Rayce Ramsay, Kenta Ploch, Justin Won

Colab Link:
https://colab.research.google.com/drive/1vWqQXzo6ue5cqKD1u5Xi-9essOg_5RU9?authuser=0#scrollTo=UrBguEAbtw-R

Breakdown:

- Kenta
  - Part I: Code & Written Response
  - Part II: Report section *(II) Describe the Data*
  - Part III: Slides for section *(II) Describe the Data*
- Justin
  - Part I: Bonus Question
  - Part II: Report sections *(I) Problem description and motivation* and *(IV) Describe your machine learning model*
  - Part III: Slides for section *(I) Problem description and motivation* and *(IV) Describe your machine-learning model*
- Rayce
  - Part II:
    - Code/Analysis (data extraction, EDA, preprocessing, modelling, performance evaluation)
    - Report sections *(III) Exploratory data analysis* and *(V) Results and Conclusions*
    - Edited/polished report for conceptual, structural, and grammatical correctness and clarity
  - Part III: Slides for sections *(III) Exploratory data analysis* and *(V) Results and Conclusions*

Answers to Part I: Sentiment Analysis with a Twitter Dataset (20pts)

A) *(1 pt) Consider the training data. What is the balance between the three classes? In other words, what proportion of the observations (in the training set) belong to each class?*

Answer: After removing the na and string sentiment value columns, the proportions were as follows:

Negative tweets: 0.374159

Neutral Tweets: 0.187407

Positive Tweets: 0.438434

B) (*1 pt) Tokenize the tweets. In other words, for each observation, convert the tweet from a single string of running text into a list of individual tokens (possibly with punctuation), splitting on whitespace. The result should be that each observation (tweet) is a list of individual tokens.*

Answer: (on the collab notebook)

***C**) (1 pt) Using a regular expression, remove any URL tokens from each of the observations.*

***Hint:** In this dataset, all such tokens begin with "http".*

Answer: (on the collab notebook)

D) *(2 pts) Remove all punctuation (,.?!;:'") and special characters(@, #, +, &, =, $, etc). Also, convert all tokens to lowercase only. Can you think of a scenario when you might want to keep some forms of punctuation?*

Answer: I would like to keep the punctuation when it determines whether a message is positive or negative. For example, "I want that." and "I want that!!!!!" become the same when you remove the punctuation but the emotion that the sentence has is different. "I want that." would have a neutral tone, which is implied by the period, while "I want that!!!!!" can be seen as someone shouting in anger, thus conveying a negative emotion. As we are trying to determine if something is negative or positive, removing this emotion will not allow our algorithm to classify the tweets with these emotions.

***E**) (1pt) Now stem your tokens. This will have the effect of converting similar word forms into identical tokens (e.g. run, runs, running → run). Please specify which stemmer you use.*

***Note**: There are several different stemmers available through nltk and Scikit-learn. I recommend the Porter stemmer, but you may use a different one if you wish.*

Answer: I have used the Porter stemmer to convert similar word forms into identical tokens.

F) *(1pt) Lastly, remove stopwords. Using the english stopwords list from nltk, remove these common words from your observations. This list is very long (I think almost 200 words), so remove only the first 100 stopwords in the list.*

Answer: (on collab)

G) *(2 pts) Now convert your lists of words into vectors of word counts. You may find Scikit-learn's CountVectorizer useful here. What is the length of your vocabulary?*

Hint: *The matrix of counts will be D × V , where D is the number of documents (tweets), and V is the number of features (word counts).*

Answer: The vocabulary length was 1000, and this was set through the "max_features" parameter for the count vectorizer, where I set max_features = 1000.

Now, I have decided to set the number of features as 1000 for several reasons. Firstly, in the list there are, most likely, a lot of words that occur only once, and including these models will make it too big and time-consuming to run. Also, having a large number of features takes up space as well. So to reduce the space dimension, I capped it at 1000.

Another reason is to avoid overfitting our predictive model. If we include all of the features, then our model will perform very well for the train_data but not so much on test and actual data. Hence we should not include all of the features in our vector. However, making our vector too small would lead to the loss of the predictive power of our algorithm. So I had to find a sweet spot that will not overfit but also not make the model perform "bad" and I thought that that would be at 1000 features.

H*) (4 pts) Recall the definition of the Naive Bayes model. If each document (tweet) is a collection of words (w1, · · · , wN) belonging to class Ck (k = 0, 1, 2), then the Naive Bayes approach models the probability of each tweet belonging to class k:*

$$\begin{aligned} P(C_k|w_1, \cdots, w_N) &\propto P(w_1, \cdots, w_N|C_k)P(C_k) \\ &= P(C_k)\Pi_{i=1}^{N} P(w_i|C_k) \end{aligned}$$

The last equality follows from our "naive" assumption that words are conditionally independent given class. The probabilities are estimated using the frequencies of words within each class (bag of words), and we assign the class label according to which of the 3 posterior class probabilities (P(Ck|w1,⋯,wN)) is the highest.

Fit a Naive Bayes model to your data. Report the training and test error of the model. Use accuracy as the error metric. Also, report the 5 most probable words in each class, along with their counts. You might find Scikit-learn's MultinomialNB() transformer useful. Use Laplace smoothing to prevent probabilities of zero.

Answer: The training error rate was approximately 0.32 while the test error rate was approximately 0.56. This was obtained by subtracting the accuracy from 1.

Observe that in the code in collab, Laplace smoothing is used because the default parameters for MultinomialNB set the smoothing parameter equal to 1.0. So even if I do not pass by any arguments in MultinomialNB(), laplace smoothing will occur.

The 5 most probable words in each class were very similar among the classes as the top 4 words were the same in every class, with the exact same rankings, with varying words for the 5th rank.

I) *(2 pts) Would it be appropriate to fit an ROC curve in this scenario? If yes, explain why. If no, explain why not.*

Answer: With the current data we have, it would not make sense to fit an ROC curve because an ROC curve is used to show a false positive rate vs true positive rate for a binary predictor but in our case, our model is a ternary predictor. This means that our model can not compute the true positive and a false positive rate with the data we have now.

So what we can instead do to modify our data would be to divide the data that we have now into ternary 2 confusion matrices. One of the matrices will calculate the true and false positive rate when the predictor predicted “neutral”, meaning that it will calculate the number of times the predictor correctly assumed the tweet was neutral and the number of times the predictor incorrectly assumed the tweet was “neutral”. We can do the same for the “positive” case, and this way we can modify the data so that we can have an ROC curve based on which predictor we want to see.

J) *(2 pts) Redo parts G-H using TF-IDF vectors instead of count vectors. You might find Scikitlearn’s TfidfVectorizer() transformer useful. Report the training and test accuracy. How does this compare to the accuracy using count vectors?*

Answer: The training and testing accuracy for the TF-IDF vectors were approximately 0.67 and 0.45 respectively. This means that the count vectorizer was more accurate for the training data by about 1% while the TF-IDF vectorizer was more accurate for test data by about 0.2%. This means that overall the countvectorizer and TF-IDF vectors were about equally accurate.

K) (3 pts) Recall lemmatization converts each word to its base form, which is a bit stronger than simply taking the stem. Redo parts E-H using TF-IDF vectors instead of count vectors. This time use lemmatization instead of stemming. Report train and test accuracy. How does the accuracy with lemmatization compare to the accuracy with stemming?

Answer: The lemmatization has a training accuracy of approximately 67.04% and a test accuracy of approximately 44.68%. We see that the accuracy of the lemmatization is greater than the accuracy with stemming for both train and test accuracy where the train accuracy was greater by about 2% and the test accuracy was greater by about 1%.

Bonus *(1 pt): Is the Naive Bayes model generative or discriminative? Explain your response.*

Answer: Naive Bayes is a generative model because it models the joint probability distribution for example, P(X, Y) where X is the data and Y is the class. It first estimates the distribution of each class and the probability of the data given to each class, so P(X|Y) then uses Bayes theorem to calculate the probability of the class given the data, so P(Y|X) hence "generating" new data/instance points by sampling from the joint distribution. Naive Bayes is not a discriminative model because discriminative models like logistic regression tries to model the decision boundary between the classes and estimate P(Y|X) directly without estimating the distribution of each class.

Part II: Having fun with NLP using the Twitter API

(I) Problem description and motivation

Given our group's collective interest in music and live performance, we were drawn to the following question: can we predict the popularity of an artist who performed at Coachella - one of the most popular music festivals in the United States - based on the number of tweets about them and their sentiment? This question is difficult to solve due to the complexity of

human language and its reliance on context. For instance, the word “sick” generally conveys a negative feeling of being ill, but it can also be used to convey positivity in particular contexts. These lines between neutral, positive, and negative sentiment are thin and make classification hard, especially when context is limited. A tweet on Twitter could only be a couple of sentences long and still simultaneously hint at both positive and negative aspects of an artist’s performance. Fortunately, the dataset of Coachella-related tweets we used for analysis helped us navigate this problem since each tweet was pre-labelled with a certain sentiment. Furthermore, we found an article by Yo-Ping Huang, Nontobeko Hlongwane, and Li-Jen Kao about a similar project where they used sentiment analysis of tweets to predict the number of likes musical artists got. While their analysis determined popularity using like counts on tweets, we categorized popularity by thresholding follower counts into a binary variable. This led to our use of logistic regression instead of the approach they took where they graphed the sentiment and number of likes to examine popularity trends.

(II) Describe the data

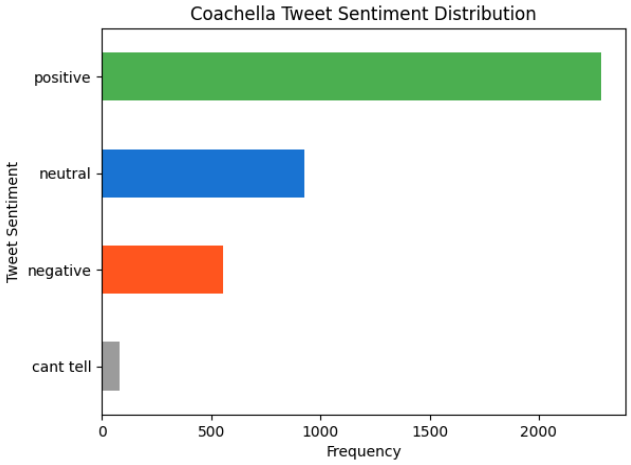
As mentioned earlier, our analysis relied on a dataset found on Kaggle containing tweets from the 2015 Coachella festival. This dataset consists of 3846 observations and 11 columns, listing columns such as “coachella_sentiment” which categorizes a tweet into positive, neutral, negative, or unsure, and “retweet_count” which indicates how many times a tweet was retweeted. We also used Tweepy, a Twitter API for Python, to extract the follower counts for 60 of the artists who performed. This was achieved by manually looking up the username of each artist and then feeding them into Tweepy’s “get_users” method, passing along the usernames and the string “user_fields=’public_metrics’” as arguments. Similar to us, we found that Robert

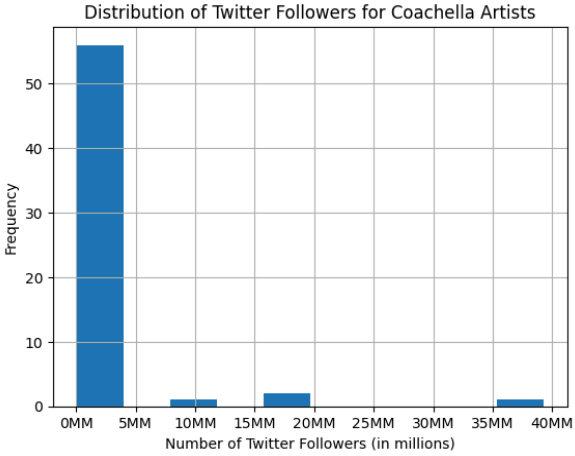
Vargas previously used the Coachella dataset to analyze the trend of the most used words between weeks one and two of the festival. Likewise, Y. -P. Huang, et al's paper found that there is a strong correlation between sentiment scores and the like count for tweets, which is similar to our pre-analysis belief that high sentiment scores result in high follower counts. A strength in our data is that every tweet mentions the word "Coachella", resulting in all tweets having a similar context. This means that context-dependent words are more likely to be categorized correctly, increasing the effectiveness of any natural language processing. However, this feature can also be viewed as a limitation since not everyone who tweets about the concert will mention the word "Coachella". For example, people who enjoyed a specific performance may post a tweet with the artist's name but not with the word "Coachella". Although the tweet is about Coachella, it would not be captured by this dataset, resulting in the dataset not being representative of the whole population. Another important limitation in our data is that tweets in this dataset are from 2015, whereas we extracted follower counts in 2024. This is a huge gap in time and could lead to a weak correlation between the two variables.

(III) Exploratory data analysis

In preparation for the modelling phase of our research, we took some time to explore and process the raw data. Out of the 3846 tweets, we discovered that 2283 (~59%) were positive, 928 (~24%) were neutral, 553 (~15%) were negative, and 82 (~2%) were ambiguous (as shown in the first plot below). We interpreted this to mean that most people enjoyed the festival and thus, enjoyed the artists performing. Of the 60 artists we extracted, we found their Twitter follower distribution to vary greatly and contain extreme outliers (as shown in the second plot below). The minimum and maximum follower counts were around 1000 and 39,000,000 followers, with the

median and 75th percentile at around 293,000 and 750,000 followers, respectively. This median ultimately influenced our decision to label an artist as popular (i.e. our outcome) if they had over 300,000 followers. With our labels set, we focused on creating the features necessary for our sentiment-based model. We used regular expressions to extract the number of tweets that each artist was mentioned in and then broke this down further by tracking how many of those tweets belonged to each sentiment class. After analyzing the number of Twitter mentions for popular versus unpopular artists, our data visually suggested that there might be a slight correlation between tweets and their follower counts. This naturally guided us to continue our exploration with a machine-learning model.





(IV) Describe your machine-learning model

The model we used for our analysis was logistic regression (i.e. binary classification). We implemented logistic regression to predict whether or not an artist has over 300,000 followers on Twitter. Logistic regression is similar to linear regression: it fits a line using the equation $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$ where z is the dependent variable, β_0 is the y-intercept, $x_1, \cdots, x_n$ are the parameters, and $\beta_1, \cdots, \beta_n$ is the amount of increase/decrease in the outcome in

one unit change of the independent variables holding everything else constant. After that, the expit function gets applied to z and returns the probability of $z = 1$ which we can threshold to determine whether or not an instance is in the positive (1) or negative (0) class. For our model, we had the number of negative, neutral, positive, and ambiguous tweets about an artist as our independent variables and the number of followers the artist has as our outcome. Logistic regression was ideal here since it is designed for binary outcomes and is simple to implement using our dataset. It also made sense to use logistic regression given that it is a supervised model and we could easily extract our labels via Tweepy. In contrast, if the labels were not known or easily retrievable, we would have had to look into unsupervised models such as clustering. To evaluate the performance of our model, we visualized the confusion matrix and ROC curve, as well as computed accuracy, precision, recall, specificity, F1 score, and AUROC score. Given that we didn't expect any serious consequences to arise from false negatives or false positives, our goal was to have general consistency across all metrics. For our baseline comparison, we attempted to compare it to the analysis in the article by Y. -P. Huang, et al, but realized it was impractical since their sample size was much larger than ours. If we had full access to the Twitter API to fetch tweets, we could have done a broader analysis that captured more than Coachella, making the baseline comparison meaningful.

(V) Results and Conclusions

In terms of performance, our model performed better than expected but still has room for improvement. We were happy with the consistency across standard metrics as we achieved 69% accuracy, 71% precision, 62% recall, 75% specificity, 67% F1 score and 69% AUROC score on our test dataset. However, a major flaw in this evaluation is that our test dataset only contained 16 observations. This makes our results questionable as the true relationship in the population is

unlikely to be represented by such a small dataset. To help combat this and stabilize our model, we ensured that our train and test sets randomly sampled a proportionate number of the positive and negative outcome classes. An interesting interpretation we gathered from the model coefficients is that the number of negative tweets about an artist correlates to higher popularity than the number of positive tweets. Intuitively this is reasonable as those who are prevalent in the media are more likely to receive criticism publicly than those who aren't, but this claim requires more research to be accepted as truth. In the future, we would like to capture more of the music community by aggregating tweets from multiple festivals of varying musical genres instead of strictly Coachella. Additionally, given the rise of Spotify and online streaming, we think using the number of Spotify streams an artist has to determine popularity would be more appropriate than Twitter followers. In conclusion, our model leaves plenty to be desired but serves as a stepping stone to further research into fan sentiment and musical artist popularity.

References

Y. -P. Huang, N. Hlongwane and L. -J. Kao, "Using Sentiment Analysis to Determine Users' Likes on Twitter," 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science

and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), Athens, Greece, 2018, pp. 1068-1073
https://ieeexplore.ieee.org/abstract/document/8512019

Vargas, Robert. *Sentiment Analysis - Coachella Music Festival*, 20 June 2020, rstudio-pubs-static.s3.amazonaws.com/638006_7dd134a9cb964ee8bfff9d6eedb2543a.html.