Springboard's Data Science Intensive program using Python

Capstone Project Report

Predicting Mobile User Demographics of TalkingData

By Adarsh Srinivas

Mentor: Hobson Lane

Problem to be solved and Motivation:

Demographics are widely used in marketing to characterize different types of customers. However, in practice, demographic information such as age, gender, and location is usually unavailable due to privacy and other reasons. The task of this challenge is to predict users' demographics (age and gender) based on their app download, usage, geolocation, and mobile device properties. A postpaid mobile user is required to create an account by providing detailed demographic information (e.g., name, age, gender, etc.). However, a recent report indicates that there is still a large portion of prepaid users (also commonly referred to as pay-as-you-go) who are required to purchase credit in advance of service use. Statistics show that 95% of mobile users in India are prepaid, 80% in Latin America, 70% in China, 65% in Europe, and 33% in the United States. Even in the U.S., the switch to prepaid plans is accelerating during the economic recession from 2008. Prepaid services allow the users to be anonymous—no need to provide any user-specific information.

Client:

1. ***TalkingData:*** TalkingData is one of the largest mobile-service provider's in China. The Kaggle competition is about predicting age and gender of the mobile user based on App-data. TalkingData is seeking to leverage behavioral data from more than 70% of the 500 million mobile devices active daily in China to help its clients better understand and interact with their audiences.

2. ***CEO of LotusFlare, Mr. Sam Gadodia:*** One of my friend's relative is a founder of a similar mobile data service providing company based in the Silicon Valley (LotusFlare). They are looking to solve some similar problems and using my algorithms developed for TalkingData, I hope to present to him my findings and see if they can be applied to solve some of their business problems. The same algorithms or after certain tweaks to the algorithms, can be used by his company in their marketing efforts, personalizing experience of their users, recommendations and so on.

Data:

The data was obtained from the ongoing Kaggle Competition.

https://www.kaggle.com/c/talkingdata-mobile-user-demographics/data

Important fields and Information in the Dataset:

The dataset provided by kaggle consists of seven different csv files. These are events.csv, app_events.csv, app_labels.csv, label_categories.csv, phone_brand_device_model.csv, gender_age_train.csv and gender_age_test.csv. A detailed description of the various data files is given below:

- **gender_age_train.csv, gender_age_test.csv** - The training and test set
    - group: This is the target variable that I am going to predict
- **events.csv, app_events.csv** - When a user uses TalkingData SDK, the event gets logged in this data. Each event has an event id, location (lat/long), and the event corresponds to a list of apps in app_events.
    - timestamp: When the user is using an app with TalkingData SDK
- **app_labels.csv** - Apps and their labels, the label_id's can be used to join with label_categories
- **label_categories.csv** - Apps' labels and their categories in text
- **phone_brand_device_model.csv** - Consists of device ids, brand, and models

The objective of this project is to predict the demographics i.e. the age group and gender for each device_id. Some of the important features given in the dataset that can be used for this prediction are phone_brand and device models in the phone_brand_device_model.csv. The category that a particular app belongs to from label_categories can also be used. As we see, there are already a lot of meaningful features given in the dataset that may be used to achieve our goal.

Data Wrangling and Cleaning:

The main challenge was that the data given by Kaggle come in the form of different csv files. In order to use it to build models, we need to perform merge operations as well as other manipulations. I have performed a number of merge operations to get the dataset in a format that can be used for predictions. This involves combining the app_labels and the label_categories files, merging app_events and events files to get a dataset with only active users, joining the active users file with the label_categories to get the device_id along with the different useful features that can be used for building the models. Secondly, the phone_brand names in the phone_brand_device_model dataset were in chinese. I have created a pandas dictionary to map and translate these chinese brand names to english for the sake of easy visualizations and data exploration. I have also performed group by operations and manipulations to find the maximum count of apps and categories for each device. There were 529 duplicate device_ids in phone, so I dropped these duplicates ids.

No additional sources or datasets were used to build models and do predictions. This is because this competition is hosted by a company TalkingData and they wanted the participants to use their company provided data to obtain accurate results.

Feature Engineering:

The features that were used to build our models were derived from phone brand, device model, app labels and the number of apps installed by each device.

I used LabelEncoder to one hot encode to help generate a large number of features. To deal with dimensionality reduction, I used csr_matrix whose sparse matrices will help us deal with these large number of features. I applied the same procedure for brand, model, apps installed and app labels.

Phone brand and device model:
To create simple yet powerful features, I label-encoded the phone brand and device model. Since the number of device models is large, a prefix was extracted from the name.

App labels and installed apps:
The same procedure was applied here and the app_ids were encoded. To know about the apps installed for each device, since the apps are related to devices through events, I merged the Primary Key (device_id) from events to app_events. Then, performed a groupby to group the result by device_id, app and the size (aggregate). For app labels, I merged the data frame applabels with dev.

In the events data, I first counted how many apps are installed for each device. I also generated other types of features extracted from app usage. The first is a simple counts on the number of installed apps. Next, I created bags of features for the installed apps and their associated labels. Finally, I combined all the features created together (from phone brand, device model, app and labels) to create X_train and X_test using hstack.

Exploratory Data Analysis and Observations:

There are 74,645 users in the training data set of Talking Data. Among these, 47,904 are males (around 64%) and 26,741 are females (around 36%).

The mean, median and mode for the variable Age are not equal as well as since the skewness and kurtosis are well over our accepted range of -1 to +1, we can say that the variable age is not normally distributed. Age group of 20 - 40 are the dominant age as we see a high peak for that age category. This means most of (maximum) Talkingdata's users fall in this age category. We also see that, females at old age are more active (use more mobile devices) than males at the same age.

Most of the talking data users are Males in the age group 23-26. The next age group is Males 32-38 with the difference between the two groups not significant. Females in the age group 27-28 are the least users of Talkingdata.

Creating a new feature, ‘hour’ from the timestamp variable in the initial dataset, we notice that hours 10 and 21 have the maximum number of events recorded in the talking data dataset. Hours 10 and 21 correspond to 10:00 AM and 9:00 PM. Such a visualisation makes sense as users are more likely to be active at 10:00 AM when they travel for work or start their day. Also, 9:00 PM represents a time after dinner and users are more likely to use their mobile and browse apps as they unwind and relax just before sleep.

Maximum events are recorded on a tuesday followed by a thursday. This is an interesting insight as you would assume maximum events to be occurring on weekends. This is further surprising as we see that the weekends (Saturday and Sunday) have the least number of events.

Filtering out the devices that are active, we find that there are 60669 active apps on devices, we notice that the maximum number of app on a single device is 1342. Finally, we observe that 75% of devices contain at most 21 apps.

Performing some data manipulation to get the number of categories for each device, we can see that 75% of devices have at most 45 categories. Maximum categories per device which are basically the outliers has 332 categories. Each user has an average of 31 categories on their device.

Performing a merge operation to see the popular phone brands of all the users, we observe that the top 5 brands of the Talkingdata users are Xiaomi, OPPO, Vivo, Samsung and Huawei. We can further see that the number of males in the top three phone brands i.e. Xiaomi, Samsung and Huawei are twice as much as the number of females for these brands.

Approach and Findings:

Our initial exploration and findings were concurrent with our claims that the features given in the dataset as well as the new features obtained through feature engineering would help us achieving good accurate results in predicting the demographics of the TalkingData users. The next step was to build machine learning models and obtain predictions on our test set. The first machine learning algorithm I applied was the Stochastic Gradient Descent (SGD) classifier. Following the scikit learn algorithm cheat sheet provided by my mentor, Mr. Hobson, this is the classifier that the sheet told me to start with according to the dataset provided by Kaggle.

In this project, our goal was not only say if an object belongs to class A or class B, but provide its probability that it belongs to these classes. Log Loss quantifies the accuracy of a classifier by penalising false classifications. The whole idea is to minimize the log loss which will lead to maximum accuracy on our classifier. So if we say, the classification is neutral and assign equal probability to each classes, the log loss would be less for e.g. around 0.8. In the second case, let's say the classifier is relatively confident in the first class. Since this is the correct classification the Log Loss is reduced to say 0.10. The third case is an equally confident classification, but this time for the wrong class. The resulting Log Loss increases to say 2.5. Keeping this concept in mind, I went ahead with my first algorithm.

The stochastic gradient descent classifier was built using scikitlearn. When the predictions were uploaded to Kaggle, I received a log loss score of 2.40891 which was quite high (Lower the Log loss, the better is your model). When the accuracy was checked using .score() on the train set, the SGD classifier gave us 0.258 which was quite low. This meant that the correlation between the predicted value and the actual truth was just 25.8%. For e.g. the SGD linear model was classifying say the green labels as red and the red ones as green and it could not find a clear decision boundary to separate the two classes.

The next model I tried was Logistic Regression. In logistic regression, we need to choose C (regularization constant) so that we obtain a good model. The default value of C is 1. I tried different values of C between 0.01-0.1. The log loss was the least for 0.02 and hence I chose 0.02 as the value of C. The correlation shown was 0.248 which was lesser than SGD but the Kaggle score from this model was 2.27331 which was much better and lesser than what we received from SGD. This meant the Logistic Regression model was better able to generalize on the test data set than SGD giving us a better log loss score. This was the default Logistic Regression model which is used to estimate the probability of a binary response based on one or more predictor.

In order to obtain better results, the next model I implemented was a Multinomial Logistic Regression model. Multinomial logistic regression is a classification method that generalizes logistic regression to multiclass problems, i.e. with more than two possible discrete outcomes. The kaggle score obtained was 2.26713, the best log loss score out of all the models. The rationale behind using Multinomial logistic regression was to optimize log loss and that was achieved using this model. My highest-scoring submission was an LB-Score of 2.26713 which gives a top 45% position in the private LB. The top score on the private LB was 2.13153.

Recommendations (So what? factor for the client):

Building demographic profiles for customers is critical and of great importance to all mobile service providers. Based on my findings and the predictions on the demographics (Age group and Gender) of the TalkingData SDK users, I recommend TalkingData the following suggestions:

1. Use this critical demographic information of your users to design better marketing strategies (identify potential customers and prevent customer churns). For e.g. specific apps/products can be recommended to young users like games, sports etc.

2. Supply users with more personalized services and focus on enhancing the communication experience.

3. Built a recommendation engine, implement personalization and behavioral targeting based on specific age groups and gender.

4. These demographic prediction results will help millions of developers and brand advertisers around the world pursue data-driven marketing efforts which are relevant to their users and catered to their preferences.

5. Produce personalized app selections, mobile data plans more likely to appeal to specific age group users using these machine learning techniques to analyze individual consumption patterns.

Future Research and Work:

To obtain a model that achieves a better log loss score than above, we can try running non-linear models. The three models that were implemented here were all linear models. A nonlinear model could provide us with better predictions on our test dataset as compared to linear models and that's a path that I certainly aim to take. One more reason that could contribute to a better model and give us a much better score is adding new features. Though, the features we used allow us to get decent predictions, we could create new features in order to see if better results can be achieved. This was a linear model based on the features that were derived from the dataset. New features created could help us generalize better on the test dataset. The features that were created here probably did not do well in generalizing on new unseen data and they warrant some thinking. Though creating the features that were used in the models were a challenge for me and required lot of work and thought, future work and some more familiarity/experience with python could help me come up with better descriptive features. I intend to continue with this project and hope to achieve even better results.

Trying nonlinear models and creating new features are the way to go ahead in this project and I am determined to research further keeping these two things in mind.