with a numeric code to represent each unique category (e.g., 10 is the categorical code for M32-38). This column is the target variable.



	device_id	app_count	phone	group_code
0	-8260683887967679142	53	小米 MI 2	10
1	7477216237379271436	26	华为 荣耀6 plus	4
2	6352067998666467520	19	华为 荣耀畅玩4X	10
3	8026504930081700361	31	小米 MI 4	7
4	-7271319853104672050	34	三星 Galaxy Note 3	8

The predictor and target variables are split into training and test sets. The random forest classifier is fitted to these two sets, and based on the predictions of the model, yields a F1 score (the chosen metric) of 0.1.



```
RANDOM FOREST CLASSIFIER
------------------------
F1 score: 0.1044042469524184
```

As can be seen, this model does a poor job at correctly predicting the gender/age group, given just the total app count and phone type. I suspect the category of mobile phone applications will be able to tremendously improve the model's performance because I believe different kinds of apps attract different types of people. Unfortunately, I was unable to make use of the category data due to limited understanding of text mining.

It is also possible that the app usage (in the case of this dataset, the date and time the app is opened) will strengthen the model's performance. For example, those belonging in the youngest age group are possibly students, and generally, students are unable to use their phones during class. If an event is logged during school time, the mobile user is probably not a student, therefore less likely to be in “M22-” or “F23-”.

This project will be revisited and will incorporate the ideas mentioned above. As of now, the two predictor variables are not enough to create a good classification model.