Predicting User Demographics Using App Usage Data

Final Report

Aaron Dsouza
Anushree Sinha
Harshit Amya
Vaishnavi Eleti
Vyom Vats

Date - December 7th 2016

Table of Contents

OBJECTIVES

Customer Personalization and Recommendation have become the unique selling points of any application developed today, for their power to impact user engagement, boost top line growth, and empower targeted advertising. Demographic information (age, gender, etc.) is the most crucial personalization tool used today. Our objective is to develop analytical models to predict Gender and Age of users[2] based on app usage logs.

The data for this project is hosted on Kaggle, and has been made available by TalkingData, China's largest third-party mobile data platform. We plan to analyze the data and estimate the probability of a user falling into demographic groups. Potential classification techniques that we are looking to use include logistic regression, XGBoost and Artificial Neural Networks. The technology stack we plan to use is Python to design and test the model, and D3/Python to build the interactive UI.

RELATED WORK

A variety of research has been conducted in the field of predicting user demographics, tapping different domains of data. One study[1] used Big Data to predict users' demographics based on their daily mobile communication patterns (calls and SMSs). Another[2] utilized device data such as the number of calls made or apps open at a particular time of the day, and predicted not only demographics like gender and age, but also job type, marital status, etc. Another[3] predicted the gender of YouTube users by leveraging user comments and user-video bipartite graphs. Yet another study[4], which is similar to our approach used app logs data from Android devices to predict demographics.

Potential limits of current practices include not using a large dataset[4], predicting results using only one algorithm[4][5], and applicability only when there is a diverse set of apps installed[5].

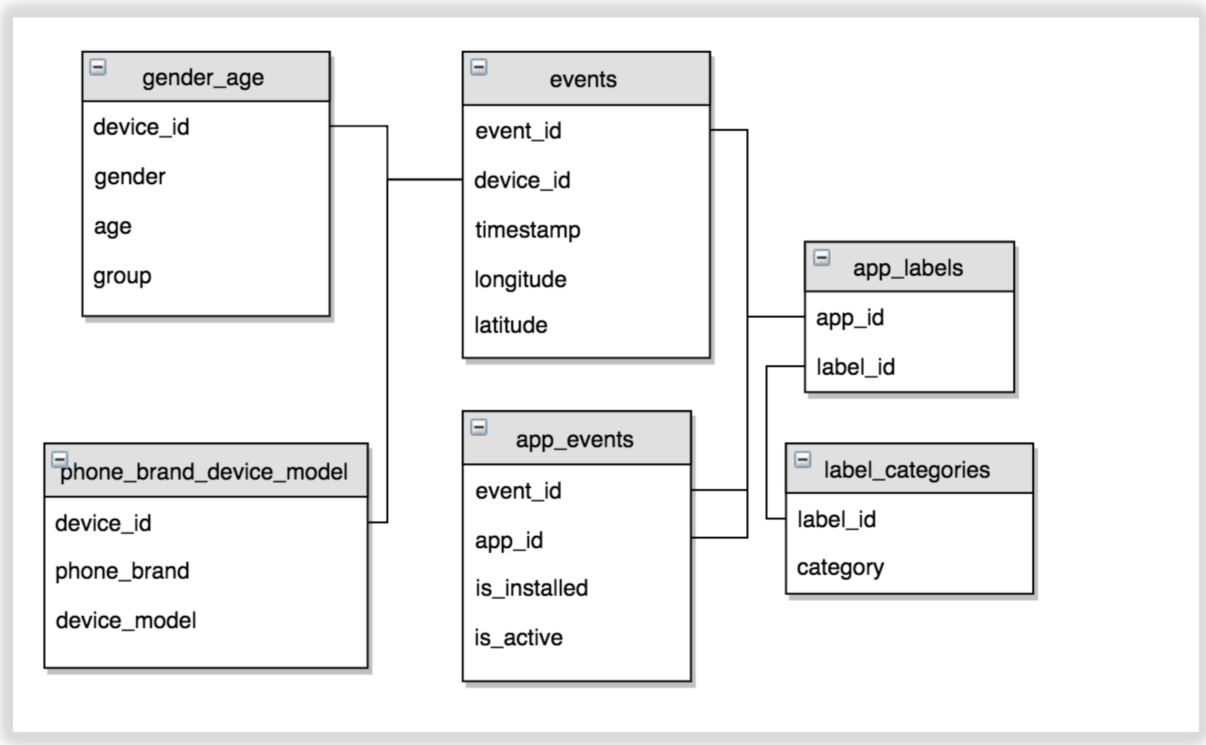
PROPOSED APPROACH

Our approach to this problem is four-fold:

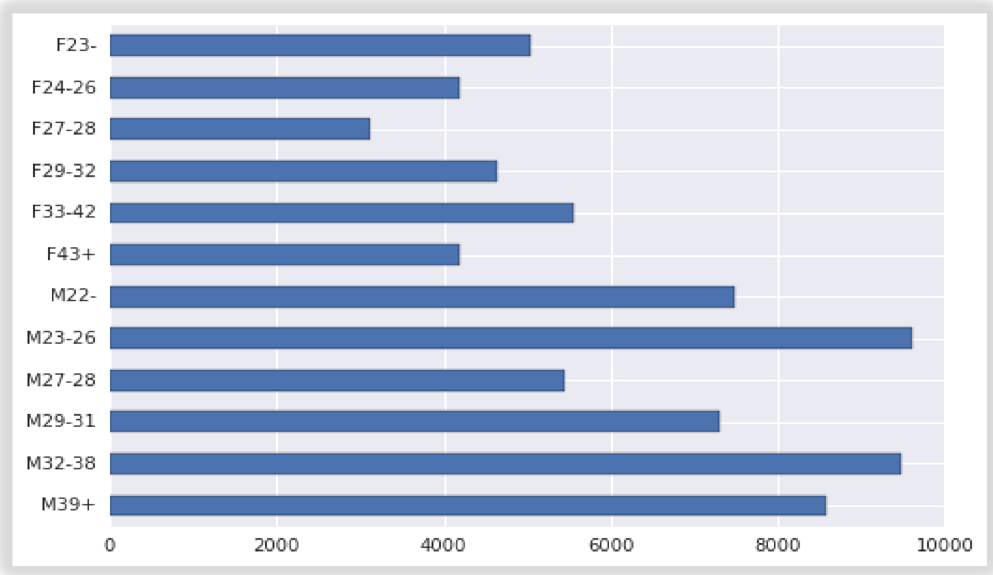
1. Exploratory Data Analysis
2. Feature Engineering
3. Data Wrangling & Model Development
4. Interactive Visualization

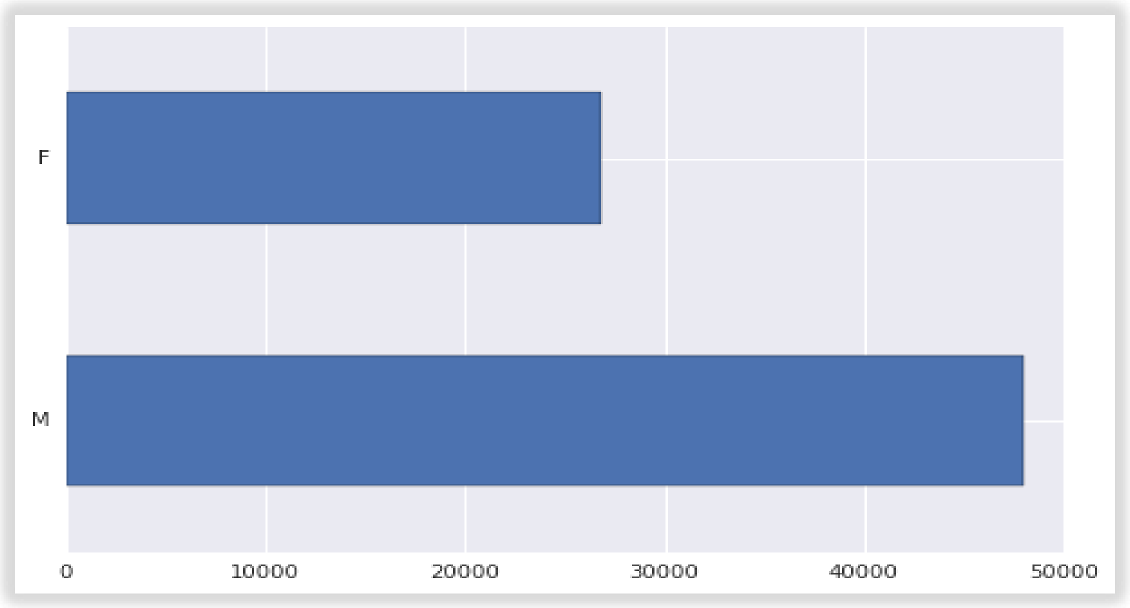
EXPLORATORY DATA ANALYSIS

We began by deepening our understanding of the dataset using some visualizations. Below is an entity-relationship diagram of the dataset:

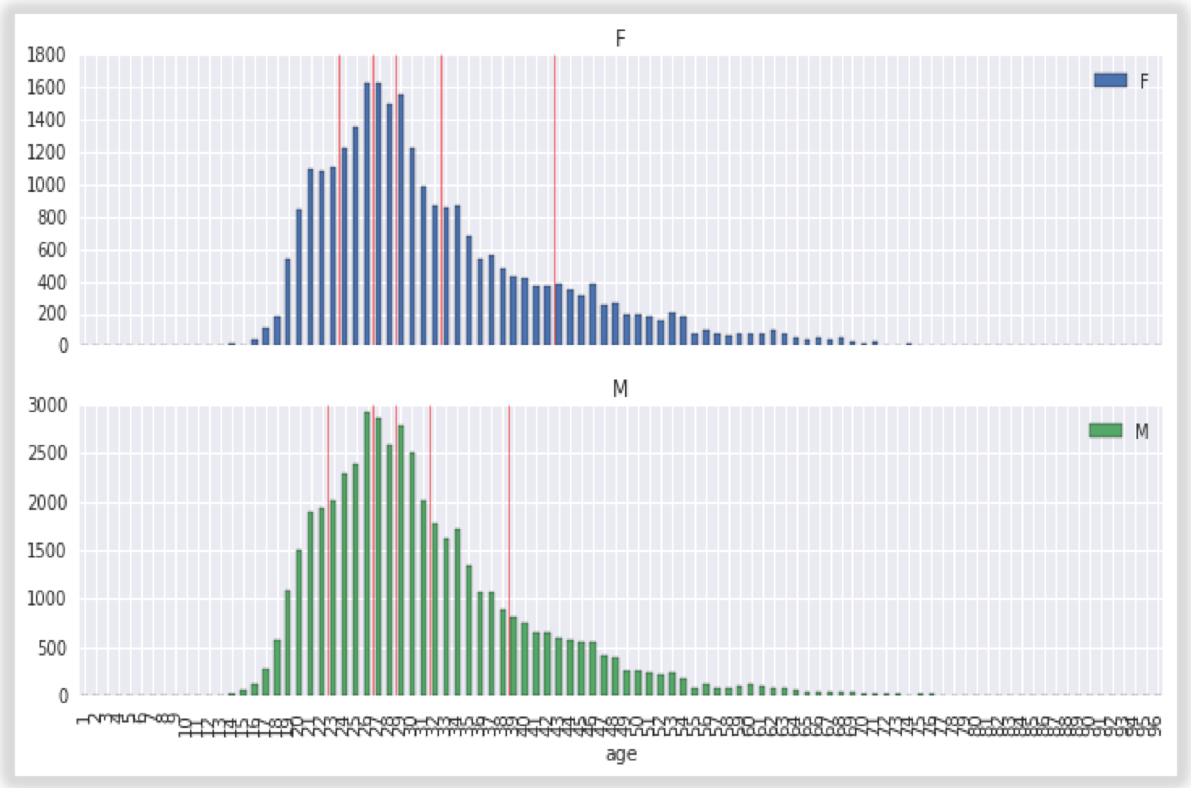


Next, we plotted age and gender distribution. Let us call these groups (F23-, etc.) target groups. Below is a representation of the count of target groups.





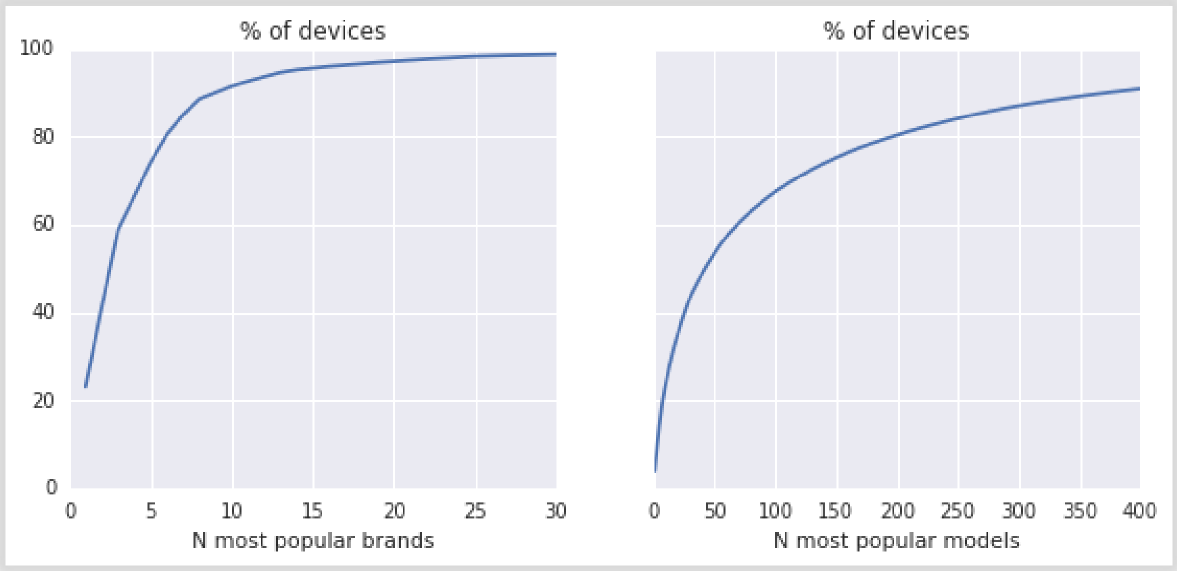
Below is a distribution of users according to age for female and males.



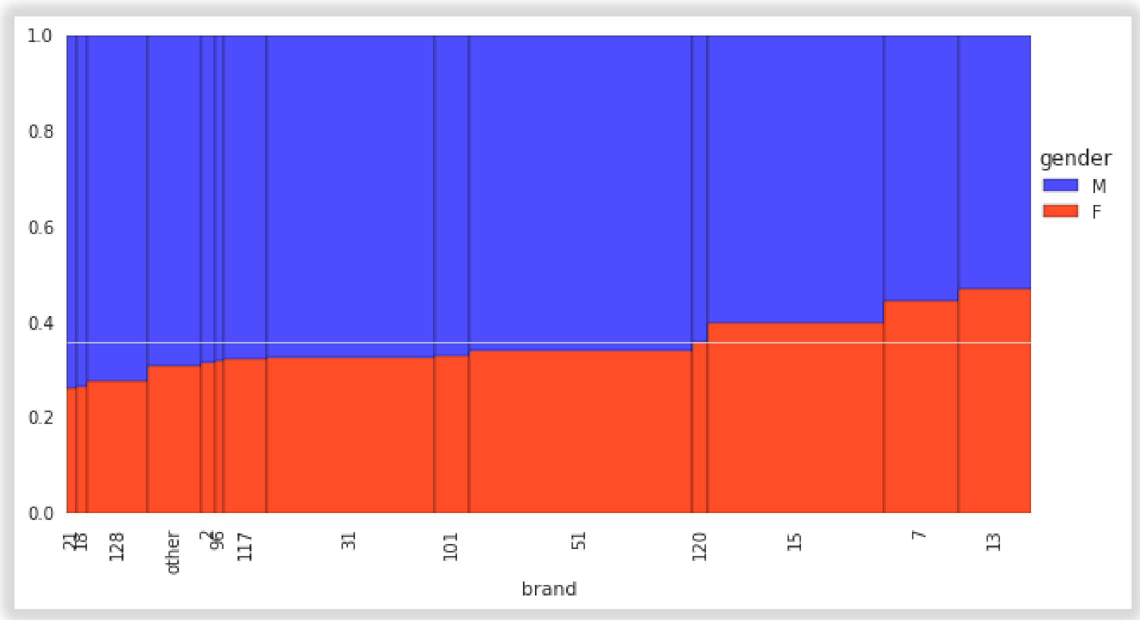
From this analysis, we learnt:

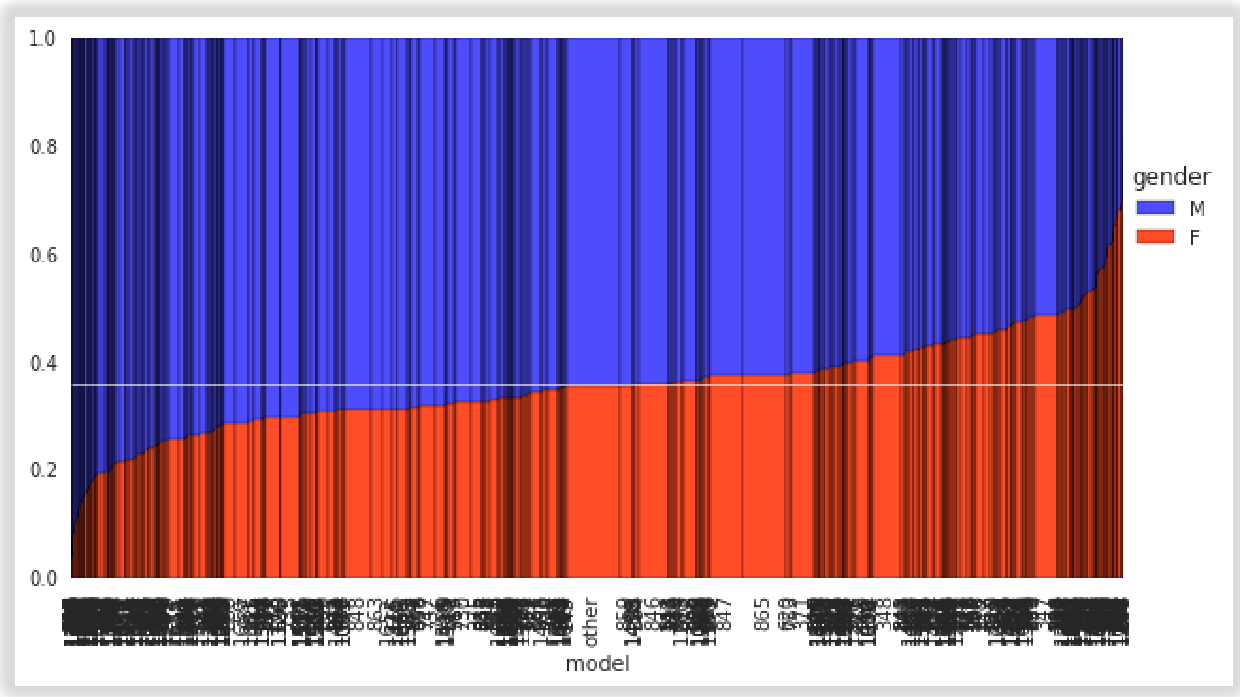
1. The file, phone_brand_device_models had double entries for 529 devices. Most of these rows were identical and could safely be removed from the dataset
2. However, 6 device_ids had different values in different rows
3. The file, phone_brand_device_models, had 187245 rows, with unique values as follows:
   device_id:186716
   phone_brand: 131
   device_model: 1599
4. Certain device models belonged to different brands. So, one could not be separated from another, and a key feature would be a combination of both model and brand.
    1. 1545
    2. 43
    3. 8
    4. 3

   This depicts that there were 1545 device models that belonged to 1 brand, 43 that belonged to 2, and so on.
5. Following 4, we concatenated our brand and model, and label encoded the result as 'model'. We then plotted two graphs to find out the popularity of a particular brand, and model among our users.

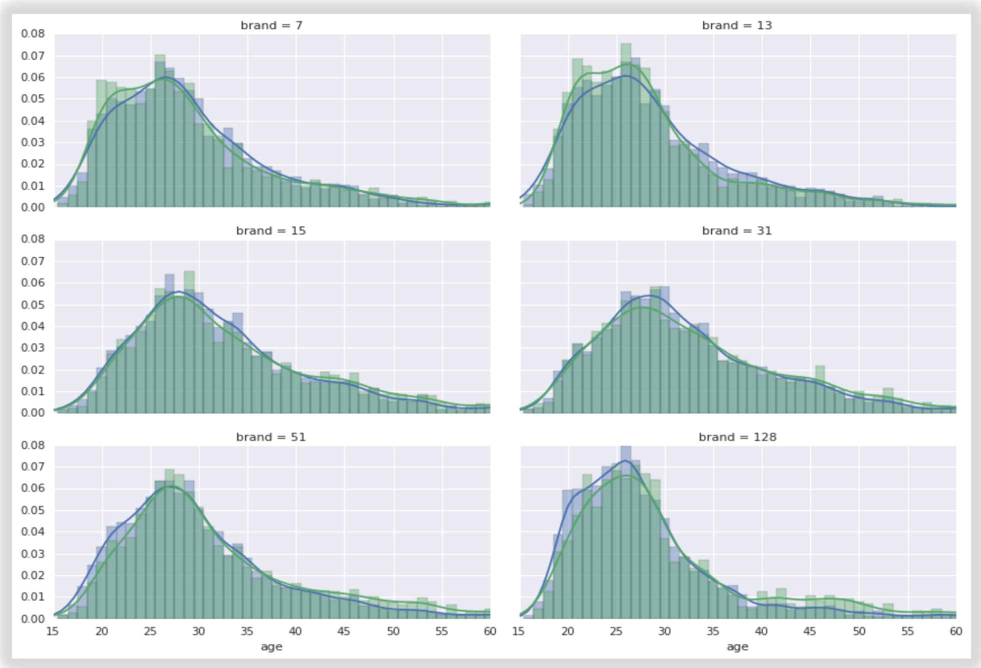


6. Below are plots of gender vs brand, and gender vs model. But, there were way too any models for us to interpret anything concrete.

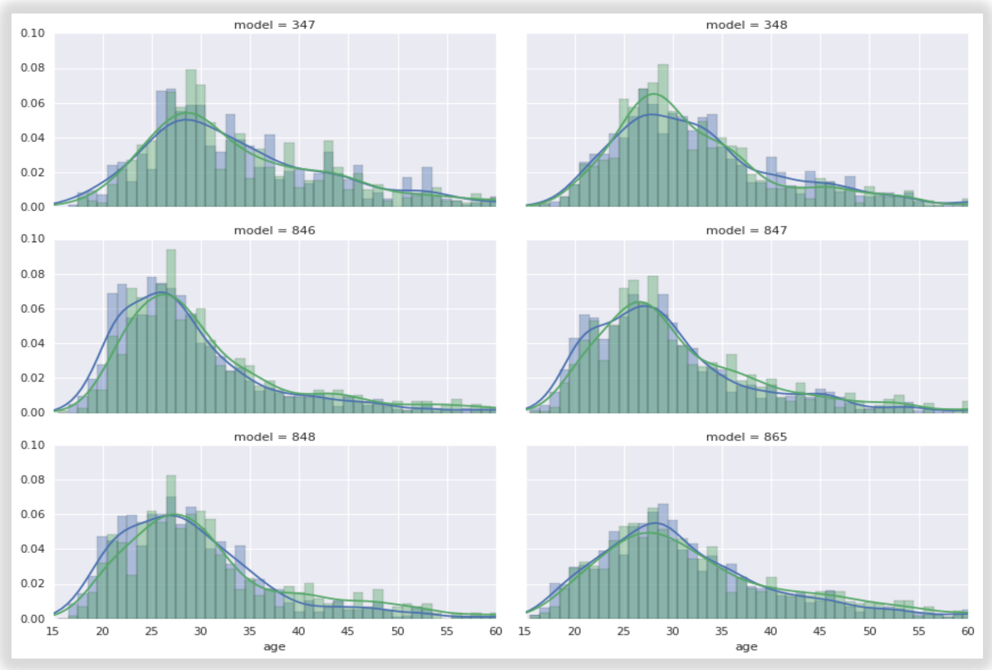




7. We plotted the age distributions by phone brand and model. These results again, did not yield any indicative results.



Age Distribution by phone_brand



Age Distribution by phone_model

FEATURE ENGINEERING

Based on the above data analysis, we narrowed down to four features:

- Phone_brand
- Device_model
- Installed Apps
- App Labels

MODEL DEVELOPMENT

Logistic Regression

The first model that we tried was logistic regression, and used the Logistic Regression classifier from scikit-learn library. The predictors for the model were the unique features that we had selected as described above.

We built eight sparse matrices, one each for the features: phone brand, device model, installed apps, and app labels and for each of the training and testing datasets. Each of these matrices had same form: the row labels were the device_ids, the unique feature values were the column labels, and the data array was all ones to mark which feature was present for which device_id.

After concatenating all the features, we created the training and test datasets to run and test our model on.

XGBoost

XGBoost is short for "Extreme Gradient Boosting", where the term "Gradient Boosting" is proposed in [18]. XGBoost is used for supervised learning problems, where we use the training data (with multiple features) x_i to predict a target variable y_i.

Implementation of XGBoost:

1. We read the 'events' table and count the number of events per device id. We then assign a unique integer to each event, and create a table 'events' with the tuple ['device_id', 'event_id', 'count'].
2. Next, we read the 'brands' table and assign a unique integer to each brand. We repeat the process of allocating the unique integers to device models. We then create a table 'pbd' with the tuple ['device_id', 'phone_brand', 'device_model'].
3. Next, we read the train data, and allot a unique integer to each of our 12 groups. We then drop the 'age' and 'gender' columns from this data and store it in a table called 'train'.

4. We also use 20% of our train data as our test data. So, we create a table called 'test' and drop all columns except 'device_id'.
5. Next, we set the parameters of XGBoost to the following:
    a. eta=0.1
    This is the learning rate of the model. Typical final values to be used: 0.01-0.2
    b. max_depth = 3
    This is the maximum depth of the tree. Used to control overfitting as higher depth will allow model to learn relations very specific to a particular sample. Typical values: 3-10
    c. subsample = 0.7
    Denotes the fraction of observations to be randomly sampled for each tree. Lower values make the algorithm more conservative and prevents overfitting but too small values might lead to under-fitting. Typical values: 0.5-1
    d. colsample_bytree = 0.7
    Denotes the fraction of columns to be randomly sampled for each tree. Typical values: 0.5-1
    These are only initial values, and we run our model for varying values of these fields.
6. Model is then trained on the training data and tested.

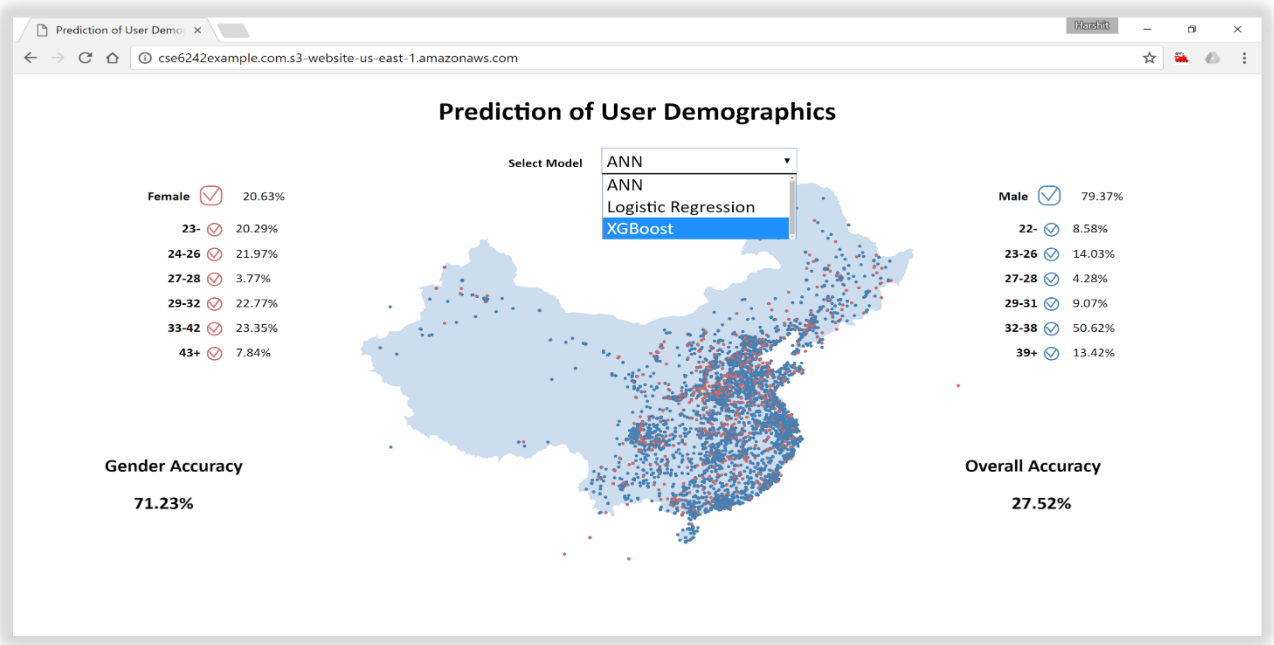
Artificial Neural Network

We developed a Neural Network model in order to better capture some probably complex patterns in our data and consequently achieve better accuracy. The data preparation was along the lines of the Logistic Regression approach. Once again, four sparse matrices were prepared, one for each of the features: phone brand, device model, installed apps, and app labels. For developing the model, keras was used, which is one of the most widely used deep learning Python libraries. Keras was used to wrap the efficient numerical computation library Theano, and allowed us to define and train our neural network models relatively easier than would have been if we would have worked directly on Theano.

Since keras does not yet have support for sparse matrices, we had to convert our sparse matrix to a numpy array. The next step was to tune the Neural Network model, for which several hyper parameter grid searching experiments were carried out, the details of which are in the 'in the experiments and evaluations section'. The final model chosen was trained for 30 epochs with a batch size of 1000 using the Adadelta optimization algorithm, and used glorot normal network weight initialization and softsign activation functions with a dropout rate of 0.6 for each of the two hidden layers. The output layer used the softmax activation layer and the model was compiled with categorical crossentropy as the loss metric.

INTERACTIVE VISUALISATION - D3.JS

In order to derive key insights from the predictions obtained on test data we have built a user interactive visualization using d3.js and HTML as depicted below:



Prediction of User Demographics

Select Model: ANN

Female	20.63%
23-	20.29%
24-26	21.97%
27-28	3.77%
29-32	22.77%
33-42	23.35%
43+	7.84%

Male	79.37%
22-	8.58%
23-26	14.03%
27-28	4.28%
29-31	9.07%
32-38	50.62%
39+	13.42%

Gender Accuracy

71.23%

Overall Accuracy

27.52%

Key features:

1. **Geospatial map:** Plotting of the devices based on the location which can be used to ascertain the density of the particular group of people geographically.
2. **Dropdown to select model**: Using this dropdown we can select a particular model which will result in visualized distribution of devices geographically and demographic wise (age and gender).
3. **Check-boxes for Gender and Age**: Different combinations can be selected which will result in different patterns for the devices.
4. **KPIs**:
    a. Accuracy based on different models (for gender and age)
    b. Percent distribution of predicted classes of the dataset.

EXPERIMENTS & EVALUATION

Analysis of Logistic Regression

To properly tune the model, we had to choose the appropriate value of the regularization constant C. Smaller values of C mean stronger regularization and its default value is 1.0. We were dealing with sparse matrices and thus had a lot of unimportant columns so we chose a value of C that resulted in stronger regularization. We decide to use a value of 0.02. Also, by default the Logistic Regression classifier solves a multiclass classification problem in a one versus rest fashion. Instead, we decided to fit a multinomial model that optimizes the multiclass logloss in order to have better accuracy.

Analysis of XGBoost

The following were our results from XGBoost. Upon setting the parameters as follows: eta = 0.1, max_depth = 3, subsample = 0.7, colsample_bytree = 0.7, we get an accuracy of 64% for gender and 17% for age-group.

Tweaking the parameters, we realized that the accuracy did change, but not by a significant number. For eta = 0.1, max_depth = 6, subsample = 0.7, colsample_bytree = 0.7, the accuracy went up to 65.2% for gender and 17.9% for age-group, but for higher values of max_depth, the model began to over fit and accuracies began to decrease.

We tried a number of such tweaks but our best results came from our first tweak, and hence we have not listed all our experimental analysis here.

Analysis of Neural Network Model

The grid-searching capability of the scikit-learning Python library was used to tune the hyper parameters of the Neural Network model.

- Tuning the Training Optimization Algorithm: Keras offers a range of state-of-the-art training optimization algorithms: SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam. Using grid search, the Adadelta algorithm was found to give the best results, and the model was accurate 17.8% of the time.
- Tuning the Network Weight Initialization: The same weight initialization method was used on each of the two hidden layers, and the grid searching was carried out among the choices of uniform, lecun_uniform, normal, zero, glorot_normal, glorot_uniform, he_normal, he_uniform. Based on the results, the glorot_normal method was used (18.1 % accuracy).
- Tuning the Neuron Activating function: The activation function controls the non-linearity of individual neurons and when to fire. Generally, the rectifier activation function is the most popular, but we ran a grid search on all the available functions (softmax, softplus, softsign, relu,

tanh, sigmoid, hard_sigmoid, and linear) and found that the softsign function gave the best results with 19% accuracy

- Tuning the Dropout Regularization: Tuning the dropout rate for regularization is necessary in order to limit overfitting and improve the model's ability to generalize. To get good results, dropout was combined with the max norm constraint. Dropout rates between 0 and 0.9 and weight constraint values between 0 and 5 were tried. The best results were obtained using a dropout rate of 0.6 and weight constraint value of 2 (19.8% accuracy)
- Tuning the Number of neurons in hidden layers: The number of neurons in both the hidden layers were provided in terms of the number of predicting variables in the dataset (say N). The number of neurons in the first layer was chosen from among values of 1.6N, 1.4N, 1.2N, and N and the number in the second layer was chosen from among 0.4N, 0.6N, 0.8N, and N. The best results were obtained by using N neurons in the first hidden layer and 0.6N neurons in the second layer. This brought up the accuracy to 22%
- Tuning the batch size and number of epochs: A variety of batch sizes and number of epochs were tried. We observed that a batch size of 1000 and 30 epochs worked best given the tradeoff between accuracy and model training time. The final accuracy was 27.52% for the age-groups and 71.23% for the gender

AMAZON WEB SERVICES

XGBoost:

The machine type chosen to be able to run this model is anaconda3-4.1.1-on-ubuntu-14.04-lts (ami-0d981a1a) with 32 CPUs and 60GB Memory. This particular instance was chosen as it is computation optimized. In comparison to other instances it has a higher ratio of vCPUs to memory boosting the performance of our model which requires distributed computing.
We have also leveraged the Elastic Block Storage available with this machine to upload our files and the scripts. This instance allows upto 30GB of data storage without requiring a dedicated storage system. Since our dataset was about 1GB we uploaded the files on to the instance through SFTP. We installed and compiled the XGBoost package to run with python that came installed with this instance and ran our models and tuned the parameters.

Logistic Regression:

As the instance was already setup for XGBoost we ran our logistic regression model on it as well and the performance was extraordinary. However, just for a logistic regression model we can choose to run the models on lower end instances which use only one vCPU.

ANN:

This particular model requires utilisation of GPU for computation. We created a ubuntu machine: DSI-Theano - ami-0e3cfa63 with GPU instance which has the deep learning packages such as Theano and Keras preconfigured to use GPU. ANN requires computational abilities of GPU and hence this particular instance was used. This particular machine also comes with a EBS for data storage and we have leveraged the same to upload our files. There was no additional setup required for this and we ran our python scripts as is. We also tried running on general purpose instances and the performance was very poor compared to this. Hence, it is suggested to use a GPU intensive instance.

Interactive Visualization:

We hosted a website on AWS with a sample domain name which connects to S3 storage where the output and the html files are stored. We created S3 buckets that interact with the website. The domain name can be later registered and the www address will be redirected to the parent location that is hosted on AWS.

CONCLUSION & FUTURE WORK

Our analysis of data and prediction of demographics was conducted using three different models, namely, Logistic Regression, Artificial Neural Networks, and XGBoost. From the exploratory data analysis, we made a number of useful conclusions and embarked upon the most relevant features of our dataset. We achieved the best accuracy from our ANN model: 71% for gender prediction and 28% for age-group prediction. We ran our models on AWS, to leverage the computability performance as our dataset is large. We procured instances that boost the performance and also hosted a website for visualizing the output. For future work, we can create an ensemble of the models we have used here to understand how this would result in better results. We could also package it as a self-service toolkit where the website, S3 and the machine are integrated to create automatic jobs to output predicted data on AWS.

PLAN OF ACTIVITIES

S.No.	Team Member	Tasks	Technology Stack
1.	Aaron D’Souza	· Exploratory Data Analysis, Feature Engineering · XGBoost	· Matplotlib, seaborn in Python 3.5 · Sklearn, pandas, numpy, scipy in Python 3.5
2.	Anushree Sinha		
3.	Harshit Amya	· Interactive Visualization, Artificial Neural Network · Artificial Neural Network, Logistic Regression	· d3.js, Sklearn, pandas, numpy, scipy in Python 3.5 · d3.js, Sklearn, keras, pandas, numpy, scipy in Python 3.5
4.	Vyom Vats		
5.	Vaishnavi Eleti	· Interactive Visualisation Prototype Cloud Integration & Performance Evaluation	· Tableau · Amazon Web Services

LITERATURE SURVEY SUMMARY

[1] How to infer user demographics such as ‘gender’ and ‘age’ based on daily mobile communication patterns like calls and messages. It is not relevant as its essence is human interactions, cross-generational and cross-gender.
[2] An iterative supervised learning framework. Helps in constructing and extracting features to develop our model. None, as it’s a descriptive paper.
[3] Prediction methodology to identify gender of YouTube users. Not relevant as it focusses more on the contrasting social and language-based genders, than the demographics as such.
[4], [5] Prediction methodology to identify demographics based on list of apps installed on user’s phone. Methods mentioned to reduce the dimensionality of the data are relevant to our project. The classification method mentioned is effective only for diverse set of apps installed which would not be the case in our model.
[6] Examination of the users of Twitter by demographics in the US. The limitations of this paper where the gender could be misinterpreted by username would be covered.
[7] Prediction methodology for demographics using user behavior and environments. App log as well as location data will be included in our model to make relevant predictions.
[8] Accuracy of ML algorithms are greatly improved using extracted features vs raw data. Methods mentioned can be used to extract meaningful and measurable predictors from the raw data.
[9] Methodological review of multiple machine learning models and comparisons among them. It helps in identifying suitable models for our prediction.
[10] Introduction to the basics, design, and applications of artificial neural network models. It helps in learning about ANNs. Overfitting of data will be taken care of in our model.
[11] Introduces TV-ANN model and compares the accuracy to traditional BP-ANN models and Linear Discriminant Analysis. Not very much relevant to our project since it focuses on binary classification problems.
[12] Explains approximating function for tree boosting. The function approximation is done over all the parameters, but we could start with a subset and increment the function for faster computation.
[13], [14] Explains XGBoost as a reliable system to scale up tree boosting algorithms. It helps as it is optimized for parallel tree construction. Only the weighted quantile approach for finding the approximate algorithm is described in the one of the paper.
[15] Discusses about diversification of visualization using D3. Useful for constructing our interactive visualization.
[16] Employs users’ web browsing behavior to predict their age and gender. It helps in feature extraction for our problem.
[17] Analysis of two data visualization strategies (d3.js and Tableau). It helps in choosing the appropriate interactive visualization.

REFERENCES

[1] Dong, Yuxiao, et al. "Inferring user demographics and social strategies in mobile social networks." *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014.

[2] Mo, Kaixiang, et al. "Report of task 3: your phone understands you." *Nokia mobile data challenge 2012 workshop, Newcastle, UK*. 2012.

[3] Filippova, Katja. "User demographics and language in an implicit social network." *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, 2012.

[4] Malmi, Eric, and Ingmar Weber. "You Are What Apps You Use: Demographic Prediction Based on User's Apps." *arXiv preprint arXiv:1603.00059* (2016).

[5] Seneviratne, Suranga, et al. "Predicting user traits from a snapshot of apps installed on a smartphone." *ACM SIGMOBILE Mobile Computing and Communications Review* 18.2 (2014): 1-8.

[6] Mislove, Alan, et al. "Understanding the Demographics of Twitter Users."*ICWSM* 11 (2011): 5th.

[7] Ying, Josh Jia-Ching, et al. "Demographic prediction based on user's mobile behaviors." *Mobile Data Challenge* (2012).

[8] Storcheus, Dmitry, Afshin Rostamizadeh, and Sanjiv Kumar. "A Survey of Modern Questions and Challenges in Feature Extraction." *Proceedings of The 1st International Workshop on "Feature Extraction: Modern Questions and Challenges", NIPS*. 2015.

[9] Dreiseitl, Stephan, and Lucila Ohno-Machado. "Logistic regression and artificial neural network classification models: a methodology review."*Journal of biomedical informatics* 35.5 (2002): 352-359.

[10] Basheer, I. A., and M. Hajmeer. "Artificial neural networks: fundamentals, computing, design, and application." *Journal of microbiological methods* 43.1 (2000): 3-31.

[11] Pendharkar, Parag C. "A threshold-varying artificial neural network approach for classification and its application to bankruptcy prediction problem."*Computers & Operations Research* 32.10 (2005): 2561-2582.

[12] Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." *Annals of statistics* (2001): 1189-1232.

[13] Chen, Tianqi, and Carlos Guestrin. "XGBoost: Reliable Large-scale Tree Boosting System."

[14] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." *arXiv preprint arXiv:1603.02754* (2016).

[15] Bostock, Michael, Vadim Ogievetsky, and Jeffrey Heer. "D³ data-driven documents." *IEEE transactions on visualization and computer graphics*17.12 (2011): 2301-2309.

[16] Hu, Jian, et al. "Demographic prediction based on user's browsing behavior."*Proceedings of the 16th international conference on World Wide Web*. ACM, 2007.

[17] Jaehoon Lee, et al. "A Comparative Analysis of Clinical Data Visualization Development Strategies: Build vs. Buy."

[18] Friedmann, Jerome H., "Greedy Function Approximation: A Gradient Booster Machine." IMS 1999 Reitz Lecture