Gender Classification from Twitter Data

Emi Kong

Ishana Narayanan

Jacqueline Zhang

I. Abstract

In this project, we used Twitter data to predict the gender of Twitter users. Our gender classification was multiclass with a user being labeled male, female or brand. The features we considered in our models included term frequencies, punctuation counts, emoji appearance, account years, number of total tweets, retweets, and favorites. We utilized many Feature Transformers from the ML Features package and built pipelines to clean and preprocess the dataset. Our models included Multinomial Logistic Regression, Random Forest, Naive Bayes, and KMeans. Using compute time and confusion matrix metrics to evaluate each model, we found that our Benchmark model of Multinomial Logistic Regression with term frequency for each tweet as the input parameter has the best accuracy and compute time. Therefore, our Benchmark model is our champion model. However, the accuracy is not impressive and thus we cannot fully predict the gender of Twitter users just from tweet and certain account information.

II. Data and Methods

Background of Data

Originally utilized to train a CrowdFlower AI gender predictor, our data set from Kaggle contains information taken from Twitter that aimed to predict Twitter user gender. It has 24,230 rows and 26 columns containing details on user gender, account profile information, and a random tweet. Since the dataset is of type 'dataframe', we perform most analyses, preprocessing, and modelling with the ML package, with the exception of some usage of RDDs and the MLlib package.

Data Cleaning

The data frame was cleaned by keeping columns deemed relevant in our models and overall purpose of predicting user gender and filtering rows containing obscure or null values. We decided to remove columns with faulty values such as location, profile images, and link colors, and columns relating to the original CrowdFlower AI gender predictor such as trusted judgments and gold standard.

We chose to keep columns **gender** and **text**, as well as other user account column information. We believed the following profile-related columns contained information that would help predict user gender:

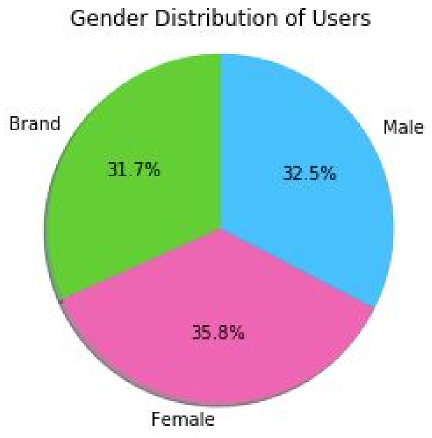
- **fav_number**: number of tweets the user has favorited
- **retweet_count**: number of times the user has retweeted or been retweeted
- **tweet_count**: number of tweets the user has posted

- **created**: date and time when the profile was created
- **tweet_created**: when the tweet in the 'text' column was created
  (All tweets were created on 10/26/2015)
- **account_years:** number of years user has had the account
  (We calculated this value by extracting the year the account was created from **created** and subtracting it from 15, the year of **tweet_created**. We then added this value to the data frame as the column **account_years**.)

With obscure values in the **gender** column, like "unknown", "London", or "6.5873E+17", the tweets dataframe was filtered to only include rows with gender values of 'brand', 'female', or 'male'. Starting with 24,230 rows, the dataset now had a total of 18,836 rows once **gender** was filtered. Discovering that the **text** column contained 1,088 null values, dataframe was again filtered to only include rows with non-empty strings in **text**, leaving a final 17,748 rows in our data frame.

Exploratory Data Analysis

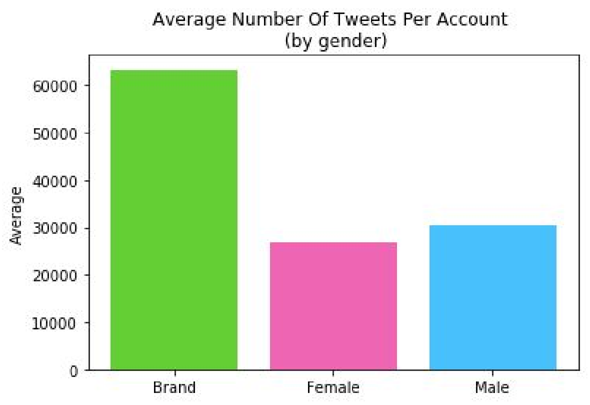
After choosing important variables for our model, we were curious to see the gender distribution within the dataset, find gender biases among user account columns, and examine the distribution of emojis, punctuation, and words within the **text** column. Throughout the analysis and visualizations, each gender is represented by a specific color, with **brand** as **green**, **female** as **pink**, and **male** as **blue**.

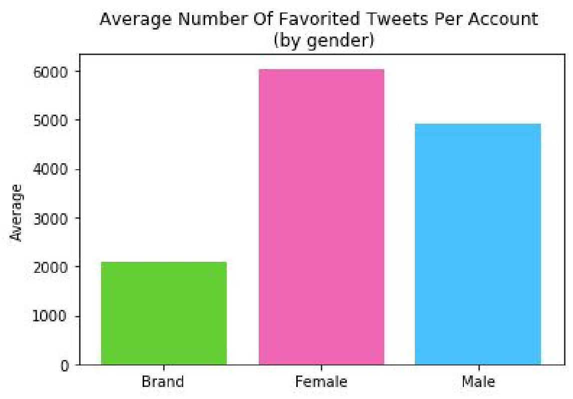


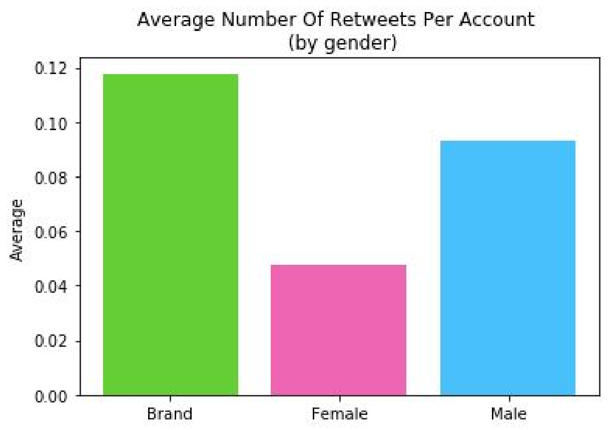
There appeared to be almost equal percentages of genders within our entire data frame. Specifically, there were 6,354 female users, 5,776 male users, and 5,618 brand users.

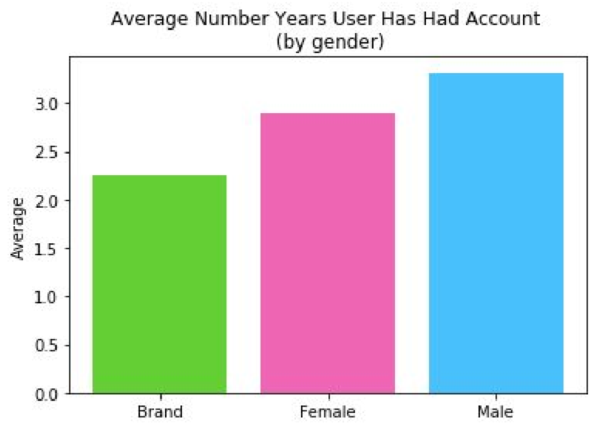
User Account Profile

Curious to see if there were gender biases among user profile data, we calculated the averages of columns **tweet_count, fav_number, retweet_count,** and **account_years** by gender.









Brand had the highest averages for number of tweets and retweets, which is consistent with a brand's purpose of using Twitter for marketing. Since a different gender had the highest average in each column, there was an overall gender bias among these four user profile characteristics.

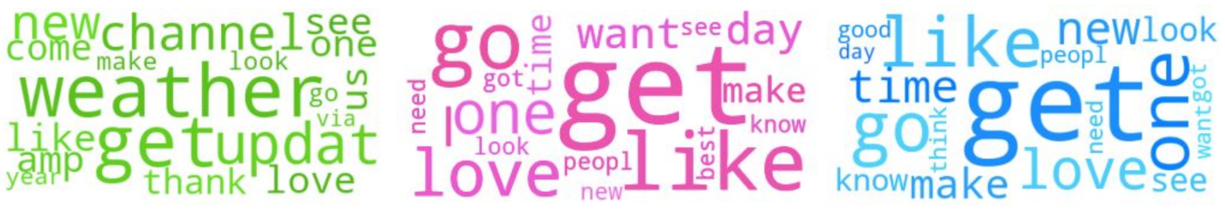
Punctuation

Within the **text** column, we were curious to see differences in punctuation usage between users. Summing up the number of punctuation marks used in each tweet, we were able to get counts for punctuation.

Emojis

We decided to explore emoji usage within our data frame. While importing, not all unicode characters transferred properly. With most emojis coming in as replacement characters, there was not a way to differentiate between each emoji. Instead, we decided to test for the presence of any emoji within a tweet by setting a boolean value (i.e. 'True' means there was at least one emoji in the tweet, 'False' means there were no emojis).

Most Used Words
We found the twenty most used words for each gender.



Overall, there were some overlapping words among genders, including "get", "like", "one", and "see". Brands seemed to use words the other two genders did not use as frequently, like "weather", "update", and "channel".

Data Preprocessing and Pipelines

To prepare and transform our dataset for modelling, we utilized many Feature Transformers from the ML Features package and built pipelines to execute changes. We chose to use pipelines for data preprocessing and feature extraction since they allow us to easily keep track of all our transformations.

Pipeline 1

In our first pipeline, we used the Imputer() and StringIndexer() functions to update the **tweet_count** and **gender** columns, respectively. After noticing during EDA that **tweet_count** had less than 10% of its values as null, we replaced the null values with the median of **tweet_count**. We chose to replace null values with the median instead of the mean to keep all values as integer type. Similarly, we wanted our **gender** column to be of numerical type instead of categorical strings and, thus, StringIndexer() was applied. This resulted in "female" being coded as 0.0, "male" as 1.0, and "brand" as 2.0.

Most of the transformations were made on our **text** column. For this feature, we aimed to change the original tweets into vectors of word frequencies. After using a SQL function to lowercase the text, we applied Tokenizer() and StopWordsRemover() in our first pipeline to tokenize and remove stop words (words that appear frequently and do not carry as much meaning). Since the ML Feature library does not support stemming, we used the SnowballStemmer() function from the NLTK package to stem the list of words produced after the first pipeline. Stemming, similar to lemmatization, allows us to reduce the words to their base word.

Since our EDA revealed uneven gender distributions for **punc**, **emojis**, and **account_years,** we believe they could be helpful in our model. Before applying our second pipeline, we updated our dataframe by adding the following columns:

- **stemmed**: stemmed words
- **punc**: punctuation counts
- **emojis**: emoji appearance
- **account_years**: number of years user has had account

Pipeline 2
In our second pipeline, we created new columns:

- **tf**: term frequencies of **stemmed** using HashingTF()
- **bigrams**: bigrams of **stemmed** using NGram()
- **btf**: term frequencies of **bigrams** using HashingTF()
- **features**: all relevant features packaged into one column using VectorAssembler()
    - This **features** column includes **tf**, **btf**, **punc, emojis, tweet_count_new, fav_number, retweet_count,** and **account_years**
- **scaledTF:** StandardScaler() applied on **tf**
- **scaledFeatures**: StandardScaler() applied on **features**

We chose to include term frequencies for single words and term frequencies for bigrams because they both contain valuable, unique information about predicting gender. Term frequencies for single words provide information about the gender themselves whereas term frequencies for bigrams provide information about gendered speech.

III. Results

<u>Models and Results</u>

Before model selection, we split our data into train and test sets with 70% of our data allocated for training and 30% for testing. Since this is a supervised machine learning problem involving classification, we decided to focus on a handful of models: Multinomial Logistic Regression, Random Forest, and Naive Bayesian.

Benchmark Model
In the benchmark model, we only considered term frequencies of individual words within each tweet (**scaledTF**) as our input variable. We applied the LogisticRegression() function to fit and transform the data, then leveraged MulticlassClassificationEvaluator() to obtain model accuracy. Our benchmark model returned an accuracy of 0.488.

Supervised Modeling
After creating our benchmark model, we developed three models which considered the additional text and account features detailed above (**scaledFeatures**). We trained another

Multinomial Logistic Regression model with these inputs and used MulticlassClassificationEvaluator() to obtain model accuracy of 0.486.

We also considered Random Forest as our primary tree ensemble method. We applied RandomForestClassifier() to **scaledFeatures** and again used MulticlassClassificationEvaluator() to obtain a model accuracy of 0.423, significantly lower than the benchmark model. Lastly, we constructed a Naive Bayes model on the training data using the NaiveBayes() function and used MulticlassClassificationEvaluator() to obtain a model accuracy of 0.461.

In addition to constructing these models, we used the F1 score as a measure of sensitivity to changes in hyperparameter. In general, changing the hyperparameters did not have a large effect on the accuracy of the models and therefore we concluded that our models have low sensitivity. We also applied paramGrid and cross validation to tune hyperparameters and validate the model. The cross validation did slightly improve the average accuracy of each model.

Model Comparison
To determine the champion model, we considered various confusion matrix metrics and model compute time.

Accuracy computes the ratio of correctly predicted observations to the total observations. From the table below, we see that the model with the best accuracy is our benchmark model with an accuracy of 0.488. Precision is the ratio of correctly predicted positive observations to the total credited positive observations. A sign of a good model is a high precision score as it indicates a low false positive rate. Naive Bayes has the best precision of 0.477, followed closely by the Benchmark model with a score of 0.469. Lastly, we calculated the F1 score, which is a weighted measure of precision and recall that takes into account both false positives and false negatives. Again, the model with the best F1 score is the benchmark model with a score of 0.488, followed closely by Multinomial Regression.

	Benchmark	**Multinomial**	**Random Forest**	**Naive Bayes**
Accuracy	0.488	0.486	0.423	0.461
F1 Score	0.488	0.482	0.313	0.456
Precision	0.469	0.450	0.384	0.477
Recall	0.589	0.661	0.988	0.447

In general, Naive Bayes had the fastest compute time, followed closely by the benchmark model, then Multinomial, and lastly Random Forest. This ordering makes sense when considering each underlying algorithm and model inputs. Naive Bayes is known to be computationally efficient, as it only needs to compute class and conditional probabilities. Furthermore, the Benchmark model only contained term frequency for each tweet, so the input spanned a smaller vector space in comparison to the other models. Random Forest had the longest computational time, which follows from the algorithm having to compute each split, construct, and bag multiple trees.

Given this information, the champion model was the Benchmark model, which is a Multinomial Logistic Regression on just the term frequency in the text column. We cross-validated our best models and found that this one still has the highest accuracy of 0.500.

Unsupervised Modeling
To validate the findings above, we constructed a K-means model to gain insight into the underlying structure of the data. Since we have three distinct classifications, we trained a K-means model to compute three cluster means and utilized the ClusteringEvaluator() class to find the silhouette squared distance between the clusters. From this, we obtained a distance of 0.077, indicating that within cluster distance was not significantly smaller than the between cluster distance. The lack of three distinct clusters as a result of the K-means model might suggest the additional inputs we provided were not significant in predicting gender.

IV. Conclusion

Even though we have a champion model, its overall accuracy of 0.500 is an indication that the model was not a good fit for the data. In general, it appeared that the additional information we were including in the model such as punctuation, emoji, and bigram term frequency did not add additional information in predicting gender. Perhaps, these models were overfitting, and a dimensionality reduction technique such as PCA or SVD should be applied to the sparse vectors produced for term and bigram frequency counts. Therefore, we conclude that we cannot predict gender solely from tweets and specific account profile data.

In the future, we hope to perform dimensionality reduction techniques on the sparse vectors created from HashingTF, as these large vectors may be causing the models to overfit the data.