SUBMISSION: SENTIMENT ANALYSIS REPORT

This document is a report on the developed Sentiment Analysis Model for text classification.

40524216
Keenan Bernard

Contents

Version History

Version	Date	Summary Of changes
0.1	November 22, 2021	Initial Creation
0.2	November 25, 2021	Text Classification, Model Diagram & Choice of Evaluation
0.3	November 26, 2021	Conclusion

Sentiment Analysis Model

The developed Sentiment Analysis Model analyzes a data set of reviews and its positive and negative attributes. The Sentiment Analysis Model then compares it against a predefined list of positive and negative keywords (extracted features) to normalized and define its state in terms of polarity in sentiment. Through training, this model can classify positive and negative keywords to predict the sentiment of the text.

Text Representation

The corpus provided included numerous reviews and the objective is to accurately predict the sentiment of these reviews based on their composition. The Text Representation Technique chosen for this model is Term Frequency-Inverse Document Frequency (TF-IDF) which help to convert text sentences into numeric vectors that are easily readable by the machine to implement NLP. This technique performs feature extraction using the most relevant words of a document as keywords. TF-IDF is a measure used in the field of machine learning to quantify the importance or relevance of string representations in a document amongst a corpus (see *Table 2* below using example documents "1" and "2"). Thus, each document in the corpus would have its vector, and the vector would have a TF-IDF score for every single word in the entire collection of documents. The motivation of TF-IDF is to statistically measure how relevant a word is in a collection of documents. TF-IDF works by multiplying two metrics, a raw count of times the word appears in a document (TF) and the inverse document frequency of words across a set of documents (IDF). IDF is calculated by taking the total number of documents and dividing it by the number of documents that contain the derived word. In short, TF-IDF is used to weigh down the frequent terms while scaling up the rare ones.

IDF formula:	**TF-IDF formula:**
idf(t, D) = log [n / df(t)] + 1	tf-idf(t, d, D) = tf(t, d) * idf(t, D)

Table 1

Word	**TF**		**IDF**	**TF*IDF**	
	1	**2**		**1**	**2**
The	1/7	1/7	Log(2/2) = 0	0	0
Laptop	1/7	0	Log(2/1) = 0.3	0.043	0
Amplifier	0	1/7	Log(2/1) = 0.3	0	0.043
Is	1/7	1/7	Log(2/2) = 0	0	0
Placed	1/7	1/7	Log(2/2) = 0	0	0
On	1/7	1/7	Log(2/2) = 0	0	0
The	1/7	1/7	Log(2/2) = 0	0	0
Floor	0	1/7	Log(2/1) = 0.3	0	0.043
Desk	1/7	0	Log(2/1) = 0.3	0.043	0

Table 2 1=”The Laptop is placed on the desk”; 2 = “The Amplifier is placed on the floor”

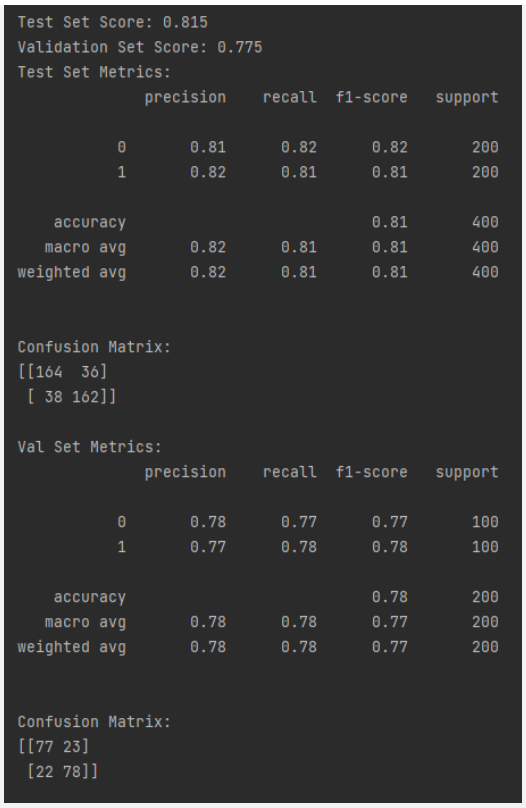
From Table 1 it is shown that the TF-IDF score of common words is 0, which indicates that these words are not significant and won’t be utilized for the analysis.

The developed sentiment analysis model utilizes TF-IDF to gather the most relevant features through vectorization, which is then fed into an ML algorithm. In the corpus provided there are words like articles (the), verbs (is) and prepositions (of) which do not help in discriminating the text. These are identified as stop words which are discarded along with special characters via preprocessing of the data. Preprocessing was done to clean the corpus provided before vectorization to improve the results of TF-IDF.

Parameter Tuning

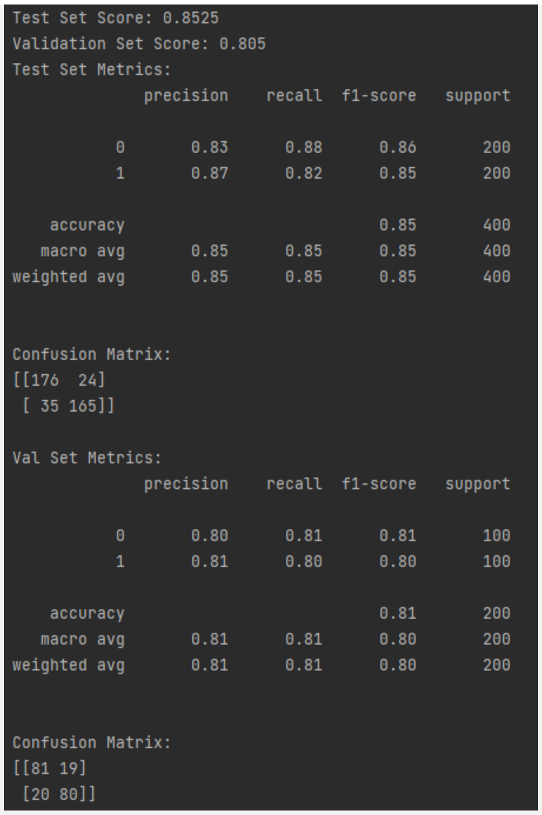
The implemented text representation technique makes use of different parameters for controlling input dimensions. The initial parameter used was the **max_features** parameter that only considers the top N features ordered by term frequency across the corpus; with further investigation, I came across the **min_df** parameter that sets the minimum frequency threshold of a word through the document for feature extraction. After compiling, the evaluation metrics showed that the **min_df** function produced better accuracy scores for the NLP model (see *Table 3* below). Tuning of the min_df parameter included modifying the frequency threshold ranging between 10 – 20; 15 resulting in better scores for the NLP. Parameter tuning was complete modifying the default **ngram_range** resulted in a minimum increase in performance. The performance of the altered parameters was measured on the “test” and “val” data sets showing better results in the confusion matrix.

MAX_FEATURES



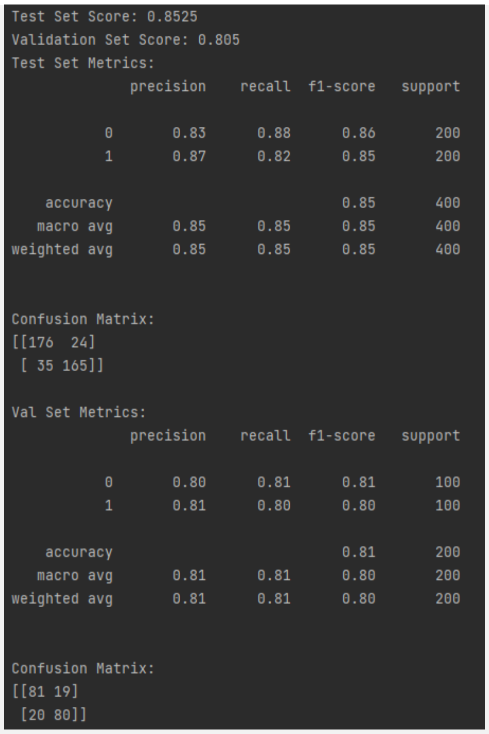
```
Test Set Score: 0.815
Validation Set Score: 0.775
Test Set Metrics:
              precision    recall  f1-score   support

           0       0.81      0.82      0.82       200
           1       0.82      0.81      0.81       200

    accuracy                           0.81       400
   macro avg       0.82      0.81      0.81       400
weighted avg       0.82      0.81      0.81       400


Confusion Matrix:
[[164  36]
 [ 38 162]]

Val Set Metrics:
              precision    recall  f1-score   support

           0       0.78      0.77      0.77       100
           1       0.77      0.78      0.78       100

    accuracy                           0.78       200
   macro avg       0.78      0.78      0.77       200
weighted avg       0.78      0.78      0.77       200


Confusion Matrix:
[[77 23]
 [22 78]]
```

MIN_DF



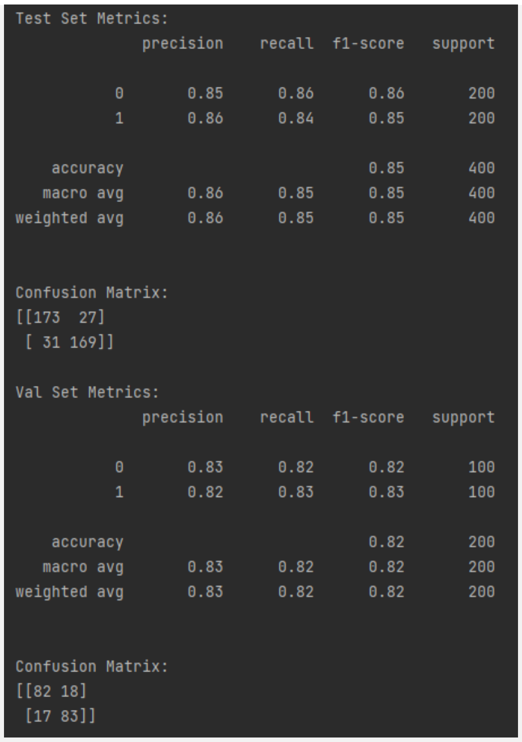
```
Test Set Score: 0.8525
Validation Set Score: 0.805
Test Set Metrics:
              precision    recall  f1-score   support

           0       0.83      0.88      0.86       200
           1       0.87      0.82      0.85       200

    accuracy                           0.85       400
   macro avg       0.85      0.85      0.85       400
weighted avg       0.85      0.85      0.85       400


Confusion Matrix:
[[176  24]
 [ 35 165]]

Val Set Metrics:
              precision    recall  f1-score   support

           0       0.80      0.81      0.81       100
           1       0.81      0.80      0.80       100

    accuracy                           0.81       200
   macro avg       0.81      0.81      0.80       200
weighted avg       0.81      0.81      0.80       200


Confusion Matrix:
[[81 19]
 [20 80]]
```

ngram_range (default (unigrams))



```
Test Set Score: 0.8525
Validation Set Score: 0.805
Test Set Metrics:
              precision    recall  f1-score   support

           0       0.83      0.88      0.86       200
           1       0.87      0.82      0.85       200

    accuracy                           0.85       400
   macro avg       0.85      0.85      0.85       400
weighted avg       0.85      0.85      0.85       400


Confusion Matrix:
[[176  24]
 [ 35 165]]

Val Set Metrics:
              precision    recall  f1-score   support

           0       0.80      0.81      0.81       100
           1       0.81      0.80      0.80       100

    accuracy                           0.81       200
   macro avg       0.81      0.81      0.80       200
weighted avg       0.81      0.81      0.80       200


Confusion Matrix:
[[81 19]
 [20 80]]
```

ngram_range (unigrams and bigrams)



```
Test Set Metrics:
              precision    recall  f1-score   support

           0       0.85      0.86      0.86       200
           1       0.86      0.84      0.85       200

    accuracy                           0.85       400
   macro avg       0.86      0.85      0.85       400
weighted avg       0.86      0.85      0.85       400


Confusion Matrix:
[[173  27]
 [ 31 169]]

Val Set Metrics:
              precision    recall  f1-score   support

           0       0.83      0.82      0.82       100
           1       0.82      0.83      0.83       100

    accuracy                           0.82       200
   macro avg       0.83      0.82      0.82       200
weighted avg       0.83      0.82      0.82       200


Confusion Matrix:
[[82 18]
 [17 83]]
```

Table 3 max_features=1400; min_df=15; ngram_range=(1,2)

Justification

The initial implementation of the Sentiment Analysis Model developed was using a Count Vectorizer (Bag of Words) technique which counts the frequency of words in a document. This model was chosen based on the size of the data sets provided, which weren't too large to make vectorization ineffective. This technique yielded subpar results; while seeking to improve the model's accuracy, I researched other vectorizers and came across TF-IDF. The TF-IDF vectorizer produced better results than the bag of words vectorizer. I concluded that the TF-IDF Vectorizer is better than Count Vectorizers (Bag of Words) because of its focus on the frequency of words present in the corpus and the importance of the word. The TF-IDF representation technique was chosen because it allows for the removal of words that are less important for the analysis, thus making the model building less complex by reducing the input dimensions and computational costs.

Text Classification

The developed Sentiment Analysis Model uses the Multinomial Naive Bayes (MNB) ML algorithm. The Naive Bayes (NM) algorithm is a linear classifier using the Bayes Theorem, referring to the strong independence assumptions in the model rather than the distribution of each feature. The Bayes Theorem helps to compute the conditional probabilities of occurrences of two events based on the joint probabilities of words and classes. The version of Naive Byes implemented for the Sentiment Analysis Model (Multinomial Naive Bayes) considers a feature vector where a given term represents the number of times it appears or very often using multinomial distribution.

Naive Bayes formula:

$$P(A|B) = [P(B|A) * P(A)] / P(B)$$

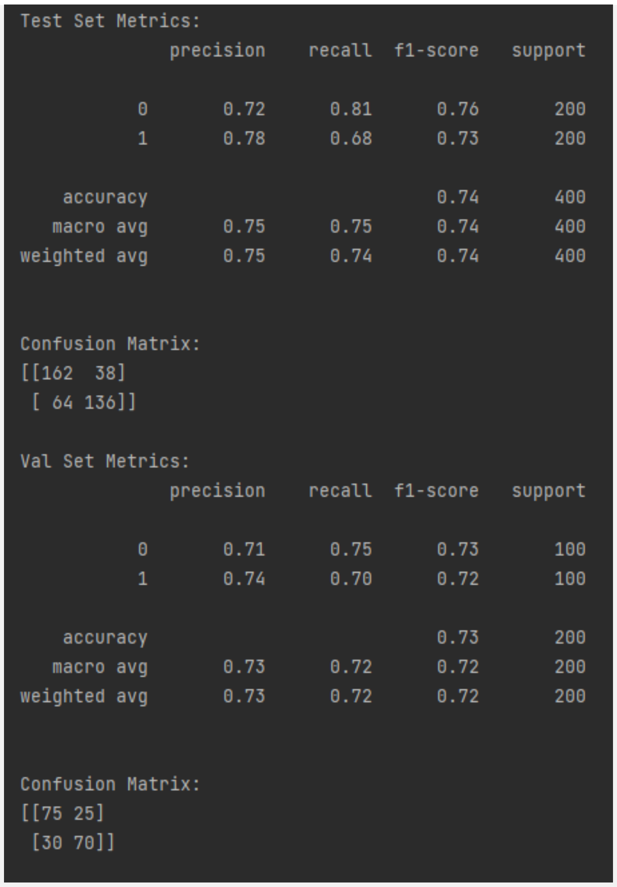
The probability of A if B is true (P(A|B)) is calculated by the probability of B if A is true (P(B|A)), times the probability of A being true (P(A)), divided by the probability of B being true (P(B)).

Using the vectors of TF-IDF which indicate the occurrence of relevant words of a document, the Multinomial Naive Bayes ML algorithm computes the probability of the text belonging to the classifier. The vectors of TF-IDF are then fitted to the ML model, training the model to accurately make predictions.

Justification

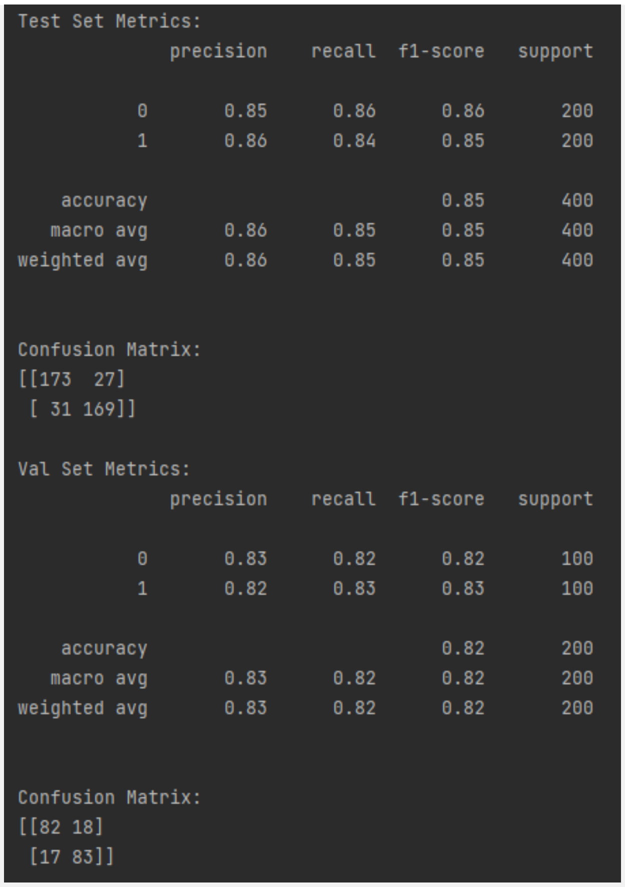
The initial research for a text classification model was done to find a model that has a low computational cost and is effective against the data sets provided. The Naive Bayes Algorithm was chosen because of its advantages as it is easy to implement and fast to predict the class of the test data set, it is better in most instances of text classification compared to models like logistic regression as it does not require large data sets for training. The first implementation of NB I was utilizing the Gaussian version paired with TF-IDF; results produced from the model were subpar when executed with the test and val data sets. Further research led me to the Multinomial version that produced better scores than the Gaussian version of Naive Bayes (see *Table 4* below). The Naive Bayes ML algorithm was chosen based on the task and data set (binary class) provided for its ease of implementation, known performance yielding great results for text classification and known popularity compared to other models like logistic regression or KNN.

Results of Gaussian Naive Bayes



```
Test Set Metrics:
              precision    recall  f1-score   support

           0       0.72      0.81      0.76       200
           1       0.78      0.68      0.73       200

    accuracy                           0.74       400
   macro avg       0.75      0.75      0.74       400
weighted avg       0.75      0.74      0.74       400


Confusion Matrix:
[[162  38]
 [ 64 136]]

Val Set Metrics:
              precision    recall  f1-score   support

           0       0.71      0.75      0.73       100
           1       0.74      0.70      0.72       100

    accuracy                           0.73       200
   macro avg       0.73      0.72      0.72       200
weighted avg       0.73      0.72      0.72       200


Confusion Matrix:
[[75 25]
 [30 70]]
```

Results of Multinomial Naive Bayes



```
Test Set Metrics:
              precision    recall  f1-score   support

           0       0.85      0.86      0.86       200
           1       0.86      0.84      0.85       200

    accuracy                           0.85       400
   macro avg       0.86      0.85      0.85       400
weighted avg       0.86      0.85      0.85       400


Confusion Matrix:
[[173  27]
 [ 31 169]]

Val Set Metrics:
              precision    recall  f1-score   support

           0       0.83      0.82      0.82       100
           1       0.82      0.83      0.83       100

    accuracy                           0.82       200
   macro avg       0.83      0.82      0.82       200
weighted avg       0.83      0.82      0.82       200


Confusion Matrix:
[[82 18]
 [17 83]]
```

Table 4 The change from Gaussian to Multinomial

Diagram of Sentiment Analysis Model

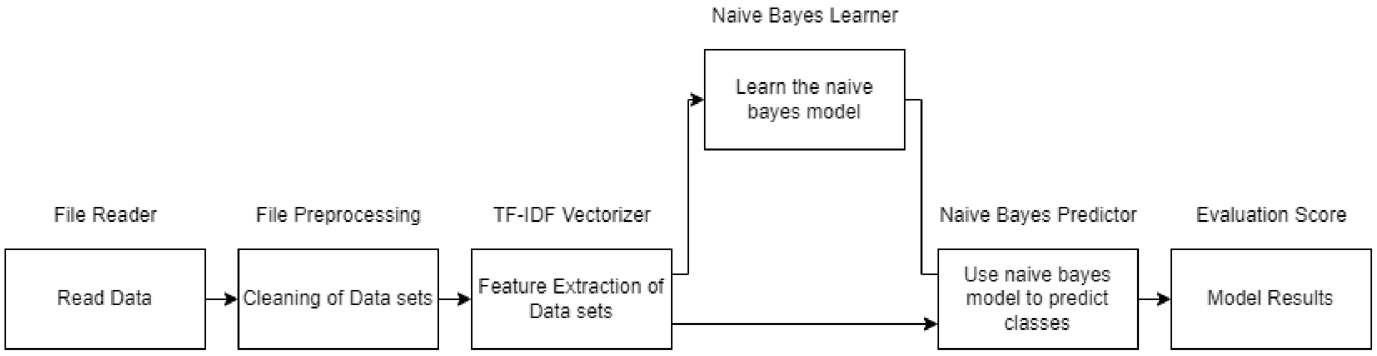


Figure 1

The Sentiment Analysis Model first reads and stores the provided data sets to their object. The stored data is then cleaned using the created “clean” function to reduce the size of English letters by converting all text to lower case, removing all special characters and punctuations, removing all non-alphabetical characters and discarding stop words. This preprocessing step is done to make the model more efficient and accurate by feeding cleaned data to the TF-IDF vectorizer. The TF-IDF vectorizer is then used for features extraction to build a vocabulary for the implemented machine learning algorithm. The extracted features are used to train the ML model to accurately predict the classes of new data. The Multinomial Navies Bayes algorithm is then able to predict the sentiment of the data sets provided (test & val) using the extracted features from the TF-IDF vectorizer. Accuracy scores are then produced to show the performance of the ML model with a confusion matrix to better understand the results.

Choice of Evaluation Metrics

To determine how well the implemented classifier performed, Accuracy scores were calculated using the ***sklearn.metrics*** library to output accuracy scores and a Confusion Matrix for evaluation. The classification report provides an Accuracy score of the model which is the ratio of correct predictions to the total number of input samples and is based on the Confusion Matrix. The accuracy metric was chosen to evaluate the implemented model as the provided data sets each had an equal number of samples belonging to each class providing truer results. The developed Sentiment Analysis Model was able to accurately predict the classification for 85% of the test data set while accurately predicting the classification for 82% of the val data set. The confusion matrix also allowed for a better understanding of predictions made by the Sentiment Analysis Model.

Reflection

The Sentiment Analysis Model examines a group of reviews to determine their positive and negative characteristics. The Sentiment Analysis Model then normalizes and defines its state in terms of polarity in sentiment using Multinomial Naïve Bayes using a defined list of vectors provided by TF-IDF. The Model was designed for text classification to accurately predict the sentiment of movie reviews. The Text Representation technique chosen is TF-IDF which is an advanced bow. Initial research of this technique with comparisons to others such as word2vec, a bag of words and many others led to the choice of this technique for my model after reading through various articles and evaluating it against my initial choices that were learned through the teaching modules. The ML model selection was done with minimal prior knowledge; Naive Bayes was selected after researching and evaluating models previously learned. Despite the disadvantages learned of the representation technique and classification model, the developed analysis model produced results that were exceptional to me. The developed model has its flaws however it does produce great results for the analysis task at hand while keeping computational costs to a minimum.

References

Huilgol, H.P. (2020). Quick Introduction to Bag-of-Words (BoW) and TF-IDF for Creating Features from Text.

https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/

Chandran, C.S. (2020). Introduction to Text Representations for Language Processing — Part 1.

https://towardsdatascience.com/introduction-to-text-representations-for-language-processing-part-1-dc6e8068b8a4

Marcinczuk, M, Gniewkowski, G, Walkowiak, T, Bedkowski, M (2021). Text Document Clustering: Wordnet vs. TF-IDF vs. Word Embeddings.

https://aclanthology.org/2021.gwc-1.24.pdf

Vadapalli, V.P. (2021). Naive Bayes Explained: Function, Advantages & Disadvantages, Applications in 2021.

https://www.upgrad.com/blog/naive-bayes-explained/#Disadvantages

Chauhan, G. (2018). All about Naive Bayes

https://towardsdatascience.com/all-about-naive-bayes-8e13cef044cf

Russell, S, Norvig, P (2010) Artificial Intelligence: A Modern Approach, Prentice Hall

Shriram, S (2021). Naive Bayes Explained: Function, Advantages & Disadvantages, Applications in 2021.

https://www.upgrad.com/blog/multinomial-naive-bayes-explained/