Alex Klein & Winston Vu

Dr. Palomino-Grubb

IST 736

6/18/2022

**Final Project: Song Lyric Classifier**

**Introduction:**

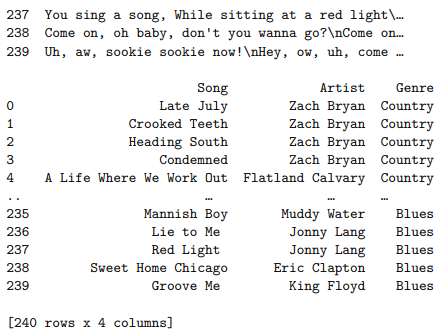
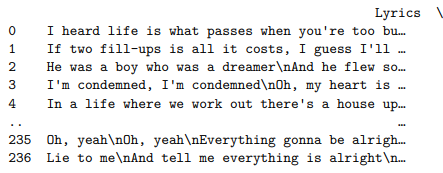
Music is a very important aspect of many people’s daily lives, often one of the only methods people have to reduce stress. People enjoy listening to music as it provides them with enjoyable melodies, lyrics, and can boost any mood. There are many genres of music, with many different artists that exist in each. The genres provide different types of lyrics and sound for people to enjoy. Many genres have overlap between each other, many genres actually emerged from existing genres. For example, country music can have influences ranging from rock and blues to reggae and reggaeton. There are also many different types of sub-genres for each genre of music as well. An example of a genre having multiple sub-genres would be the different styles of rock, such as alternative rock, classic rock, and punk rock. Everyone has different preferences of music they prefer. Those preferences can depend on the lyrics written by the artist, the melodies and sounds of the song, the instrumentation choices, and the voice that is sung by the artist. Each song is different from another, but there may be similar themes, word choice, melodies, sounds, instrument use, and moods within each song.

It is easy to distinguish a song based on a singer’s voice, beats, instrumentation, lyrics, and sound of the song. Even the genre of the song can help distinguish the song or artist. In apps, such as Spotify, there are playlists that are created that correlate to the mood, sound, and general artist, but there are no playlists that are created based on lyrics and genre. Lyrics are harder to distinguish since some lyrics can seem like they belong to other genres or can be hard to understand. Without an artist present, it would be difficult to note the lyrics of a song to its exact genre. This project’s goal is to see whether a machine learning algorithm would be able to predict the genre of a song based solely on the song’s lyrics. With some genres, it may be hard to predict due to the influences that other genres may have on that specific genre. Other genres might be easily interpreted due to the lyrics and words themselves. If a machine learning algorithm could correctly predict and classify the right genre of a song from its lyrics, then it would be massive for these music streaming platforms and for the happiness of the public.

**Method:**

This section of the report will discuss the methodology used in this project. The first step of this project was determining a problem that could be solved using different text mining and machine learning techniques. The problem this report aims to remedy is music streaming services, such as Spotify, having such a massive influx of new music each day that it is impossible for them to manually review and classify each song in real time. The reason that song review and classification is important on these music platforms is because apps like Spotify create custom playlists for their users based on their music preferences. These playlists include songs classified by different artists, genres, moods, and more. Therefore, classifying a song by genre is an essential step for the platform and could be enhanced by utilizing text mining and machine learning algorithms to automate the process. Once basic genre classification is mastered the next step can be subgenre and mood of song within the genre classifications. The goal of this project and report is to be able to correctly predict the genre classification of a song based on its lyrical content.

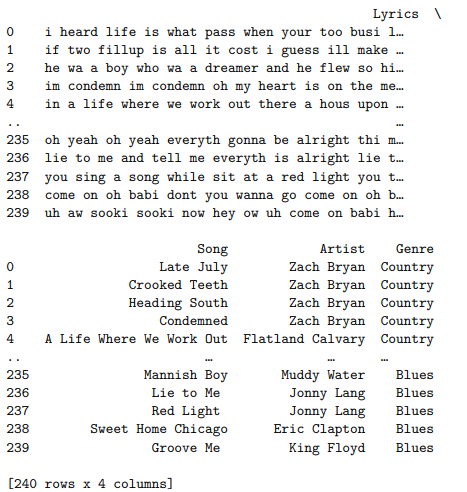
After determining the goal of this project, a dataset was manually collected and stored in a csv file. That dataset consists of 240 rows, one row for each individual song, and four columns. The column names are as follows; Lyrics (the lyrics to each song), Song (the name of the song), Artist (the artist for the song), and Genre (the genre of the song). Each field’s values are in text format. There are six different genres in the Genre field of the dataset. Those genres are; Blues, Country, Pop, R&B, Rap, and Rock. Each genre is equally balanced, consisting of 40 songs in each. Each song is unique, and the corresponding data was manually scrapped from different song lyrics hosting websites. The dataset was originally half the size at the project checkpoint. Based on feedback and recommendations from the project checkpoint, the dataset was doubled as one of the methods to improve the classification prediction results. After the data was manually collected and consolidated in the csv file a Jupyter Notebook was created for the project.



The above image is of the dataframe loaded in the Jupyter Notebook.

Once the Jupyter Notebook was created the following packages were imported for use in the project. Those packages are; os, re, sys, random, nltk, nltk.corpus stopwords, nltk.tokenize word\_tokenize, nltk FreqDist, matplotlib.pyplot, pandas, seaborn, sklearn.feature\_extraction.text CountVectorizer & TfidfVectorizer, sklearn.model\_selection train\_test\_split & cross\_val\_score, Sklearn.naive\_bayes MultinomialNB, Sklearn.metrics classification\_report, Sklearn.svm LinearSVC, Sklearn svm, Sklearn.metrics confusion\_matrix, Sklearn.neural\_network MLPClassifier, and Sklearn.ensemble RandomForestClassifier. After the packages were imported, the dataset in csv format was read into a pandas dataframe using the read\_csv command with the following settings, engine = ‘python’ and encoding = ‘unicode-escape’.

The dataframe was inspected to confirm that it loaded correctly while also identifying any data cleaning that needed to be performed on the text within the Lyrics field. The first thing found in need of cleaning was the regex code of “\n” within the song lyric text. A function was created and applied to the Lyrics field to replace all occurrences of “\n” with “ ” (a blank space). After removing all instances of “\n”, a function defined to remove punctuation and special characters, convert all text to lowercase, and remove numerical digits from the Lyrics field was created and applied. Next, another function was created that used nltk’s PorterStemmer to stem all the words in the Lyrics field of the dataframe. The final data cleaning step performed was using the dropna command with, how = ‘any’ and inplace = True, to drop any NaN values from the dataframe. Prior to the project checkpoint stopwords were also removed. However, based on the recommendations and advice during the checkpoint, stopwords were no longer removed from the Lyrics field in a bid to increase model classification prediction accuracy. The models did perform better, therefore stopwords were not removed from the Lyrics field in the final version of this project being presented in this report.



The above image is of the cleaned dataframe in the Jupyter Notebook.

Once the data cleaning was complete, the next step was to process the dataframe for use in the machine learning algorithms. The first step was the creation of the testing and training subsets from the full dataset. These training and testing subsets are required to train the machine learning models and then test their classification prediction accuracy. The train\_test\_split command was used to create the four necessary training and testing subsets, X train, X test, Y train, and Y test. The settings used for the train\_test\_split command will be listed and then explained. Those settings are as follows, (df['Lyrics'], df['Genre'], stratify=df['Genre'], test\_size=0.3, random\_state=1). The df[‘Lyrics’] sets the X to the Lyrics field. The df[‘Genre’] sets the Y to the Genre field. X indicates the features being used to determine the Y, or target. In other words, the Lyrics of the song are used to determine the Genre of the song. Stratify is set to the Genre field in order to make sure that the training and testing subsets consist of equal and balanced amounts of songs from each genre in each subset. The test size was set to 30 percent in line with standard recommendations. The test size of 30 percent automatically made the training size the remaining 70 percent of the data. Finally, random state being set to one is essentially setting the seed value of the random shuffle to one. Setting the seed value of the random shuffle to a value ensures that the data is randomly shuffled the same way each time the code is run. If random state wasn’t set or used, each time the code was rerun the data inside the training and testing subsets would be different, leading to different results from each model. Using the seed value essentially randomizes the data once and keeps that same randomization going forward.

Following the creation of the training and testing subsets, two different vectorizer packages were set up. Those vectorizers were Sklearn’s CountVectorizer and TfidfVectorizer packages. The CountVectorizer had the following setting, encoding = ‘latin-1’. The CountVectorizer uses unigrams by its default settings. The TfidfVectorizer instance used the default settings, also using unigrams by default. The CountVectorizer unigram instance was used with the MultinomialNB (MNB) and Neural Network (NN MLP) models. The TfidfVectorizer unigram instance was used with the LinearSVC (SVM) and RandomForestClassifier (RandomForest) models. All four models mentioned are packages from Sklearn. After setting up the vectorizers, the fit\_transform command was used to created vectorized forms of the X train and X test subsets for both vectorizers. Those vectorized X train and X test subsets (Unigram CountVectorized X train and test subsets and Unigram TfidfVectorized X train and test subsets) were used to train their respective models.

The next steps of the methodology were repeated for each model and will therefore only be explained once. The unigram CountVectorized X train and X test subsets were used to train the MNB and NN MLP models. The unigram TfidfVectorized X train and X test subsets were used to train the SVM and RandomForest models. The CountVectorized based models had the following settings, MultinomialNB() had default settings while MLPClassifier used the following, (solver = ‘lbfgs’, random\_state=8, hidden\_layer\_sizes = (5,4,3), max\_iter = 10000). Solver was set to lbfgs because that is the best setting for a small dataset like the dataset used in this project. The hidden layer sizes and max iterations were set based on the number of genres and the number of iterations needed to successfully run the neural network with the highest prediction accuracy. Random state was explained earlier in this section and was used to keep the results from changing each time the code was ran. Each model was fit to the CountVectorized X train and Y train subsets, then the predict command was then used on the CountVectorized X test subset to create the Y prediction dataset. Next, the classification\_report command was used on the Y test subset and the Y prediction dataset and outputted the classification report. The final step was using the cross\_val\_score command on each model, the CountVectorized X test subset, and the Y test subset in order to perform 10-fold cross-validation.

 The above steps were repeated using the unigram TfidfVectorizer, instead of the unigram CountVectorizer, for the SVM and RandomForest models. The SVM and RF algorithms both used their default settings. Both models were fit to the TfidfVectorized X train subset and the Y train subset before using the predict command on the TfidfVectorized X test subset to create the Y prediction dataset. The classification\_report command was used on the Y test and Y prediction datasets for both models to output the classification report. Finally, the cross\_val\_score command was used on each model, the TfidfVectorized X test subset, and the Y test subset in order to perform 10-fold cross-validation.

After running the four models with either, the unigram CountVectorizer or the unigram TfidfVectorizer, the process was repeated using unigram and bigrams. The above steps used for the unigram vectorizers were repeated with the following changes to the two vectorizers. The CountVectorizer used encoding = ‘latin-1’ and ngram\_range = (1,2). N-gram range 1,2 utilizes unigrams and bigrams rather than just the default unigrams used previously. The TfidfVectorizer was set to ngram\_range = (1,2) for the same reason. The two new instances of the Vectorizers for unigrams and bigrams were then used the same way as the original unigram vectorizer instances were. The same steps were used on the four models to fit and train the models, make Y predictions, output the classification reports, and to perform 10-fold cross-validation. The outputs of the unigram-based models could then be compared against the outputs from the unigram and bigram-based models in order to determine which led to better results.

**Result:**

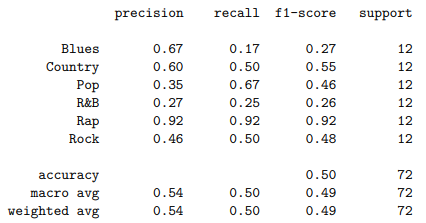
This section of the report will discuss the results of the methodology described above. The results will be explored, analyzed, and then compared in order to determine the model had the best performance.

To start, understanding the two vectorizers and the four models used in this project is necessary. The two vectorizer packages from Sklearn take the text from the Lyrics field and transforms them into numbers that can be understood by the different models. Both vectorizers perform basic pre-processing steps on the text such as making all text lowercase, removing special characters and numbers, and tokenizing the text. The CountVectorizer turns the text into a sparse matrix with the model’s default number of features (words). It does so by counting the word frequency in the corpus, determining the most recurring words to use as features, and then creating the sparse matrix. The CountVectorizer gives equal weightage to each word used in the corpus (Using CountVectorizer to Extracting Features from Text, 2020). The TfidfVectorizer works in a different manner. The TfidfVectorizer consists of two different parts, the Tf (term-frequency) and the idf (inverse document frequency). Those two parts are multiplied against each other to create the values. The Tf is the number of times a word is in a single text within the corpus. The idf is the log of the number of total texts in the corpus divided by the number of texts the word is in. The TfidfVectorizer utilizes the weightage of the Tf and document frequency (Chaudhary, 2020). Both vectorizers create matrices that can be used by the models and also outputted as an array.

The four models used in this project are the MNB (MultinomialNB), SVM (LinearSVC), RandomForest (RandomForestClassifier), and the NN MLP (MLPClassifier). Each model utilizes a different algorithm that performs classification tasks in this project. Those algorithms and their basic functions will be briefly explained here. The MNB model is a probabilistic algorithm that uses the Bayes theorem to tag a text and to calculate the probability of each tag within a given sample. The tag with the highest probability is the output (Naive Bayes text classification, 2009). SVM stands for Support Vector Machine. SVMs are a supervised machine learning algorithm used for regression or classification, although they perform better on classification tasks. SVMs work by creating a hyperplane using vectors from the dataset. The data points are separated into different classes within the n-dimensional space. The goal is to create the best line, meaning to place the data point into the best class within the hyperplane (Nelson, 2020). The RandomForest algorithm is a form of decision tree modeling used for classification and regression tasks. RandomForest’s use decision forests (combination of decision trees) to predict classifications. The decision forests are made up of random decision trees that use random samples of the data and each tree node has random features selected to generate the best split (Mbaabu, 2020). The final model is the NN MLP algorithm. The NN MLP is the multi-later perceptron classifier, a form of neural network that utilizes an optimized log-loss function using LBFGS. Neural Networks were designed to simulate the human brain, specifically the neurons within the brain. Neural Networks consist of weighted nodes and hidden layers. Each layer consists of a number of nodes that connect to each other. The neural networks train themselves from the training dataset and create weights for each node until the optimal weight is found (Brownlee, 2016).

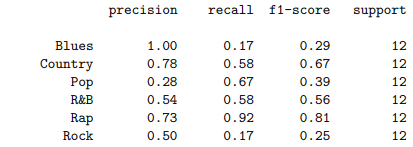
Now that the vectorizers and models have been briefly explained the results can be examined. The classification reports and 10-fold cross-validated accuracies screenshots will be posted below and then analyzed and compared.

Unigram MNB:





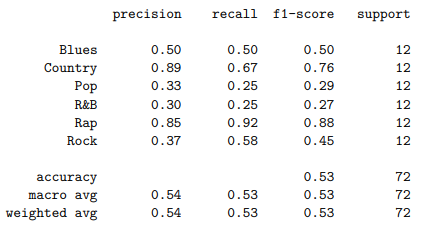
Unigram and Bigram MNB:





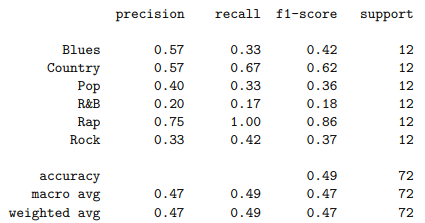


Unigram SVM:



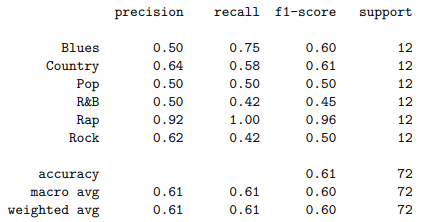


Unigram and Bigram SVM:



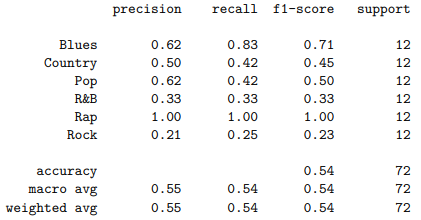


Unigram RandomForest:



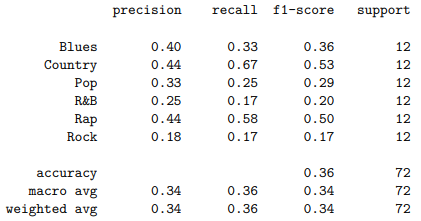


Unigram and Bigram RandomForest:



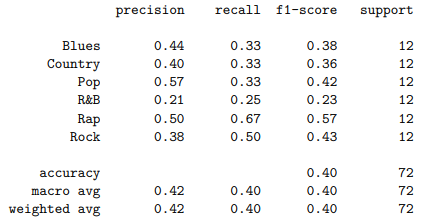


Unigram NN MLP:





Unigram and Bigram NN MLP:





The results of the above models show that the unigram instances of each model out performed the unigram and bigram instance of those same model based on the 10-fold cross-validation accuracies. K-fold cross-validation works by resampling by a set value (k value) on a limited data sample. 10-fold means that the k value is set to 10, the gold standard number to use for k-fold cross-validation. K-fold cross validation shuffles the dataset randomly and then splits it into the k value number of groups. One group is then used as the test data set and the remaining groups are combined as the training set. The model is fit on the training set and predicts the test set, the predictions are then compared against the real values to establish the predictions accuracy. The prediction accuracy is stored and the process is repeated so that each group is the test data set once. The prediction accuracies from each run are then averaged to determine the final k-fold cross-validated accuracy. K-fold cross-validation is considered the gold standard in determining the performance of machine learning algorithms and that is why it is the first metric we are analyzing.

The unigram MNB model’s 10-fold cross-validated accuracy was 41.96 percent compared to the unigram and bigram MNB models 32.32 percent. The unigram SVM model’s 10-fold cross-validated accuracy was 46.96 percent compared to the unigram and bigram SVM models 44.11 percent. The unigram RandomForest model’s 10-fold cross-validated accuracy was 52.68 percent compared to the unigram and bigram RandomForest models 45.71 percent. Finally, the unigram NN MLP model’s 10-fold cross-validated accuracy was 27.86 percent compared to the unigram and bigram NN MLP models 19.46 percent. The highest performing model was the unigram RandomForest model and the second highest performing model was the unigram SVM model. This makes sense as the RandomForest and SVM models are often considered two of the better classification algorithms. The worst performing model was the NN MLP. The poor performance of the NN MLP model was understandable because neural networks often need large datasets to perform well. The dataset used in this project is considered very small by neural network standards.

As explained in the method section of this report, the dataset was doubled to 240 songs after the project checkpoint. During the checkpoint, another recommendation was to experiment with stopwords removal and word stemming to determine the combination that led to the best model performance. The combination that led to the highest model performance was determined by testing each combination. The best combination is what was described in the method section, using word stemming but not using stopwords removal. That combination led to the highest model performance by a significant margin. The biggest improvement of performance was found after doubling the dataset in size. This makes sense because when the dataset was only 120 songs there wasn’t enough songs in each genre in the testing subset to support 10-fold cross-validation. There were only 6 songs in each genre in the testing subset. Because of that, at the project checkpoint, 6-fold cross-validation was utilized instead of 10-fold cross validation. After doubling the dataset in size, 10-fold cross-validation was possible. The fact that the 10-fold cross-validated accuracies for all the models were higher than the 7-fold cross-validated accuracies was important in confirming the improvement in performance. Doubling the dataset in size not only grew the number of songs in each genre in the testing subset to 14, but also increased the number of songs in each genre in the training subset from 14 to 28. Having more songs to train the models helped improve their performance as well as enabled the use of 10-fold cross-validation. No longer removing stopwords also led to marginal model performance improvements, just not to the extent doubling the dataset had on model performance.

After examining and comparing the 10-fold cross-validated accuracies for each model we can look at the classification reports to further understand each model’s performance. There are four fields in the classification report with values for each genre. The first field is Precision. The Precision field shows the percentage of true positive predictions in relation to total positive predictions. The second field is Recall. Recall measures the percentage of true positive predictions in relation true positive and false negative predictions combined. The third field is F1 Score. F1 Score is the weighted mean of precision and recall. The closer that the F1 score is to 1 the better the models performance. Below the genre values for those four fields is the Accuracy, Macro Avg (of accuracy), and Weighted Avg (of accuracy). The Accuracy value measures the amount of true positive and true negative predictions divided by the number of samples. In other words, the Accuracy value indicates the percentage of predictions that were correct. The Macro Avg and Weighted Avg give the Accuracy value averaged by weight or macro. There is also a field called Support which has a value for each genre. The support value indicates the number of values in each genre in the testing dataset, in this case 12 songs in each genre within the testing subset. Having a Support value of 12 for each genre indicates that each genre is balanced in the testing dataset.

The classification report Accuracy values, Weighted Avg values, and Macro Avg values all tend to be higher than the 10-fold cross-validated accuracies. This is to be expected. As explained previously, k-fold cross-validation resamples the training and testing datasets, retrains and tests the models, and averages their accuracies k number of times. Doing so can expose model overfitting, underfitting, handles data outliers better, and gives a much more accurate accuracy value. Therefore, the k-fold cross-validated values are more important than the Accuracy values from the classification report. The Precision, Recall, and F1 values can help explain the model performance in more granular detail.

After examining and analyzing the results from this project, the following outcomes were determined. The current 10-fold cross-validated accuracies from the models, from either unigrams or unigrams and bigrams, are not accurate enough to determine the genre of a song based on its lyrics alone. Even the most accurate model, the unigram RandomForest, only achieved slightly above 50 percent genre classification accuracy. Based on the model performance improvements after doubling the size of the dataset, further increasing the dataset size in a balanced manner could lead to further performance improvements. However, utilizing a larger dataset will eventually hit a convergence point where further increasing the dataset will no longer lead to better performance. Utilizing different pre-processing and data cleaning techniques could also further improve the results.

**Conclusion:**

 The integration of music into society has brought a new dynamic to people’s everyday lives. There are many artists and singers who write and release songs daily, and there are many songs from each artist to listen to. There are also many different genres of music. People can choose to listen to a variety of genres, such as country, rap, pop, and more. Each person’s favorite genre to listen to varies. Some people may listen to a couple types of genres but may avoid others. People’s preferences are diverse and constant evolving. Each genre and each individual song are different. The melodies, sounds, instrumentation, artist, voice, and tone all vary in every genre and every song. Even songs with similar topics between each other can have noticeable differences between them.

Lyrics are an aspect of a song that is currently difficult to classify the song by. Without any other variables, such as sound and singer, many song lyrics can be hard to classify, especially since many songs cover the same topics such as love or heartbreak. Songs melodies can impact the mood of a person, but the lyrics tell the listeners a story from the singer’s perspective. Lyrics are important for many listeners, and if people are trying to create a playlist with songs that are similar lyrically, it can be difficult to create such a playlist. All music genres have similar aspects to each other. Many genres influence each other or overlap. There are also sub-genres, such as country rock, jazzy pop, and slow rap, that merge existing genres into a new genre altogether. These aspects create more difficulty when trying to classify a songs genre based on its lyrics.

This project utilizes lyric classifying models, trying to see whether a machine learning algorithm can predict a songs genre solely from its lyrics. The four models used to classify the genre of songs from their lyrics in this project were Multinomial Naïve Bayes, Support Vector Machines, Random Forest, and MLP Neuro Network. These models were evaluated using 10-fold cross-validation to determine their classification prediction accuracies. The unigram-based models results performed better than the unigram and bigram-models. This means that it is better to predict words singularly than using a combination of single and two-word combinations. The results from this project are not perfect but do indicate that with further exploration and optimization that song lyrics can be used to identify that songs genre. For example, the genre of rap had the highest f-1 score out of all the genres. This indicates that it is easier to predict rap songs genre correctly based on their lyrics. It is worth noting that the unigram RandomForest model had the highest 10-fold cross-validated accuracy of all the models at 52.68 percent. That accuracy may be low, but based on the scope of this project was a success. A prediction accuracy that is higher than 50 percent means that with further optimization and improvements the RandomForest model could become more and more accurate. The questions remaining are whether a dataset with more values would have a higher accuracy and whether different pre-processing and data cleaning steps would improve the results. Overall, classifying the genre of a song from its lyrics requires more work to be done, but the overall baseline is a huge success.

**References:**

Brownlee, J. (2016, May 16). *Crash Course On Multi-Layer Perceptron Neural Networks*. Machine Learning Mastery. https://machinelearningmastery.com/neural-networks-crash-course/

Chaudhary, M. (2020, September 17). *TF-IDF Vectorizer scikit-learn*. Medium. https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a

Mbaabu, O. (2020, December 11). *Introduction to Random Forest in Machine Learning*. Engineering Education (EngEd) Program | Section. https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

*Naive Bayes text classification*. (2009). Stanford.edu. https://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html

Nelson, D. (2020, February 8). *What are Support Vector Machines?* Unite.AI. https://www.unite.ai/what-are-support-vector-machines/

*Using CountVectorizer to Extracting Features from Text*. (2020, July 15). GeeksforGeeks. https://www.geeksforgeeks.org/using-countvectorizer-to-extracting-features-from-text/

‌