SENTIMENT ANALYSIS OF SONG LYRICS USING PYTHON

Prepared by : **BEYZA NUR KEBELİ**
Student No : **090170306**

Submission Date : **June 18, 2022**

Course : **MAT 4902E**
Supervisor : **ASSOC. PROF. AYBİKE ÖZER**

Table of Contents

Table of Figures

1. Introduction

Emotion classification of songs has become a popular area of interest with the development of music listening applications' features. Now, music applications offer recommendations based on a user's previous song choices and mood. To improve the recommendation systems, song sentiment analysis is needed to be studied and improved. For the sentiment analysis, audio signals or lyrics of the songs can be used. The aim of this study is to perform sentiment analysis on song lyrics to classify them into categories such as positive and negative. In order to achieve this, different techniques can be used. The techniques that are used most frequently can be seperated into categories such as machine learning-based techniques, lexicon-based techniques, and hybrid techniques [1]. In this study, Support Vector Machine (SVM), which is a supervised machine learning model, and a lexicon-based module named VADER in Python will be used [2]. This study aims to show how a machine learning model can be used with a lexicon-based module as a hybrid technique to classify the emotion of songs. Apart from presenting the results, I find the fields of sentiment analysis and machine learning valuable enough to analyze and gain a better understanding of their difficulties.

For data collection of the study, the website Kaggle was used. The dataset contains a collection of songs in English from different music genres. The study was done using Python programming language because of its libraries such as pandas, sci-kit learn, VADER, and many more.

After using the lexicon-based module VADER for labeling the dataset, SVM was applied to classify the songs. For comparative analysis, precision, recall, f1-score, and support were used. The study results show that the model performs well to classify the songs, with a precision rate of 83% for positive songs and 89% for negative songs. It can be said that hybrid techniques, such as the one used in this study, can perform well for the song emotion classification problem.

2. Scope of Design and Areas of Usage

Nowadays, music streaming platforms are very popular because of the features they offer to listeners. The most used feature of these applications is their song recommendation system. Song recommendation systems suggest similar songs to what you most listen to. Search and selection of songs that were once performed on the basis of Title, Artist, or Genre, now also use mood as a new and important attribute of music [3]. Thus, focusing on the sentiment of lyrics of songs would very much improve song recommendation systems. Analyzing the lyrics of songs falls under the work of sentiment analysis. Sentiment analysis is a field of Natural Language Processing (NLP) that computationally determines the subjective meaning of a given text material to decide on its sentiment and emotion.

Applications of sentiment analysis are mostly used by companies to analyze customer feedback and social media responses to improve their customer services and marketing. Sentiment analysis can be used for many more purposes like monitoring politics and social issues. For many reasons like the ones listed above, sentiment analysis became an area of

interest over time. *Figure 1* shows the increased interest in this topic from the year 2004 to this day.

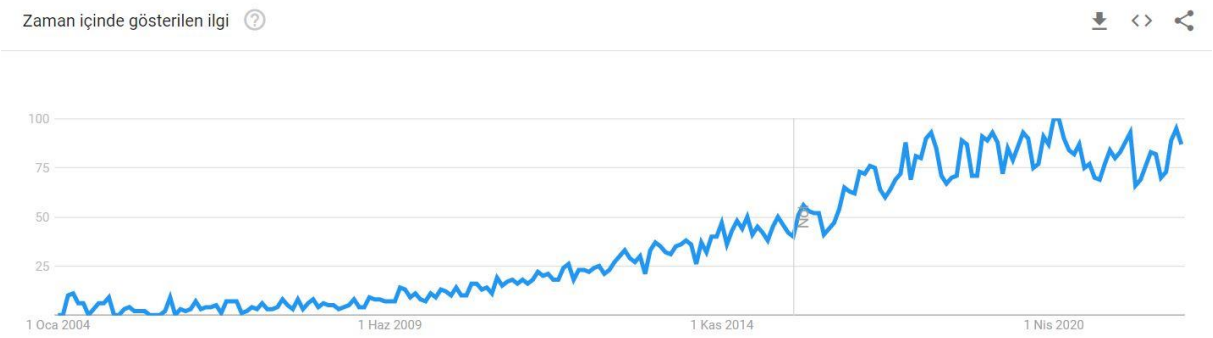

Figure 1: Google Trends (www.google.com/trends) data showing the relative popularity of search string ‘‘sentiment analysis’’.

3. Conducted Studies

3.1 Literature Research

The classification for the sentiment of songs can have labels more than just positive, negative, and neutral, as well. For example, Napiel et al. labeled the songs into 5 categories which were anger, disgust, fear, joy, and conscientiousness using IBM Tone Analyzer [4]. Another example is Kim et al. using many labels such as Joy, Fear, Anger, Sadness, Anticipation, Trust, Surprise, Disgust and emotions between these labels [5].

Sentiment analysis of songs can be done using the audio data as well which is what was done in the early 2000s. From the 1990s to the 2000s, song sentiment analysis was studied under acoustic signal processing (ASP). However, since catching sentiment hints from audio data is hard, audio-based song sentiment analysis was found to be inefficient [6]. At the moment, studies on the topic of song sentiment analysis are mostly done using text data, which is the lyrics of songs. Not only it is more efficient to work with lyrics for emotion classification, but also it is easier to collect and process than audio signals [7]. Because of these reasons, this study will also be focused on using the lyrics of songs for the classification.

In general, sentiment analysis is not an area of machine learning. However, using machine learning techniques definitely outdo human-produced baselines [8]. Some of these machine learning techniques can be listed as Naive Bayes, Maximum entropy classification, and Support Vector Machine (SVM). Pang et al. used the techniques listed above to classify movie reviews and showed that SVMs work the best while Naive Bayes performed worst, although the differences are not that large [8]. This was a big improvement in the field. Since SVM gave

the most accurate results, this study will focus on SVM for the machine learning model part. Moreover, He et al. used these three machine learning algorithms in their study about how feature extraction can improve music emotion classification and also received good results using the SVM algorithm [7]. Another method for performing text sentiment analysis would be CNNs (Convolutional Neural Networks), which is a deep learning method. Chen et al. used traditional CNNs and CNNs with SVM and saw that CNNs with SVM performed better than the traditional CNNs model [9]. It can be seen that sentiment analysis, and more generally NLP, became an important direction of the field of AI. This is because of the development of internet and the accumulation of large amount of text data [9].

SVM is a supervised (feed-me) machine learning algorithm that can be used for both classification and regression challenges [10]. In this study, SVM is used for a classification challenge. SVM for classification works with a labeled dataset and learns how to do the labeling. The main principle of SVMs for classification is determining linear separators in the search space which can best separate the different classes. The study will focus on linearly separable dataset and there will be two different classes for the separation, which are positive and negative. The reason why these two classes were chosen to work with will be explained in Section 3.2.

A linear classifier (separator) has the form

$$f(x) = w^T x$$

where w is the weight vector.

Using the Perceptron algorithm, we can find w, the hyperplane, that separates the categories. In two dimensions, the seperator is a line and in three dimensions, the seperator is a plane. Here, we are generalizing these and use the term “hyperplane” for a classification problem in n-dimensions.

Perceptron algorithm works as followed:

1. Initiliaze $w = \vec{0}$.
2. While there is i such that $f(x_i)y_i < 0$, do

$w := w - \alpha x_i sign(f(x_i))$.

Here, $y_i \in \{-1, 1\}$.

If the data is linearly separable, then the algorithm will converge and at convergence, the hyperplane can be found as $w = \sum_{i=1}^{N} \alpha_i x_i$.

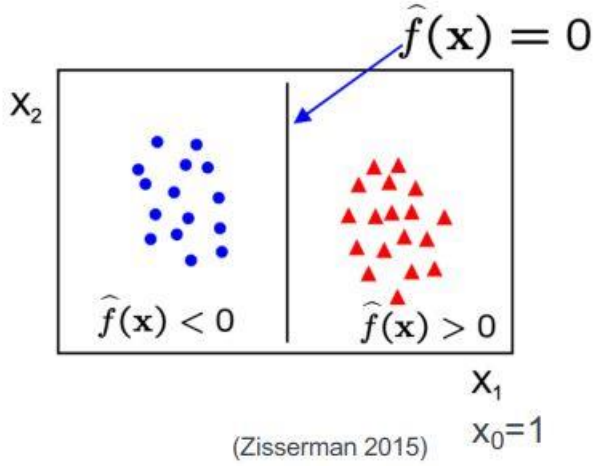

Figure 2: Support Vector Machine for classification of linearly seperable data.

An example of a linear separation in two dimensions can be seen in *Figure 2.*
Even though the separator can be found using the Perceptron algorithm, the Support Vector Machine algorithm is about finding the best hyperplane, here is a line, that separates the classes.

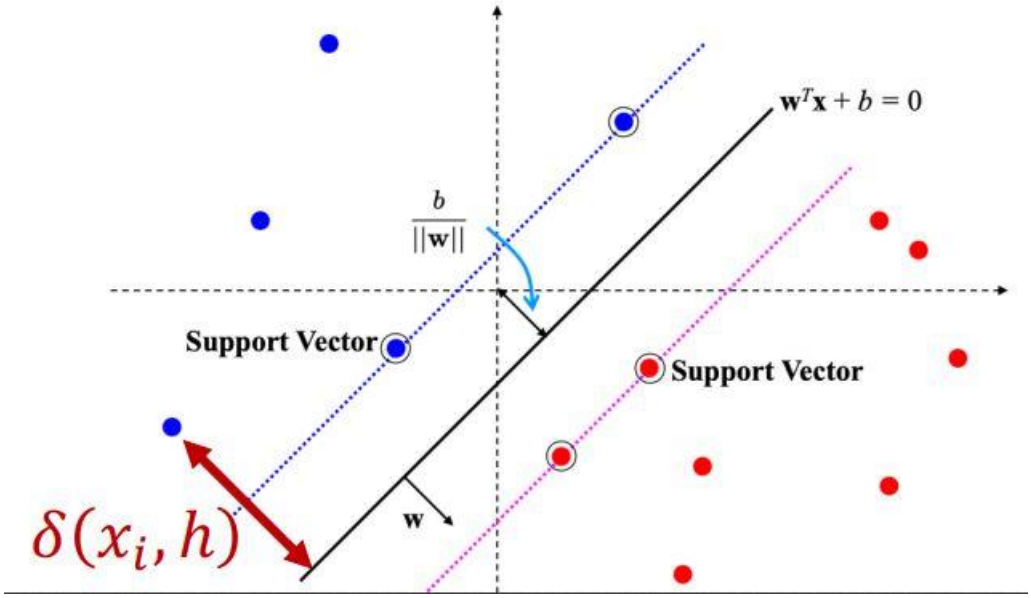

Figure 3: Support Vector Machine optimization.

In *Figure 3,* there are three lines that separate the classes. Assume the line in the middle to be the hyperplane h. Let the distance of each point to the hyperplane h to be $\delta(x_i, h)$. The distance can be computed as:

$$\delta(x_i, h) = \frac{y_i(w^T x_i + b)}{|w|}$$

The distance needs to be maximized to find the best hyperplane. To maximize this expression, we need to maximize $\frac{2}{|w|}$. The weight vector we get by the maximum value of $\frac{2}{|w|}$ is the best weight vector/hyperplane. In short, if we define the distance from the separating hyperplane to the nearest expression vector as the margin of the hyperplane, then the SVM selects the maximum margin separating hyperplane.

It is not always this easy to find the best hyperplane. In cases like *Figure 4*, we need to define a parameter called the soft margin. The soft margin parameter specifies a trade-off between hyperplane violations and the size of the margin. Here, the data is linearly separable but either there will be outliers, or the hyperplane will not be optimal as the best hyperplane. That is why the soft margin parameter is needed.

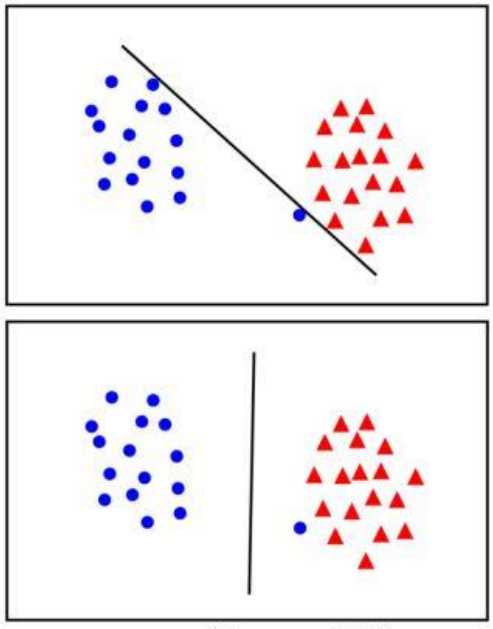

Figure 4: SVM for classification of linearly separable data that needs a soft margin.

SVM needs a kernel function for the classification. Kernel functions can change according to the classification problem and one can build their own kernel for their own problem [11]. Different kernel functions can be linear, polynomial, Gaussian, graph and string kernels. For this study, because the dataset is linearly separable and have 2 labels, a "linear kernel" will be used.

3.2 Dataset Creation and Description

The dataset for the study was provided from the website "Kaggle" under the name of "Song Lyrics Dataset" [12]. There were song lyrics of 21 artists and each of the artists' songs was in different CSV files. To create the complete dataset for the study, I imported the CSV files for 20 artists. I avoided one of the artists because the language of their songs was not English.

After importing the separate CSV files, I merged them together and turned the combined dataset into a pandas Data Frame so that it would be easier to operate on as seen in *Figure 5*.

In [4]:
```
#Importing the song lyrics of artists and combining them
df_ar = pd.read_csv("C:/Users/Lenovo/Downloads/archive/csv/ArianaGrande.csv")
df_be = pd.read_csv("C:/Users/Lenovo/Downloads/archive/csv/Beyonce.csv")
df_bi = pd.read_csv("C:/Users/Lenovo/Downloads/archive/csv/BillieEilish.csv")
df_ca = pd.read_csv("C:/Users/Lenovo/Downloads/archive/csv/CardiB.csv")
df_ch = pd.read_csv("C:/Users/Lenovo/Downloads/archive/csv/CharliePuth.csv")
df_co = pd.read_csv("C:/Users/Lenovo/Downloads/archive/csv/ColdPlay.csv")
df_dr = pd.read_csv("C:/Users/Lenovo/Downloads/archive/csv/Drake.csv")
df_du = pd.read_csv("C:/Users/Lenovo/Downloads/archive/csv/DuaLipa.csv")
df_ed = pd.read_csv("C:/Users/Lenovo/Downloads/archive/csv/EdSheeran.csv")
df_em = pd.read_csv("C:/Users/Lenovo/Downloads/archive/csv/Eminem.csv")
df_ju = pd.read_csv("C:/Users/Lenovo/Downloads/archive/csv/JustinBieber.csv")
df_ka = pd.read_csv("C:/Users/Lenovo/Downloads/archive/csv/KatyPerry.csv")
df_kh = pd.read_csv("C:/Users/Lenovo/Downloads/archive/csv/Khalid.csv")
df_la = pd.read_csv("C:/Users/Lenovo/Downloads/archive/csv/LadyGaga.csv")
df_ma = pd.read_csv("C:/Users/Lenovo/Downloads/archive/csv/Maroon5.csv")
df_ni = pd.read_csv("C:/Users/Lenovo/Downloads/archive/csv/NickiMinaj.csv")
df_po = pd.read_csv("C:/Users/Lenovo/Downloads/archive/csv/PostMalone.csv")
df_ri = pd.read_csv("C:/Users/Lenovo/Downloads/archive/csv/Rihanna.csv")
df_se = pd.read_csv("C:/Users/Lenovo/Downloads/archive/csv/SelenaGomez.csv")
df_ta = pd.read_csv("C:/Users/Lenovo/Downloads/archive/csv/TaylorSwift.csv")
df_all = [df_ar, df_be, df_bi, df_ca, df_ch, df_ch, df_co, df_dr, df_du, df_ed, df_em, df_ju, df_ka,
          df_kh, df_la, df_ma, df_ni, df_po, df_ri, df_se, df_ta]
df = pd.concat(df_all) #Making the combined dataset a pandas dataframe
df.head()
```

Out[4]:

	Artist	Title	Album	Date	Lyric	Year	Unnamed: 0
0	Ariana Grande	thank u, next	thank u, next	2018-11-03	thought i'd end up with sean but he wasn't a m...	2018	NaN
1	Ariana Grande	7 rings	thank u, next	2019-01-18	yeah breakfast at tiffany's and bottles of bub...	2019	NaN
2	Ariana Grande	God is a woman	Sweetener	2018-07-13	you you love it how i move you you love it how...	2018	NaN
3	Ariana Grande	Side To Side	Dangerous Woman	2016-05-20	ariana grande nicki minaj i've been here all ...	2016	NaN
4	Ariana Grande	no tears left to cry	Sweetener	2018-04-20	right now i'm in a state of mind i wanna be in...	2018	NaN

Figure 5: Importing and merging the datasets.

It can be seen in *Figure 5* that there were unnecessary columns in the dataset such as “Year” and “Unnamed: 0”. The “Year” column is not needed on the data frame since there is already the “Date” column. Also, because it was all null values for the “Unnamed: 0” column, that column could be removed as well.

In [5]:
```
#Cleaning and organizing the dataset
del df["Unnamed: 0"]
del df["Year"]
df.drop(df.loc[df['Album'] == "Unreleased Songs"].index, inplace=True)
df = df.dropna()
```

Figure 6: Organizing the dataset.

To organize the dataset, unreleased songs of artists were also removed. There were some songs that were added by other people that were not originally by the artists in the dataset because the dataset could be updated after its first upload to Kaggle. I removed those songs and the songs that had null values for some rows to avoid any inconvenience that could happen in the study. The code for this part can be seen in *Figure 6*.

```
In [6]: df.groupby("Artist")["Title"].count()

Out[6]: Artist
        Ariana Grande     53
        Beyoncé           62
        Billie Eilish     42
        Cardi B           26
        Charlie Puth      54
        Coldplay          61
        Drake             73
        Dua Lipa          45
        Ed Sheeran        48
        Eminem            77
        Justin Bieber     54
        Katy Perry        52
        Khalid            32
        Lady Gaga         59
        Maroon 5          43
        Nicki Minaj       47
        Post Malone       42
        Rihanna           92
        Selena Gomez      44
        Taylor Swift     106
        Name: Title, dtype: int64
```

Figure 7: Information about the dataset.

After the organization on the dataset, the dataset contained songs from 20 artists and the number of songs for each artist can be seen in *Figure 7*. The artists chosen for the research are from different popular genres such as pop, rap, hip-hop, and rock. This is important for the dataset to be efficient for the machine learning model. The number of total songs in the dataset is 1112 with 10 columns that contain information about the songs.

As stated previously, to use SVM for classification, the dataset needs to be labelled. Using the VADER module in Python, I calculated the sentiment value of the song lyrics. Lyrics in the dataset were already fitted to be used with VADER. So, I did not have to work on cleaning and organizing the lyrics column of the dataset. VADER computes the negative, positive, neutral, and compound score for the given text data, which here it is the lyrics.

VADER takes several aspects into account to decide the compound score of the text data. These can be listed as [13]:

1. Punctuation: Punctuation could signal increased sentiment intensity. For example, VADER computes “Good!!!” as a more positive sentence than “Good”.
2. Capitalization: Capitalization could signal an emphasis on the emotion of the sentence, making a positive sentence have more of a positive sentiment and a negative sentence have more of negative sentiment.
3. Degree modifiers: Words like “really”, and “extremely” alter the sentiment intensity of the text.

4. Conjunctions: Words like “but” changes the trajectory of the sentence which needs to be taken into account for a fair analysis of the sentiment.
5. Preceding Tri-gram: VADER understands negated sentences to a certain level. A negated sentence would be “The food here isn’t really all that great”.
VADER also understands slang words and acronyms in the text which is an advantage considering that popular songs these days tend to have a lot of slang words.

```
In [6]:
#Getting the sentiment score using VADER module
analyzer = SentimentIntensityAnalyzer()
df['compound'] = [analyzer.polarity_scores(x)['compound'] for x in df['Lyric']]
df['neg'] = [analyzer.polarity_scores(x)['neg'] for x in df['Lyric']]
df['neu'] = [analyzer.polarity_scores(x)['neu'] for x in df['Lyric']]
df['pos'] = [analyzer.polarity_scores(x)['pos'] for x in df['Lyric']]

In [7]:
df.head()
```

Out[7]:

	Artist	Title	Album	Date	Lyric	compound	neg	neu	pos
0	Ariana Grande	thank u, next	thank u, next	2018-11-03	thought i'd end up with sean but he wasn't a m...	0.9998	0.061	0.517	0.422
1	Ariana Grande	7 rings	thank u, next	2019-01-18	yeah breakfast at tiffany's and bottles of bub...	0.9983	0.058	0.686	0.256
2	Ariana Grande	God is a woman	Sweetener	2018-07-13	you you love it how i move you you love it how...	0.9982	0.010	0.759	0.231
3	Ariana Grande	Side To Side	Dangerous Woman	2016-05-20	ariana grande nicki minaj i've been here all ...	-0.6497	0.067	0.868	0.065
4	Ariana Grande	no tears left to cry	Sweetener	2018-04-20	right now i'm in a state of mind i wanna be in...	0.7234	0.113	0.748	0.139

Figure 8: Data Frame with the sentiment scores.

I created columns for each of the scores as seen in *Figure 8*. However, only the compound score value is significant for deciding on the sentiment of the lyrics.
With the compound score interval given in *Figure 9* [13], I created a function to assign the sentiment of the songs in three classes: positive, negative, and neutral.

1. **positive sentiment**: `compound` score >= 0.05
2. **neutral sentiment**: (`compound` score > -0.05) and (`compound` score < 0.05)
3. **negative sentiment**: `compound` score <= -0.05

Figure 9: Value interval for the sentiments with VADER.

```
In [14]: 1 len(df[df['VADER sentiment'] == "Positive"]) #Number of positive songs
Out[14]: 758

In [17]: 1 len(df[df['VADER sentiment'] == "Negative"]) #Number of negative songs
Out[17]: 344

In [18]: 1 len(df[df['VADER sentiment'] == "Neutral"]) #Number of neutral songs
Out[18]: 10
```

Figure 10: Number of negative, positive, and neutral songs.

The dataset that contained 1112 in total turned out to have 758 positive, 344 negative and 10 neutral songs according to the evaluation of VADER which can be seen in *Figure 10.* The number of neutral songs in the dataset is not enough for a machine learning model to train properly. That is why neutral songs are removed from the dataset and classification is done only in two categories: positive and negative. Also, removing the neutral songs made the classification problem a linear classification.

After tidying the dataset as described in this section, I saved the dataset, which contains 1102 songs with positive and negative labels, into a CSV file to work with it for the machine learning model.

The number of songs for similar studies in this field differs from each other. While Xia et al. worked with 2,653 songs, Kim et al. worked with 425 songs. Although each study uses a different number of songs, the number of songs is important for the machine learning model to learn efficiently.

3.3 SVM for Classification

Firstly, I started by importing the dataset created before into the new Jupyter Notebook. I separated the data into train and test data. While the model will learn from the train data, its precision will be tested on the test data. Like most of the examples, I separated the data with an 80:20 ratio. This means that 80% of the dataset is for training, and 20% is set apart for testing.

```
In [5]: 1 len(training_data[training_data['VADER sentiment'] == "Positive"])
Out[5]: 608

In [6]: 1 len(training_data[training_data['VADER sentiment'] == "Negative"])
Out[6]: 274

In [7]: 1 len(testing_data[testing_data['VADER sentiment'] == "Positive"])
Out[7]: 150

In [8]: 1 len(testing_data[testing_data['VADER sentiment'] == "Negative"])
Out[8]: 70
```

Figure 11: Number of positive and negative songs in train and test datasets.

After seperating the dataset into train and test datasets, the ratio of positive and negative songs in these datasets needed to be similar. If the train dataset contained mostly positive or negative songs, it would not work good enough to classify the test dataset into categories. So, as seen on *Figure 11*, I checked how many positive and negative songs both of the datasets contained and the ratio was satisfying for the model to work.

Like most machine learning algorithms that also require the data to be numerical, SVM model requires vectors as input to work. Thus, the text data in the "Lyric" column needs to be turned into vectors. TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer is an algorithm used to transform text data into vectors and will be used for this study. There are other algorithms such as Bag of Words Model and Word Counts Model as well. Short descriptions about these models would be appropriate here.

Bag of Words Model: This model takes the unique words into a vocabulary list from the given documents and returns a vector for each document by checking with the vocabulary. If the word in the vocabulary list is also in the selected document, the value in the vector is 1. Otherwise, it is zero. This makes each document vector maintain the same length that of vocabulary length.

Word Counts Model: The model tokenizes the collection of documents and forms a vocabulary list, like the Bag of Words model. Then, the model uses this list to encode new documents. CountVectorizer function from the scikit-learn library in Python can be used for this algorithm.

TF-IDF Model: The model combines two concepts which are Term Frequency (TF) and Document Frequency (DF). Term Frequency is the number of how many times a term appears in a document. Document Frequency is the number of documents containing a specific term. While term frequency shows the significance of a specific term in a document, document frequency shows how common the terms are. This model differs from the other two because it highlights the use of frequent words in documents seperately. The model does this by firstly creating a frequency matrix where the columns are the unique terms throughout all documents and the rows are each of the documents. The values for the cells are 1 or 0,

according to whether that term is in the document or not. Inverse Document Frequency (IDF) is the weight of a term and calculated as [14]:

$$idf_i = \log(\frac{n}{df_i})$$

Here, idf_i is the IDF score for the term i, df_i is the number of documents containing the term i, and n is the total number of documents.
Then, the TF-IDF score is calculated as a multiplication of the term frequency matrix with its IDF, the formula is given by:

$$w_{i,j} = tf_{i,j} \times idf_i$$

Here, $w_{i,j}$ is the TF-IDF score of term i in document j, and $tf_{i,j}$ is the term frequency for term i in document j.
Using this formula, the model calculates the IDF vector and multiplies it with the TF matrix respectively. TfidfVectorizer function from the sci-kit learn library in Python.
The most appropriate model for this study from the models listed above is the TF-IDF weighting model because it performs significantly better than other methods for emotion classification of song lyrics [5].

```
In [9]:
1 #Using Tfidf model for vectorizing the lyrics
2 vectorizer = TfidfVectorizer(sublinear_tf=True, use_idf = True, stop_words = 'english')
3 train_vectors = vectorizer.fit_transform(training_data['Lyric'])
4 test_vectors = vectorizer.transform(testing_data['Lyric'])
```

Figure 12: Applying TF-IDF vectorizer on the datasets.

As seen in *Figure 12*, I vectorized the "Lyric" column of train and test datasets using the TfidfVectorizer function from the sci-kit learn library. "stop_words" attribution in the function has a significant effect on the performance of the SVM model. Stop words list is a set of commonly used words in any language. Since these words are the most common words in any language, they do not carry any sentiment in general. For a study in text mining, and more specifically sentiment analysis, we can focus on the important words that carry sentiment instead, if we get rid of the stop words. Some examples of stop words for English would be "a", "the", "is", and "are". There are many more stop words and by using this attribution in the function, the function got rid of the stop words and did not turn them into an element of the vectors. All of the songs in our dataset are in English, so I used the stop words of English. After the vectorization, the dataset was ready to be put into the SVM model.

```
In [9]:
import time
from sklearn import svm
from sklearn.metrics import classification_report
#Classification with SVM and kernel is linear
classifier_linear = svm.SVC(kernel='linear')
t0 = time.time()
classifier_linear.fit(train_vectors, training_data['VADER sentiment'])
t1 = time.time()
prediction_linear = classifier_linear.predict(test_vectors)
t2 = time.time()
time_linear_train = t1-t0
time_linear_predict = t2-t1
# results
print("Training time: %fs; Prediction time: %fs" % (time_linear_train, time_linear_predict))
report = classification_report(testing_data['VADER sentiment'], prediction_linear, output_dict=True)
print('positive: ', report['Positive'])
print('negative: ', report['Negative'])
```

Figure 13: SVM model code.

In the code seen in *Figure 13*, we are using a linear kernel to perform SVM classification. This code chunk fits the vectors obtained from the lyrics with the "VADER sentiment" column on the training dataset and uses it to predict the "VADER sentiment" column for the test dataset, again by using the vectors obtained. To see the results, I first used the time() function to see the time the model trains and predicts. Then, I used the classification_report() function of the sci-kit learn library to get a text report showing the main classification metrics. The output of the code chunk can be seen in *Figure 14*.

Training and predicting time for the model turned out to be rather short, which is a good thing about the dataset and the model. Moreover, as seen in *Figure 14*, the main classification metrics are Precision, Recall, F1-Score, and Support. Precision, also called positive predictive value, shows how precise the model was when predicting the label. Recall, also known as sensitivity, is the fraction of relevant instances to all instances. So, precision and recall both show the relevance of the model. While F1-Score is the harmonic mean of precision and recall, support is the number of samples of the true response that lies in each class of target values.

```
Training time: 0.516040s; Prediction time: 0.147183s
positive:  {'precision': 0.8333333333333334, 'recall': 0.9666666666666667, 'f1-score': 0.8950617283950617, 'support': 150}
negative:  {'precision': 0.8913043478260869, 'recall': 0.5857142857142857, 'f1-score': 0.7068965517241379, 'support': 70}
```

Figure 14: Classification report of the model.

4. Discussion on Results Obtained

The success rate of the model can be seen in the classification report in *Figure 14*. The precision score for positive songs is 83% and for negative songs, it is 89%. Furthermore, the recall values for positive and negative songs are 96% and 58%, respectively. The recall value for negative songs can be lower because there were fewer negative songs in the dataset. Compared to other work done in the field, these results are considered successful. Overall, the f1-score being 89% for positive songs and 70% for negative songs shows that the SVM algorithm is well suited for this classification.

5. Conclusion and Future Work

Sentiment analysis has become a popular area of NLP. This study also focused on sentiment analysis applications on song lyrics. The steps of this study can be summarised as labeling song lyrics that are from different genres of music with a lexicon-based module in Python and then, applying SVM, which is a machine learning algorithm, to see the classification success of the algorithm. Since different techniques for sentiment analysis were used together, the method for this study is a hybrid technique and from the results obtained, it can be said that by using machine learning algorithms with lexicon-based algorithms, efficient results can be obtained in the field of sentiment analysis of song lyrics.

For further improvement, the study can focus on doing the classification with more labels than just positive and negative. So, seeing whether using this method with a bigger dataset and more labels perform good or not could be more comprehensive research. Also, the improvements in this field look like deep learning algorithms are going to be more popular in studies like this. So, using deep learning algorithms instead of machine learning algorithms for sentiment analysis of song lyrics can be the next level in the field.

6. References

[1] Madhoushi, Z., Hamdan, A. R., & Zainudin, S. (2015, July). Sentiment analysis techniques in recent works. In 2015 science and information conference (SAI) (pp. 288-291). IEEE.

[2] Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

[3] Çano, E., & Morisio, M. (2017, March). Moodylyrics: A sentiment annotated lyrics dataset. In Proceedings of the 2017 International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence (pp. 118-124).

[4] Napier, K., & Shamir, L. (2018). Quantitative sentiment analysis of lyrics in popular music. Journal of Popular Music Studies, 30(4), 161-176.

[5] Kim, M., & Kwon, H. C. (2011, November). Lyrics-based emotion classification using feature selection by partial syntactic analysis. In 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence (pp. 960-964). IEEE.

[6] Xia, Y., Wang, L., & Wong, K. F. (2008). Sentiment vector space model for lyric-based song sentiment classification. International Journal of Computer Processing Of Languages, 21(04), 309-330.

[7] He, H., Jin, J., Xiong, Y., Chen, B., Sun, W., & Zhao, L. (2008, December). Language feature mining for music emotion classification via supervised learning from lyrics. In International Symposium on Intelligence Computation and Applications (pp. 426-435). Springer, Berlin, Heidelberg.

[8] Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Sentiment classification using machine learning techniques. arXiv preprint cs/0205070.

[9] Chen, Y., & Zhang, Z. (2018, May). Research on text sentiment analysis based on CNNs and SVM. In 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA) (pp. 2731-2734). IEEE.

[10] Sentiment Analysis using SVM. Access address https://medium.com/@vasista/sentiment-analysis-using-svm-338d418e3ff1

[11] Staab, S., Baier, A., Hedeshy, R., & Hager, J. (2021). Support Vector Machines. Universität Stuttgart.

[12] Song Lyrics Dataset. Access address https://www.kaggle.com/datasets/deepshah16/song-lyrics-dataset

[13] VADER Sentiment Analysis. Access address https://github.com/cjhutto/vaderSentiment

[14] TF-IDF Simplified. Access address https://towardsdatascience.com/tf-idf-simplified-aba19d5f5530