PREDICTING USER DEMOGRAPHICS

THINKFUL FINAL CAPSTONE

KEVIN LAM

BACKGROUND

- Found on Kaggle: https://www.kaggle.com/c/talkingdata-mobile-user-demographics
- TalkingData, China's largest third-party mobile data platform wanted to leverage the behavioral data of its users in order to help its clients better understand and interact with their respective audiences.

WHY CARE ABOUT IDENTIFYING USER DEMOGRAPHICS?

- Being able to accurately identify the demographic of one's audience can help the company better personalize their experience for their users.
- Can accurately identify what sort of market the client has and what sort of method the client wants to employ in order to gain more users, or retain their current userbase.

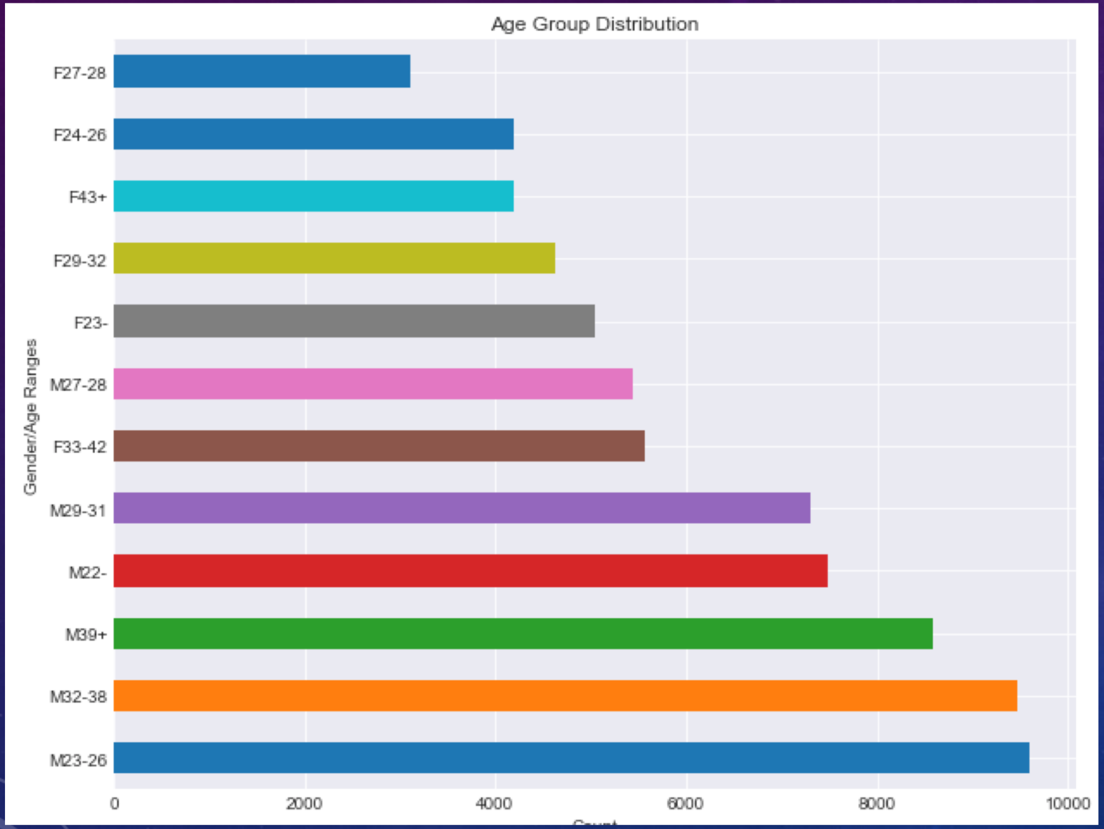
GOAL

- Find the best model to accurately identify the 12 different groups in the target portion of the data set.
- Best will be defined by accuracy and computational time of the model.

DATA EXPLORATION PRELUDE

- 6 different data sets that possessed different attributes that I had to stitch together.
  - gender_age_train.csv - the main data set. Possessed the target column 'group'
  - events.csv - when a user uses an app that has an ad that TalkingData tracks, the event gets logged in this data. Each event has an event id, location (lat/long), timestamp.
  - app_events.csv - the event corresponds to a list of apps in app_events and whether that app was active or in the background.
  - 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 (ex. games, reference, social media)
  - phone_brand_device_model.csv - device ids, brand, and models

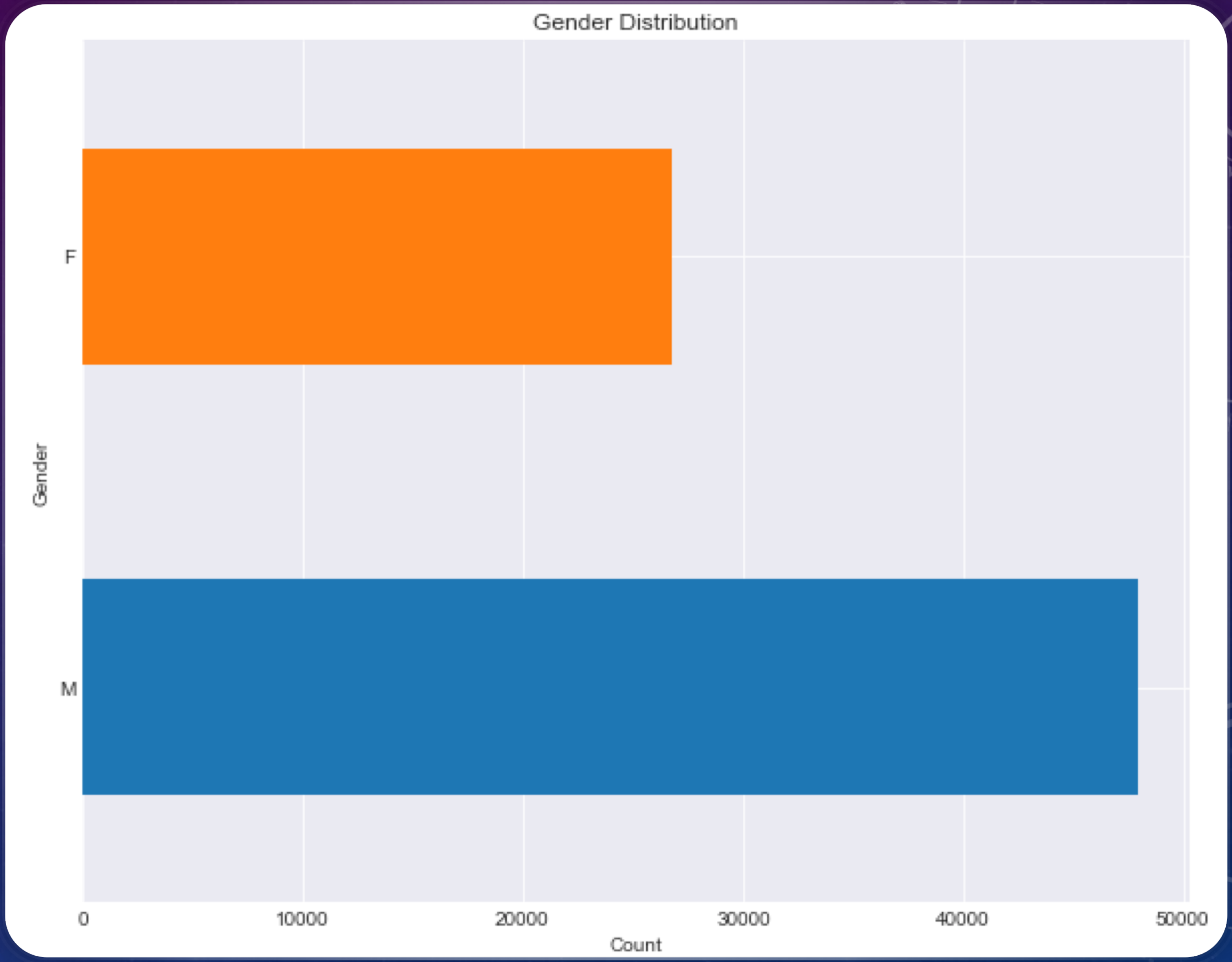
DATA EXPLORATION



- Looking at the target data, we can see that the majority of the user base are males.

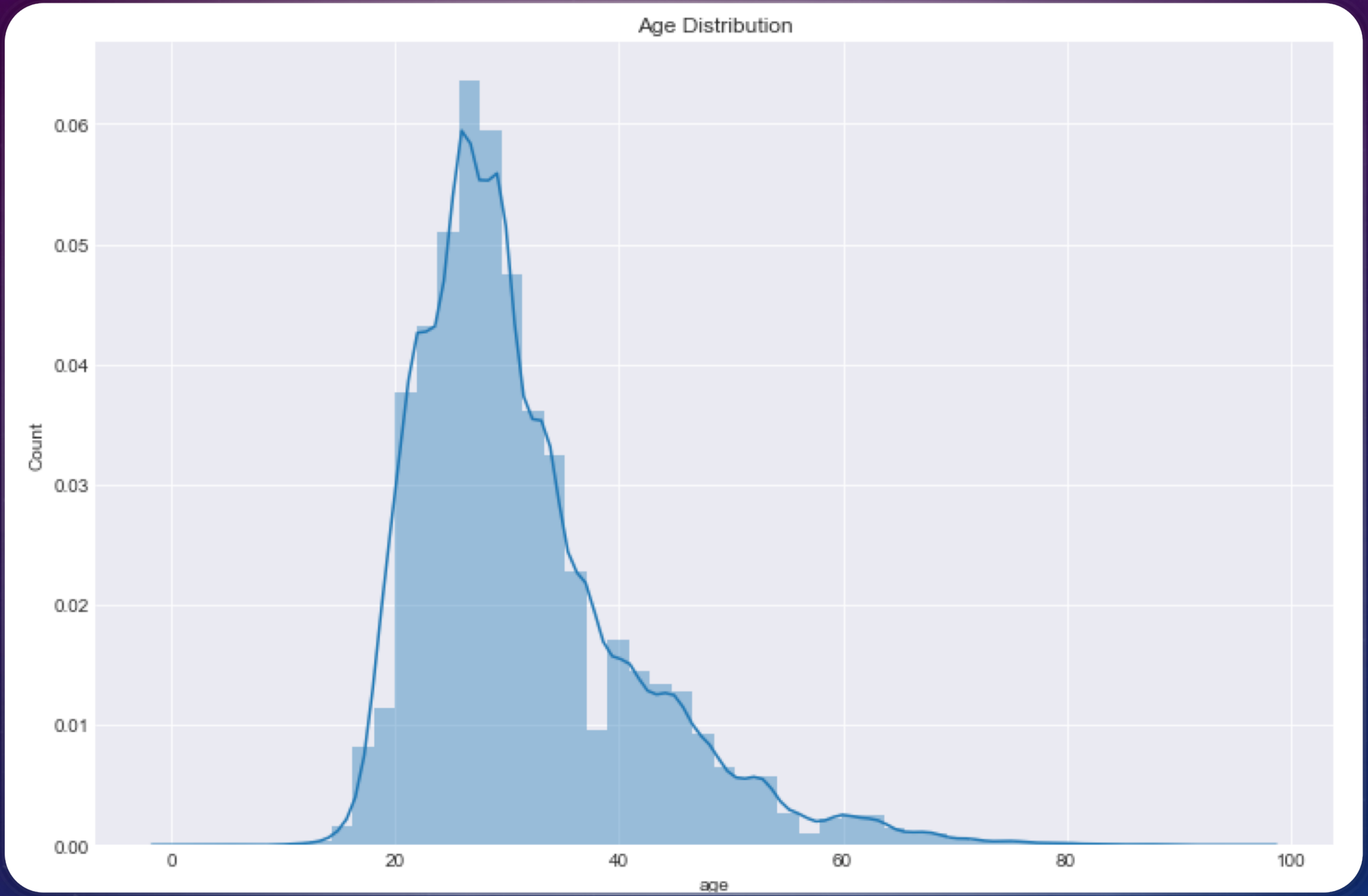
- The majority are young adult males in their early 20s.
- We can see that there are less females in this data set but it could be that less female users use apps that TalkingData tracks.

DATA EXPLORATION



- Here, we see that there are more guys than girls in this data set.
- There are almost double amount of males than amount of females.

DATA EXPLORATION



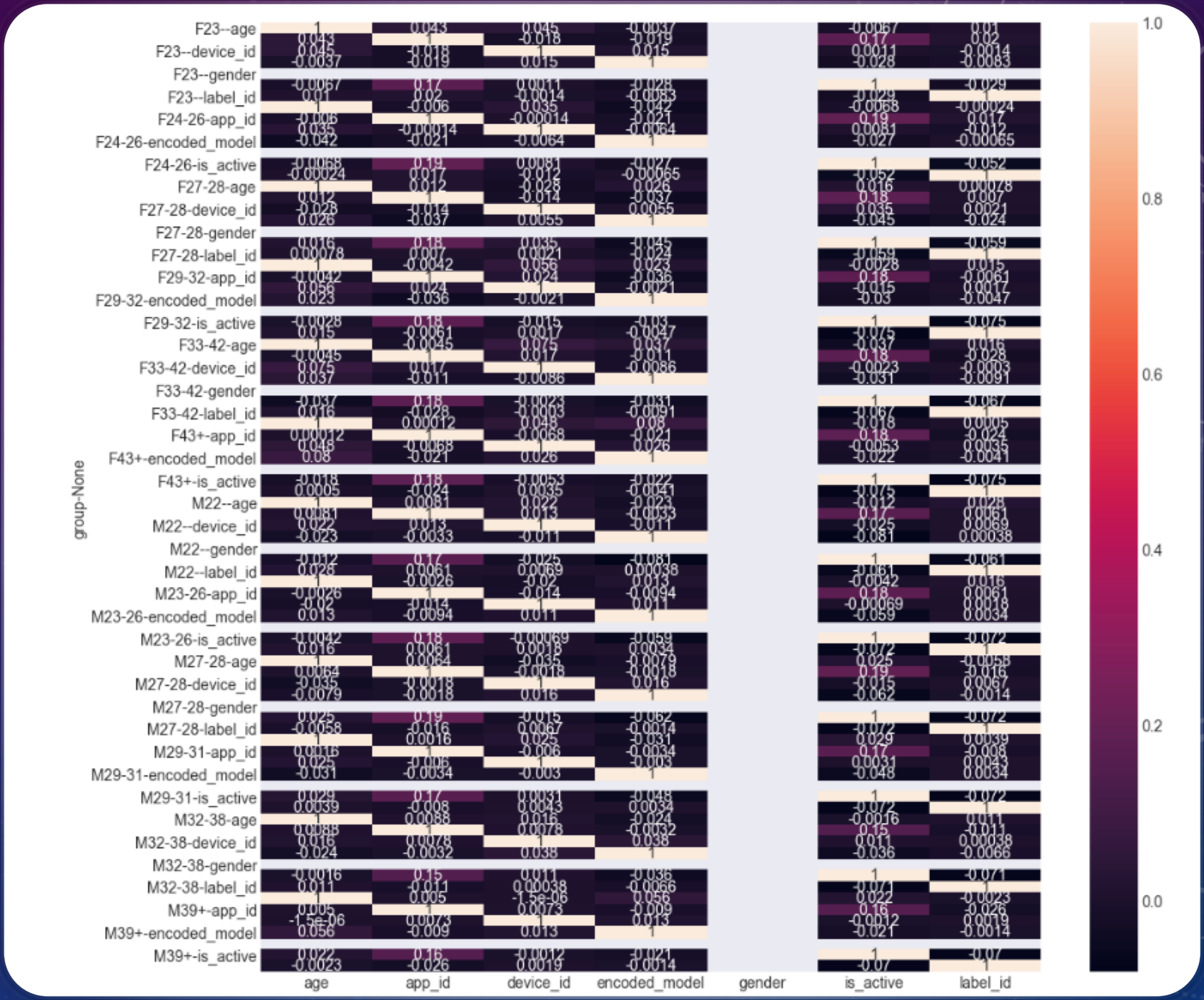
- We can see that the majority of users are young adults from mid 20s to 30s.
- There are less older adult users and it could be for a multitude of reasons.

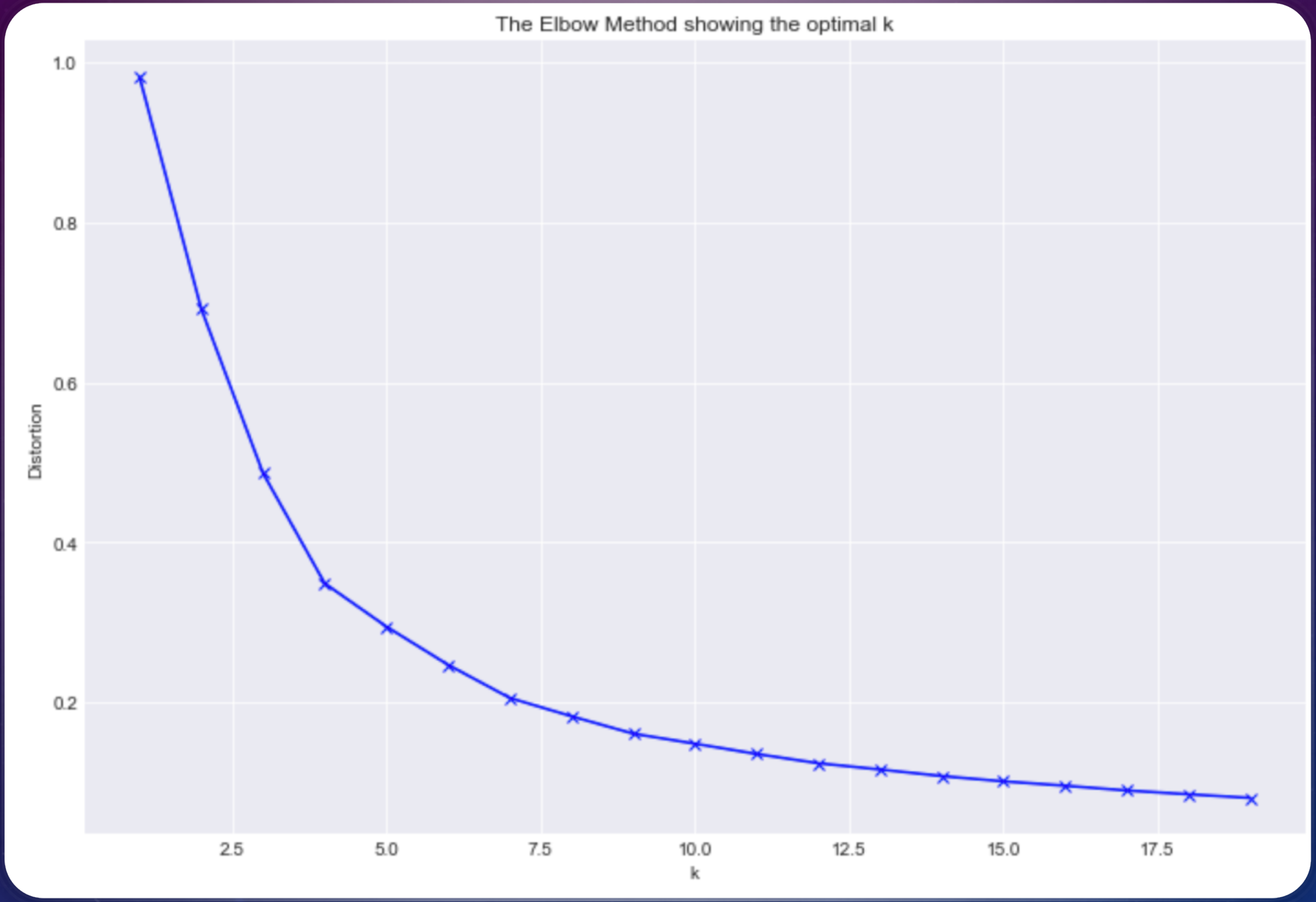
DATA PRE-PROCESSING

- 6 separate csv files.
- Merge files based on ‘app_id’ and/or ‘device_id’ columns.
- Created dummies for the label_id (label of app) in order to create more features.
- After creating the dummies and merging the files, the data frame has (7,034,111, 493)
- Ended up down sampling the data set into 50,000 rows for each class of data in the group to fix the class imbalance (12 classes in the target data set)

DATA TUNE UP

- Made a heat map to see if there was any multi collinearity present.
- Faced another problem of having low to negative multi collinearity instead.
- Performed a train/test split with 25% holdout.

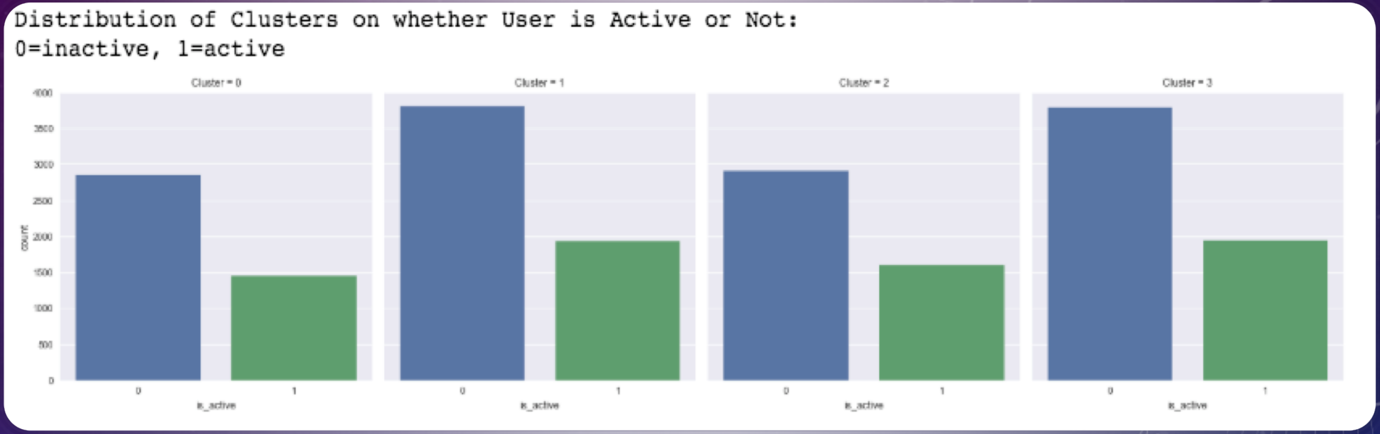




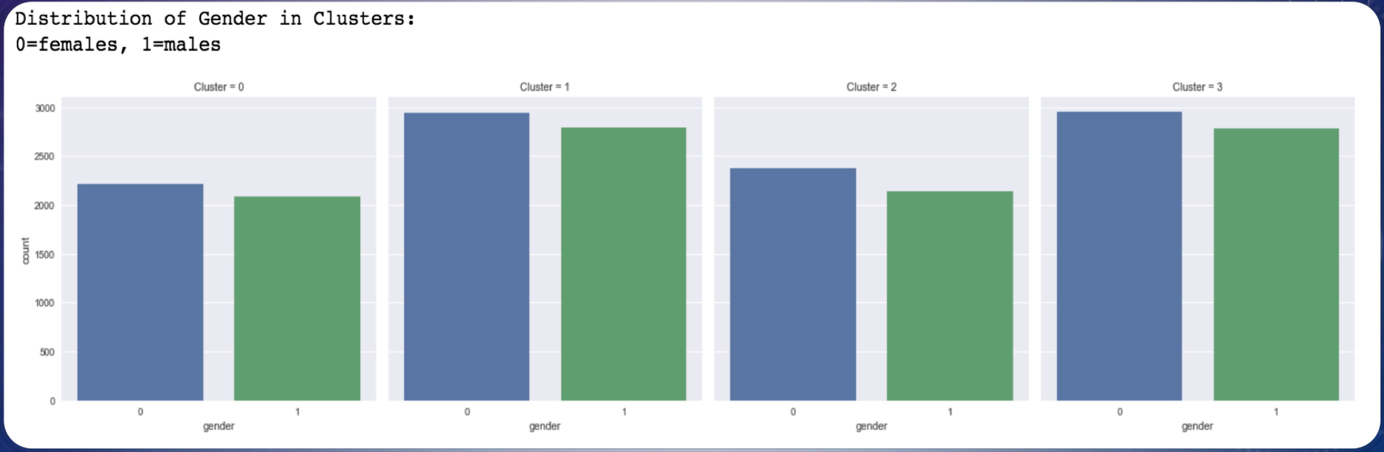
K-MEANS

- Originally thought that 12 would be the optimal amount of clusters but the elbow method shows that approximately 4 clusters is the optimal amount.
- Used k-means clustering to learn more about the data, but it was just an even distribution of data between the 4 clusters.

K MEANS CLUSTER DISTRIBUTION



- As we can see here, the clusters look quite similar, and the data points are almost evenly distributed throughout the clusters without any one cluster having an overwhelming majority.



PERFORMANCE RUBRIC

- Best Parameters: Parameters that allowed for the best classifying score.
- Average 5 Fold Cross Validation Score - resampling the dataset into 5 equal sized samples and running through the models again.
- Classification Report (Average scores between the different classes)
  - Precision – (TP/(TP+FP)) Ability of the classifier not to label as positive a sample that is negative.
  - Recall – (TP/(TP+FN)) Ability to find all positive samples.
  - F1- a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0.

MODEL 1: LOGISTIC REGRESSION WITH PCA

- Jumped right into PCA for this model due to the fact that I had low/negative multicollinearity between my data.
- GridSearchCV parameters: C:10,000, max_iter:200, multi_class: multinomial, solver:lbfgs
- Used L-BFGS because it's a more computationally efficient form of Newton's method. Faster due to only using a few vectors to reconstruct the matrix.
- Parameters: C:1000, max_iter: 25, multi_class: multinomial, solver:lbfgs
- Average of 92% accuracy.
- Avg. Precision: 92% || Avg. Recall: 92% || Avg. F1: 92%

RANDOM FOREST WITH SELECT K BEST

- Disclaimer: I had mistaken k for number of classes instead of features so I didn't give this enough k values.
- GridSearchCV parameters: k:12, max_depth:16, n_estimators:1500
- Average of 56% accuracy.
- Avg. Precision: 56 % || Avg. Recall: 56 % || Avg. F1: 53%

RANDOM FOREST (PCA)

- Had more confidence in random forest without feature selection due to it being an ensemble model.
- Best Parameters: n_estimators:1000, max_features:8, max_depth:8
- Avg. Accuracy: 94.03%
- Avg. Precision: 93% || Avg. Recall: 93% || Avg. F1: 93%
- PCA Best Parameters: n_estimators: 800, max_features:6, max_depth:10
- PCA Avg. Accuracy: 91.71%
- PCA Avg. Precision: 90% || PCA Avg. Recall: 89% || PCA Avg. F1: 89%

X GRADIENT BOOSTED MODEL WITH PCA

- Model was extremely overfitted even after instilling high regularization parameters and took a long time to run.
- Long computational time was also why I switched to XGB since it runs faster and reaches convergence faster than regular gradient boosting due to XGB employing stochastic gradient descent.
- Employed PCA and used 100,000 rows of data.
- Parameters: max_depth:5
- Average of 93.29% accuracy.
- Avg. Precision: 90% || Avg. Recall: 90% || Avg. F1: 90%

NEURAL NETWORK: MULTILAYER PERCEPTRON MODEL WITH PCA

- Used a sequential model based on TensorFlow and Keras
- Adam optimizer, elu as the activation function.
- 8 layers, used batch normalization, 1250 batch size, 20 epochs.
- Test Accuracy: 98.33%

CONVOLUTIONAL NEURAL NETWORK WITH PCA

- Normally, CNN is used for image classification, but I wanted to see if I could use CNN on this 1D data set and see if I could obtain good results.
- Had to increase the dimensional size of my training data, vectorize my target data before CNN would run.
- 3 convolutional layers, 6 dense layers, batch normalization, 1250 batch size, 10 epochs.
- Test Accuracy: 70.84%

MOVING FORWARD

- Originally, I had a lot of time to do the project so I didn’t use Google Cloud computing but that could be worth trying out.
- Try adding different models to see if there are any other strong performing models with a decent run time.
- Try using another optimizer for my neural network.

CONCLUSION

- In terms of convenience in implementation, **Random Forest** was the easiest model to run that performed the strongest with minimal parameter tuning. One would have to be careful of overfitting due to the volume of the data though.
- Coming in second, **Neural Network with PCA** was a super strong performer that took only half the amount of time as the time needed for random forest.

QUESTIONS?

- Find my notebook at: https://github.com/lamkg/Thinkful-Final-Capstone/blob/master/Final%20Capstone.ipynb