**2020-2021**

**BANGALORE NORTH UNIVERSITY**



**A PROJECT REPORT ON**

**“SENTIMENT ANALYSIS ON HOTEL REVIEWS”**

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE **6TH SEMESTER**

PROJECT WORK OF

BACHELOR OF COMPUTER APPLICATIONS

SUBMITTED BY

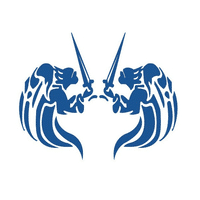
**KADUM KOMUT(R1818823)**

UNDER THE GUIDANCE OF

BHARGAVI K

DEPARTMENT OF BCA

KRUPANIDHI DEGREE COLLEGE, BANGALORE-35



KRUPANIDHI DEGREE COLLEGE

#12/CHIKKABELLANDUR(V), CARMELARAM(P), VARTHUR(H)

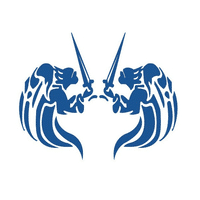
BANGALORE-560035

KARNATAKA.INDIA

KRUPANIDHI DEGREE COLLEGE

NO.12/1, CHIKKABELLANDUR, CARMELARAM POST

VARTHUR(H), BANGALORE-35



**CERTIFICATE**

This is to certify that the major project work entitled

**“SENTIMENT ANALYSIS ON HOTEL REVIEWS”**

Has been carried out by

**KADUM KOMUT (R1818823)**

In partial fulfillment of the requirement for the award of

Bachelor of Computer Applications

Degree by the Bangalore North University during the academic year 2020-2021

BHARGAVI K Mrs. ROOPINI Prof P.M Shyjan Dr. Badrunnisa S

GUIDE NAME H.O.D DEAN PRINCIPAL

EXAMINERS: DATE OF EXAMINATION

1. …………………………..
2. …………………………..

**ACKNOWLEDGEMENT**

We are thankful to God almighty for helping us to complete our project successfully. We would like to express our deep sense of gratitude and sincere thanks to our department of computer application. We would like to express our deep sense of gratitude to Prof. Dr.Suresh Nagpal, chairman of, Krupanidhi Group of Institutions, and Ms.Geeta Nagpal, vice-chairman, Krupanidhi Group of Institutions.

We express our sincere thanks to our dean sir Prof . P.M. Shyjan, Dr. Badrunnisa S, principal, Krupanidhi Degree College, for providing us the facilities. We are whole downheartedly thankful to Asst. Prof. Miss Roopini, Head of the Bachelor of Computer Application and our guide whose encouragement and best wishes provided the impetus for this project. We would like to thank my faculty guide, Miss Bhargavi R.

For All her valuable inputs and consult support towards us throughout our institutional training period and providing us an opportunity to learn outside the classroom and constantly motivating me to give our best. It was a truly wonderful learning experience.

We are also thankful for and fortunate enough to get constant support and guidance from all teaching staff of the department of BCA which helped us in completing our project work. Also, we would like to extend our sincere regards to all the non-teaching staff of the department of computer science for their timely support.

**CHAPTER 1**

**INTRODUCTION**

Natural language processing (NLP) is the intersection of computer science, linguistics, and [machine learning](https://builtin.com/data-science/introduction-to-machine-learning). The field focuses on communication between computers and humans in natural language and NLP is all about making computers understand and generate human language. [Applications of NLP](https://builtin.com/artificial-intelligence/natural-language-processing-examples-applications) techniques include voice assistants like Amazon's Alexa and Apple's Siri, but also things like machine translation and text-filtering.

NLP has heavily benefited from recent advances in machine learning, especially from deep learning techniques. The field is divided into three parts:

* **Speech Recognition:** The translation of spoken language into text.
* **Natural Language Understanding:** The computer’s ability to understand what we say.
* **Natural Language Generation:** The generation of natural language by a computer.

Sentiment analysis, also referred to as opinion mining, is a sub-machine learning task where we want to determine which is the general sentiment of a given document. Using machine learning techniques and natural language processing we can extract the subjective information of a document and try to classify it according to its polarity such as positive, neutral, or negative. It is a really useful analysis since we could possibly determine the overall opinion about selling objects, or predict stock markets for a given company, if most people think positively about it, possibly its stock markets will increase, and so on. Sentiment analysis is far from being solved since the language is very complex (objectivity/subjectivity, negation, vocabulary, grammar,...) but it is also why it is very interesting to work on.

In this project, I choose to try to classify reviews from Hotel Websites into “positive” or “negative” sentiment by building a model based on probabilities. Hotels play a crucial role in traveling and with the increased access to information new pathways of selecting the best ones emerged. With this dataset, consisting of 20k reviews collected from Kaggle, you can explore what makes a great hotel and maybe even use this model in your travels!.

**CHAPTER 2:**

**LITERATURE SURVEY**

In this project phase, we will study some of the machine learning algorithms based on a mathematical model which classifies a given dataset into one of the categories namely positive and negative in my case.

In 2019, Saad and Yang [1] have aimed for giving a complete tweet sentiment analysis based on ordinal regression with machine learning algorithms. The suggested model included preprocessing reviews as the first step and with the feature extraction model, an effective feature was generated. The test results have shown that the suggested model has attained the best accuracy, and also DTs were performed well when compared to other methods.

In 2019, Afzaal has recommended a novel approach of aspect-based sentiment classification, which recognized the features in a precise manner and attained the best classification accuracy. Moreover, the scheme was developed as a mobile application, which assisted the tourists in identifying the best hotel in the town, and the proposed model was analyzed using real-world datasets. The results have shown that the presented model was effective in both recognition as well as classification.

In 2020, Kumar et al. [6] have presented a hybrid deep learning approach named ConVNet-SVMBoVW that deals with the real-time data for predicting the fine-grained sentiment. To measure the hybrid polarity, an aggregation model was developed. Moreover, SVM was used for training the BoVW to forecast the sentiment of visual content. Finally, it was concluded that the suggested ConvNet-SVMBoVW was outperformed by the conventional models.

Kouloumpis et al. [51] highlighted the efficacy of existing lexical resources and linguistic features for conduct SA on Twitter messages and similar micro-blogging posts. The researchers postulated that micro-blogging is more relevant and appropriate in comparison to part-of-speech features and those belonging to existing sentiment lexicon. The authors concluded that inclusion of microblogging features is not likely to augment training data. Hybrid classification was also found to be more relevant and showed promising results in another research [52] involving rule-based classification and supervised learning processes. In the context of natural language processing (NLP), SA has grown in stature as one of the most researched topics since the turn of the century [53, 54]. Researchers have been constantly examining sentences and assorted types to streamline SA methods. Concept level SA system (psenti) as has been postulated by Mudinas et. al. have been shown to be more effective than lexicon-based system [55]. Results obtained from their research reveal that hybrid approach is considerably more efficient and returned more accurate results. In their research [12], Tripathy et. al. proposed four ML algorithms for sentiment classification. These were Naïve Bayes, Maximum Entropy, Stochastic Gradient Descent and Support Vector Machine. Their research demonstrated that accuracy can be achieved through progressive classification. Their research was conducted on popular movie review website – IMDB. Deploying an n-gram approach, the authors were able to produce consistent high accuracy using a combination of TF-IDF and count vectorizer technique. To counter the problem of words and punctuation symbols that although were understandable to humans, had no formal definition in the English lexicon, the authors developed a new list of such words to assist in SA. Mitchel et al. modeled sentiment detection in their research paper. The focus of their work was to show applicability of sentiment detection as a problem that involved sequence tagging [56]. Their work was expanded upon by later research work [57] that examined embedding of words and automated combination of features through neural networks. Arun et. al. proceeded to conduct SA on tweets involving demonetization in Indian economy. Their research approach [58] was to extract data from Twitter and convert such into text. This text was to act as input dataset. SA was then performed following removal of stop words so that determination of polarity of the words could be carried out and the actual tweets, therefore, could be identified as either positive or negative. A new method was postulated for SA on demonetization and its subsequent effects and the wave of public opinion that it had unleashed. Bigrams, data cleaning, polarity, sentiment scores and graphical methods were all used in the research. In conducting a comparative analysis of assorted approaches for SA and topic detection, researchers [59] examined a collection of tweets in Spanish. Lemmatizers, stemmers, n-grams, negations, valence shifters, Twitter hashtags and semantics were explored and presented in a detailed study. The authors opined that lack of context and extreme short nature of text involved, tweets are difficult to make clear assessment of. In another research study involving e-commerce and online product reviews, the authors [60] stated that public opinions and those of buyers of products sold on online retail platforms would have far-reaching significance in terms of profitability and economic viability. As such, entrepreneurs and enterprises invest in opinion mining and SA to work out ways to stay ahead of the competition. The authors use fuzzy rulebased systems (FRBS) with Mamdani and Takagi Sugeno Kang (TSK) models deployed in FRBAS R package. The authors go on to compare these models with alternate classification techniques and grade them on the lines of precision, recall, f-measure, performance and accuracy. Exploring the feasibility and efficacy of SA in the context of Arabic tweets, the authors [61] have observed that there is a significant lack of resources to carry out SA in Arabic. The authors conducted their research using Jordanian Arabic tweets and examined such using a range of supervised ML approaches. Their research revealed that SVM classifier with TF-IDF and bigrams was comparatively better than Naïve Bayesian one. For their experiment, the authors had collected close to a couple of thousand Arabic tweets posted in Jordanian dialect. The two ML algorithms of SVM and NB were then run using a range of ngrams with varying weighting schemes and following application of stemming techniques. In conclusion, the authors observed that SVM classifier using stemmer and TF-IDF weighing than any other known Arabic SA research observations. A novel approach was adopted by Socher et. al. when the authors [62] proposed a semisupervised recursive auto-encoding approach for forecasting sentiment distributions without the benefit of sentiment lexica or rules that involved shifting of polarity. In their paper [63], the authors proposed recursive neural tensor network (RNTN) to analyse and discern phrases and sentences of varying length. In another paper [64], an alternative form of sentiment classification was proposed to analyse sentences of different lengths and words. While hyper parameter tuning for word vectors has been the research of choice [65], deep recursive neural network (DRNN) has been selected by the authors [66] for conducting sentiment classification assignments. Interestingly, the authors [1] postulate that for obtaining improved classification results, larger datasets involving crowd sourcing and semi-supervised learning held far greater potential. The authors used Naïve Bayes, Support Vector Machine and K-Nearest Neighbour to conduct comparison tests. Their experiments revealed that while SVM was most useful in turning in highest precision, it was KNN that achieved highest recall consistently.

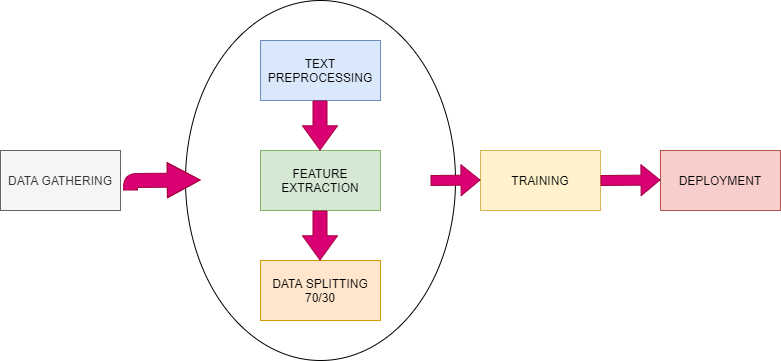
**2.1 Existing system**

Existing approaches to sentiment analysis can be grouped into three main categories: knowledge-based techniques, statistical methods, and hybrid approaches.[[42]](https://en.wikipedia.org/wiki/Sentiment_analysis#cite_note-%E2%80%9CCambria-42) Knowledge-based techniques classify text by affect categories based on the presence of unambiguous effect words such as happy, sad, afraid, and bored.Statistical methods leverage elements from [machine learning](https://en.wikipedia.org/wiki/Machine_learning) such as [latent semantic analysis](https://en.wikipedia.org/wiki/Latent_semantic_analysis), [support vector machines](https://en.wikipedia.org/wiki/Support_vector_machines), "[bag of words](https://en.wikipedia.org/wiki/Bag_of_words)", "[Pointwise Mutual Information](https://en.wikipedia.org/wiki/Pointwise_Mutual_Information)" for Semantic Orientation,[[5]](https://en.wikipedia.org/wiki/Sentiment_analysis#cite_note-Turney02-5) and [deep learning](https://en.wikipedia.org/wiki/Deep_learning). More sophisticated methods try to detect the holder of sentiment (i.e., the person who maintains that affective state) and the target (i.e., the entity about which the effect is felt).[[45]](https://en.wikipedia.org/wiki/Sentiment_analysis#cite_note-Kim+Hovy06-45) To mine the opinion in [context](https://en.wikipedia.org/wiki/Context_(language_use)) and get the feature about which the speaker has opined, the grammatical relationships of words are used. Grammatical dependency relations are obtained by deep parsing of the text.[[46]](https://en.wikipedia.org/wiki/Sentiment_analysis#cite_note-DeyHaque08-46) Hybrid approaches leverage both machine learning and elements from [knowledge representation](https://en.wikipedia.org/wiki/Knowledge_representation) such as [ontologies](https://en.wikipedia.org/wiki/Ontologies) and [semantic networks](https://en.wikipedia.org/wiki/Semantic_network) to detect semantics that is expressed subtly, e.g., through the analysis of concepts that do not explicitly convey relevant information, but which are implicitly linked to other concepts that do so.[[47]](https://en.wikipedia.org/wiki/Sentiment_analysis#cite_note-%E2%80%9CHussain-47)

**2.2 Proposed system**

We apply a simple mathematical machine learning classifier which is namely Naive Bayes and to classify the problem into binary classes such as positive or negative. The reason behind the algorithms is to make use of these probabilistic models to predict the needed outcome and the need to maintain the accuracy high with less training time and fewer datasets. Probabilistic models work charmingly, with fewer datasets to predict with great accuracy which any other algorithm would inefficiently perform

**CHAPTER 3**

**SYSTEM ARCHITECTURE**

**3.1 DATASET GATHERING**

The data has been collected from the Kaggle website which contains a huge number of datasets for different machine learning fields. As in the dataset’s official directory, The uploader has mentioned having collected the dataset through crawling the trip advisor and gathering a total of 20,000 reviews, and labeled the data with certain ratings ranging from 1 to 5.

The dataset, however, is imbalanced, and to balance the structure, some additional processing has been performed with the help of libraries such as pandas. The label with the ratings of 5 has been converted into a positive review and the label with the ratings of 1 and 2 has been converted into a negative review for the sole purpose of our classifier to work in a better way. The dataset contains two columns namely reviews and ratings. The reviews column contains reviews written by actual users for the specific hotel. For our use case only strict to sentiment analysis, the hotel information has been discarded and only the reviews and its corresponding ratings have been taken.

The dataset contains unnecessary words which do not put any weight into our sentiment analysis, For our algorithm to perform better, These noises have to be processed.

Some examples of datasets.



Table 2.1.1: Example of TripAdvisor dataset posts annotated with their corresponding sentiment,

**3.2 TEXT PREPROCESSING**

Text preprocessing is a method to clean the text data and make it ready to feed data to the model. Text data contains noise in various forms like emotions, punctuation, text in different cases. When we talk about Human Language then, there are different ways to say the same thing, And this is only the main problem we have to deal with because machines will not understand words, they need numbers so we need to convert text to numbers in an efficient manner.

In this project, Various pre-processing techniques have been used.

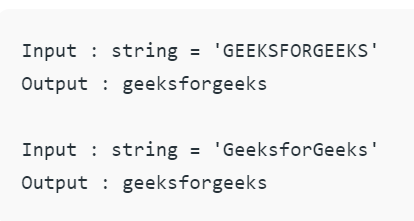
* **Remove punctuations:** It involves removing everything except for the numbers and alphabets for clearer context.

****

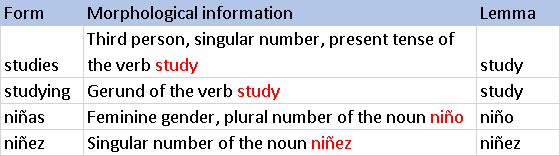
* **Remove Stopwords:** Stopwords are the most commonly occurring words in a text which do not provide any valuable information. stopwords like they, there, this, where, etc are some of the stopwords.

****

* **Lower:**  All the alphabets are lowercase in case any new input appears to be uppercase and the classifier could not predict.

****

* **Lemmatization:** Lemmatization is similar to stemming, used to stem the words into root words but differs in working. Actually, Lemmatization is a systematic way to reduce the words into their lemma by matching them with a language dictionary.

****

**3.2 FEATURE EXTRACTION**

Machine Learning algorithms learn from a predefined set of features from the training data to produce output for the test data. But the main problem in working with language processing is that machine learning algorithms cannot work on the raw text directly. So, we need some feature extraction techniques to convert text into a matrix(or vector) of features.

In this project, we use Countvectorizer or simply known as Bag of words. It is one of the most fundamental methods to transform tokens into a set of features. The BoW model is used in document classification, where each word is used as a feature for training the classifier.

**3.2.1 BAG OF WORDS**

A bag-of-words model, or BoW for short, is a way of extracting features from text for use in modeling, such as with machine learning algorithms. The approach is very simple and flexible, and can be used in a myriad of ways for extracting features from documents. A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

* A vocabulary of known words.
* A measure of the presence of known words.

It is called a “*bag*” of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

The intuition is that documents are similar if they have similar content. Further, that from the content alone we can learn something about the meaning of the document. The bag-of-words can be as simple or complex as you like. The complexity comes both in deciding how to design the vocabulary of known words (or tokens) and how to score the presence of known words.

## Example of the Bag-of-Words Model

Let’s make the bag-of-words model concrete with a worked example.

**Step 1: Collect Data**

Below is a snippet of the first few lines of text from the book “A Tale of Two Cities” by Charles Dickens, taken from Project Gutenberg.

* It was the best of times,
* it was the worst of times,
* it was the age of wisdom,
* it was the age of foolishness,

For this small example, let’s treat each line as a separate “document” and the 4 lines as our entire corpus of documents.

**Step 2: Design the Vocabulary**

Now we can make a list of all of the words in our model vocabulary.

The unique words here (ignoring case and punctuation) are:

* “it”
* “was”
* “the”
* “best”
* “of”
* “times”
* “worst”
* “age”
* “wisdom”
* “foolishness”

That is a vocabulary of 10 words from a corpus containing 24 words.

**Step 3: Create Document Vectors**

The next step is to score the words in each document.

The objective is to turn each document of free text into a vector that we can use as input or output for a machine learning model.

Because we know the vocabulary has 10 words, we can use a fixed-length document representation of 10, with one position in the vector to score each word.

The simplest scoring method is to mark the presence of words as a boolean value, 0 for absent, 1 for present.

Using the arbitrary ordering of words listed above in our vocabulary, we can step through the first document (“It was the best of times“) and convert it into a binary vector.

The scoring of the document would look as follows:

* “it” = 1
* “was” = 1
* “the” = 1
* “best” = 1
* “of” = 1
* “times” = 1
* “worst” = 0
* “age” = 0
* “wisdom” = 0
* “foolishness” = 0

As a binary vector, this would look as follows:

[1, 1, 1, 1, 1, 1, 0, 0, 0, 0]

The other three documents would look as follows:

* "it was the worst of times" = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]
* "it was the age of wisdom" = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]
* "it was the age of foolishness" = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]

All ordering of the words is nominally discarded and we have a consistent way of extracting features from any document in our corpus, ready for use in modeling.

New documents that overlap with the vocabulary of known words, but may contain words outside of the vocabulary, can still be encoded, where only the occurrence of known words are scored and unknown words are ignored.

You can see how this might naturally scale to large vocabularies and larger documents.

**Managing Vocabulary**

As the vocabulary size increases, so does the vector representation of documents.

In the previous example, the length of the document vector is equal to the number of known words.You can imagine that for a very large corpus, such as thousands of books, that the length of the vector might be thousands or millions of positions. Further, each document may contain very few of the known words in the vocabulary.

This results in a vector with lots of zero scores, called a sparse vector or sparse representation.

Sparse vectors require more memory and computational resources when modeling and the vast number of positions or dimensions can make the modeling process very challenging for traditional algorithms.

As such, there is pressure to decrease the size of the vocabulary when using a bag-of-words model.

There are simple text cleaning techniques that can be used as a first step, such as:

* Ignoring case
* Ignoring punctuation
* Ignoring frequent words that don’t contain much information, called stop words, like “a,” “of,” etc.
* Fixing misspelled words.
* Reducing words to their stem (e.g. “play” from “playing”) using stemming algorithms.

A more sophisticated approach is to create a vocabulary of grouped words. This both changes the scope of the vocabulary and allows the bag-of-words to capture a little bit more meaning from the document.

In this approach, each word or token is called a “gram”. Creating a vocabulary of two-word pairs is, in turn, called a bigram model. Again, only the bigrams that appear in the corpus are modeled, not all possible bigrams.

For example, the bigrams in the first line of text in the previous section: “It was the best of times” are as follows:

* “it was”
* “was the”
* “the best”
* “best of”
* “of times”

A vocabulary then tracks triplets of words is called a trigram model and the general approach is called the n-gram model, where n refers to the number of grouped words.

Often a simple bigram approach is better than a 1-gram bag-of-words model for tasks like documentation classification.

**Scoring Words**

Once a vocabulary has been chosen, the occurrence of words in example documents needs to be scored.

In the worked example, we have already seen one very simple approach to scoring: a binary scoring of the presence or absence of words.

Some additional simple scoring methods include:

* Counts. Count the number of times each word appears in a document.
* Frequencies. Calculate the frequency that each word appears in a document out of all the words in the document.

**Word Hashing**

You may remember from computer science that a hash function is a bit of math that maps data to a fixed size set of numbers.

For example, we use them in hash tables when programming where perhaps names are converted to numbers for fast lookup.

We can use a hash representation of known words in our vocabulary. This addresses the problem of having a very large vocabulary for a large text corpus because we can choose the size of the hash space, which is in turn the size of the vector representation of the document.

Words are hashed deterministically to the same integer index in the target hash space. A binary score or count can then be used to score the word.

This is called the “hash trick” or “feature hashing“.

The challenge is to choose a hash space to accommodate the chosen vocabulary size to minimize the probability of collisions and trade-off sparsity.

**TF-IDF**

A problem with scoring word frequency is that highly frequent words start to dominate in the document (e.g. larger score), but may not contain as much “informational content” to the model as rarer but perhaps domain specific words.

One approach is to rescale the frequency of words by how often they appear in all documents, so that the scores for frequent words like “the” that are also frequent across all documents are penalized.

This approach to scoring is called Term Frequency – Inverse Document Frequency, or TF-IDF for short, where:

* Term Frequency: is a scoring of the frequency of the word in the current document.
* Inverse Document Frequency: is a scoring of how rare the word is across documents.

The scores are a weighting where not all words are equally as important or interesting.

The scores have the effect of highlighting words that are distinct (contain useful information) in a given document.

**Limitations of Bag-of-Words**

The bag-of-words model is very simple to understand and implement and offers a lot of flexibility for customization on your specific text data.

It has been used with great success on prediction problems like language modeling and documentation classification.

Nevertheless, it suffers from some shortcomings, such as:

* **Vocabulary**: The vocabulary requires careful design, most specifically in order to manage the size, which impacts the sparsity of the document representations.
* **Sparsity**: Sparse representations are harder to model both for computational reasons (space and time complexity) and also for information reasons, where the challenge is for the models to harness so little information in such a large representational space.
* **Meaning**: Discarding word order ignores the context, and in turn meaning of words in the document (semantics). Context and meaning can offer a lot to the model, that if modeled could tell the difference between the same words differently arranged (“this is interesting” vs “is this interesting”), synonyms (“old bike” vs “used bike”), and much more.

**3.2.2 BAG OF WORDS WORKING**

Consider the documents-

* ‘This is first document’
* ‘This is the second document’
* ‘and this is the third’

Two steps are involved in Bag of words:

1. **FIT:** Learn a vocabulary dictionary of all tokens in the raw documents and returns the list of uniquely identified words present in the documents.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **and** | **document** | **first** | **is** | **second** | **the** | **third** | **this** |

1. **TRANSFORM:** Transform the documents to document-term matrix and return a sparse matrix with the word present in the document as 1 and not present as 0.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **and** | **document** | **first** | **is** | **second** | **the** | **third** | **this** |
| **document1** | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| **document2** | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| **document3** | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |

**3.3 DATASET SPLITTING**

One of the first decisions to make when starting a modeling project is how to utilize the existing data. One common technique is to split the data into two groups typically referred to as the *training* and *testing* sets[23](https://bookdown.org/max/FES/data-splitting.html#fn23). The training set is used to develop models and feature sets; they are the substrate for estimating parameters, comparing models, and all of the other activities required to reach a final model. The test set is used only at the conclusion of these activities for estimating a final, unbiased assessment of the model’s performance. It is critical that the test set not be used prior to this point. Looking at the test sets results would bias the outcomes since the testing data will have become part of the model development process.

How much data should be set aside for testing? It is extremely difficult to make a uniform guideline. The proportion of data can be driven by many factors, including the size of the original pool of samples and the total number of predictors. With a large pool of samples, the criticality of this decision is reduced once “enough” samples are included in the training set. Also, in this case, alternatives to a simple initial split of the data might be a good idea; see Section [3.4.7](https://bookdown.org/max/FES/resampling.html#inside-resampling) below for additional details. The ratio of the number of samples (nn) to the number of predictors (pp) is important to consider, too. We will have much more flexibility in splitting the data when nn is much greater than pp. However when nn is less than pp, then we can run into modeling difficulties even if nn is seemingly large.

There are a number of ways to split the data into training and testing sets. The most common approach is to use some version of random sampling. Completely random sampling is a straightforward strategy to implement and usually protects the process from being biased towards any characteristic of the data. However, this approach can be problematic when the response is not evenly distributed across the outcome. A less risky splitting strategy would be to use a *stratified* random sample based on the outcome. For classification models, this is accomplished by selecting samples at random *within* each class. This approach ensures that the frequency distribution of the outcome is approximately equal within the training and test sets. When the outcome is numeric, artificial strata can be constructed based on the quartiles of the data. For example, in the Ames housing price data, the quartiles of the outcome distribution would break the data into four artificial groups containing roughly 230 houses. The training/test split would then be conducted within these four groups and the four different training set portions are pooled together (and the same for the test set).

Splitting your dataset is essential for an unbiased evaluation of prediction performance. In most cases, it’s enough to split your dataset randomly into [three subsets](https://en.wikipedia.org/wiki/Training,_validation,_and_test_sets):

* The training set is applied to train, or fit, your model. For example, you use the training set to find the optimal weights, or coefficients, for [linear regression](https://realpython.com/linear-regression-in-python/), or [neural networks](https://en.wikipedia.org/wiki/Artificial_neural_network).
* The test set is needed for an unbiased evaluation of the final model. You shouldn’t use it for fitting or validation.

The dataset splitting ratio has been kept different for Naive Bayes

**3.3.1 NAIVE BAYES DATASET SPLITTING**

* **Training Set:** 80%
* **Test Set:** 20%

**3.4 MODEL TRAINING**

Model training is the phase in the data science development lifecycle where practitioners try to fit the best combination of weights and bias to a machine-learning algorithm to minimize a loss function over the prediction range. The purpose of model training is to build the best mathematical representation of the relationship between data features and a target label (in supervised learning [glossary link]) or among the features themselves (unsupervised learning [glossary link]). Loss functions [glossary link] are a critical aspect of model training since they define how to optimize the machine learning algorithms. Depending on the objective, type of data, and algorithm, data science practitioners use different types of loss functions. One of the popular examples of loss functions is Mean Square Error

Once we have applied the different steps of the preprocessing part, we can now focus on the machine learning part. There are three major models used in sentiment analysis to classify a sentence into positive or negative: and Naive Bayes. Let’s first introduce the Naive Bayes, which is well known for its simplicity and efficiency for text classification.

**3.4.1 NAIVE BAYES CLASSIFIER THEORY**

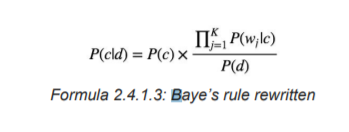
In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive)independence assumptions between the features. Naive Bayes classifiers are highly scalable, requiring several parameters linear in the number of variables (features/predictors) in a learning problem. Maximum Likelihood training can be done by evaluating a closed-form expression (a mathematical expression that can be evaluated in a finite number of operations), which takes linear time. It is based on the application of the Bayes rule given by the following formula:

P(C=c|D=d) = P(D=d|C=c) \* P(C=c) / P(D=d)

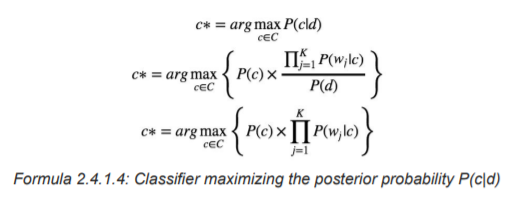
where D denotes the document and C the category (label), d and c are instances of D and C and P(D = d) = ∑ (D |C )P(C ). We can simplify this expression by,

P(c|d) = P(d|c)\*P(c)/P(d)

In our case, a Review d is represented by a vector of K attributes such as d = (w, w, ..., w ). Computing P(d|c) is not trivial and that's why the Naive Bayes introduces the assumption that all of the feature values wj are independent given the category label c. That is, for i =/ j , wi and wj are conditionally independent given the category label c. So the Bayes rule can be rewritten as



Based on this equation, maximum a posterior (MAP) classifier can be constructing by seeking the optimal category which maximizes the posterior P(c|d) :



Note that P(d) is removed since it is a constant for every category c.

There are several variants of Naive Bayes classifiers that are:

● The **Multi­variate Bernoulli Model**: Also called binomial model, useful if our feature vectors are binary (e.g 0s and 1s). An application can be text classification with a bag of words model where the 0s 1s are "word does not occur in the document" and "word occurs in the document" respectively.

● The **Multinomial Model**: Typically used for discrete counts. In-text classification, we extend the Bernoulli model further by counting the number of times a word $w\_i$ appears over the number of words rather than saying 0 or 1 if a word occurs or not.

● The **Gaussian Model**: We assume that features follow a normal distribution. Instead of discrete counts, we have continuous features.

For text classification, the most used and considered as the best choice is the Multinomial Naive Bayes.

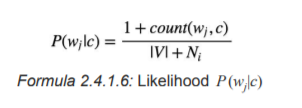
The prior distribution P(c) can be used to incorporate additional assumptions about the relative frequencies of classes. It is computed by:

**P(c) = Ni / N**

**Formula: Prior Distribution P(c)**

where N is the total number of training reviews and N is the number of training reviews in class i c.

The likelihood P(w |c) is usually computed using the formula:



where count(w, c) is the number of times that word occurs within the training reviews of class j wj c, and |V | = ∑ the size of the vocabulary. This estimation uses the simplest smoothing j wj method to solve the zero probability problem that arises when our model encounters a word seen in the test set but not in the training set, Laplaceor add­one since we use 1 as constant. We will see that Laplace smoothing method is not effective compared to other smoothing methods used in language models.

**3.4.1 NAIVE BAYES CLASSIFIER APPLICATION USING EXAMPLE**

**FORMULA:**

**P ( A | B ) = P( B | A ) \* P( A ) / P( B )**

**DATASET:**

|  |
| --- |
| **X = { x1, x2, x3 …….xn } { y }** |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **f1** | **f2** | **f3** | **…..** | **fn** | **y** | | **x1** | **x2** | **x3** | **…...** | **xn** | **y1** | |

**DERIVATION:**

|  |
| --- |
| **P ( y | x1, x2, x3…..xn ) = P( x1 | y )\* P( x1 | y )\* P( x1 | y ) ……..P( xn | y ) \* P(y)**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **P(x1)\* P(x2)\* P(x3)......P(xn) -> Constant for all**  **P ( y | x1, x2, x3…..xn ) = P(y) ∑ P(xi | y)**  **P ( y | x1, x2, x3…..xn ) = argmax ( P(y) ∑ P(xi | y) ) -> main formula for classifier** |

**FEATURES:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **the** | **food** | **delicious** | **bad** | **output** |
| **1** | **1** | **1** | **0** | **1(yes)** |
| **1** | **1** | **0** | **1** | **0(no)** |
| **0** | **1** | **0** | **1** | **0(no)** |
|  | | | | |
| **Sentence1** | **The Food is delicious** | | | |
| **Sentence2** | **The food is Bad** | | | |
| **Sentence3** | **Food is Bad** | | | |
|  | | | | |
| **To Classify**  **Sentence - Good(yes) or Bad(no)** | | | | |
| **For yes, p(y=yes) = 1/3, p(y=no) = 2/3**  **Sentence - delicious food** | | | | |
| **p(y=yes) \* p(x1|y=yes) \* p(x2|y=yes)**  **⅓ \* 1 \* ⅓** | | | | |
| **For no** | | | | |
| **p(y=yes) \* p(x1|y=yes) \* p(x2|y=yes)**  **⅔ \* 0 \* ⅔** | | | | |
| **The higher the value of the probability of yes or no, the output would be selected from there.** | | | | |

**3.5 CODING**

**home.html**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Hotel Review Sentiment Analysis</title>**

**<link rel="icon" href="../static/hotel.png">**

**<!-- <link rel="stylesheet" type="text/css" href="../static/css/styles.css"> -->**

**<link rel="stylesheet" href="../static/w3.css">**

**<link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='css/styles.css') }}">**

**</head>**

**<body>**

**<div class="w3-center w3-padding-16 w3-red w3-text-white w3-xxlarge">**

**Hotel Review Sentiment Analysis**

**</div>**

**<div class="ml-container container w3-center w3-margin-top">**

**<form action="{{ url\_for('predict')}}" method="POST">**

**<textarea name="message" style="outline: none;" placeholder="Enter your review here" rows="6" cols="50" required></textarea>**

**<br/>**

**<input type="submit" class="btn-info w3-button w3-red w3-xlarge w3-round-large w3-hover-brown" value="Predict">**

**</form>**

**</div>**

**</body>**

**</html>**

**result.html**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Hotel Review Sentiment Analysis</title>**

**<link rel="icon" href="../static/hotel.png">**

**<link rel="stylesheet" type="text/css" href="../static/styles.css">**

**<link rel="stylesheet" href="../static/w3.css">**

**</head>**

**<body>**

**<div class="w3-center w3-padding-16 w3-red w3-xxlarge">**

**Hotel Review Sentiment Analysis**

**</div>**

**<div class="results w3-center w3-bold w3-panel">**

**<h3 class=" w3-padding-16 w3-blue">"{{message}}"</h3>**

**{% if my\_nb\_prediction == 1%}**

**<div class="w3-green padding-large w3-border w3-round-large">**

**<h2>Naive Bayes classifier</h2>**

**<h2><b>POSITIVE REVIEW <br> with a predicted probability of {{('%.2f'%(nb\_percentage\*100))}}%</b></h2>**

**</div>**

**{% elif my\_nb\_prediction == 0%}**

**<div class="w3-red padding-large w3-border w3-round-large">**

**<h2>Naive Bayes classifier</h2>**

**<h2><b>NEGATIVE REVIEW <br>with a predicted probability of {{('%.2f'%(nb\_percentage\*100))}}%</b></h2>**

**</div>**

**{% endif %}**

**</div>**

**</body>**

**</html>**

**NaiveBayes.ipynb**

|  |
| --- |
| **# import pandas and pickle import pandas as pd import pickle # read the csv review dataset trip = pd.read\_csv('../dataset/tripadvisor\_hotel\_reviews.csv') # Let's create a new data frame   trip = trip[(trip['Rating']==5)|(trip['Rating']==2)|(trip['Rating']==1)][['Review','Rating']]  # Lets modify the Rating column trip['Rating'] = trip['Rating'].apply(lambda rating: 'Pos' if rating==5 else 'Neg') # reseting the index because after removing some rows, the index gets crowded trip.reset\_index(inplace=True) trip.head()**    **trip['Rating'].value\_counts()**    **#Data cleaning and preprocessing import re import nltk from nltk.corpus import stopwords from nltk.stem.porter import PorterStemmer from nltk.stem import WordNetLemmatizer # Lemmatization object ps = WordNetLemmatizer() corpus = [] # Text preprocessing # keep only text based # lower all the letters # split the words # for i in range(0,len(trip)): # review = re.sub('[^a-zA-Z]'," ",trip['Review'][i]) # review = review.lower() # review = review.split() # review = [ps.lemmatize(word) for word in review if not word in stopwords.words('english')] # review = ' '.join(review) # corpus.append(review) # the data preprocessing takes time # clean data clean\_data = pd.read\_csv('clean\_data.csv') corpus\_clean\_data = clean\_data.to\_numpy().flatten() corpus\_clean\_data\_usable = corpus\_clean\_data.tolist() from sklearn.feature\_extraction.text import TfidfVectorizer cv = TfidfVectorizer(max\_features=20000,ngram\_range=(2,2)) X = cv.fit\_transform(corpus\_clean\_data\_usable) temp = cv.fit(corpus\_clean\_data\_usable).get\_feature\_names() pd.DataFrame(temp).to\_csv('temp.csv') y = pd.get\_dummies(trip['Rating']) y = y.iloc[:,1].values y**  **-> array([0, 1, 1, ..., 0, 0, 0], dtype=uint8)**  **#train test split from sklearn.model\_selection import train\_test\_split X\_train, X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size=0.20,random\_state=3) #training model using Naive bayes classifier  from sklearn.naive\_bayes import MultinomialNB spam\_detect\_model = MultinomialNB().fit(X\_train,y\_train) # for the accuracy spam\_detect\_model.score(X\_test,y\_test)  y\_pred = spam\_detect\_model.predict(X\_test) #compare y test and y\_pred #confusion matrix is a 2x2 matrix and it tells, #how many number of elements are correctly predicted.  from sklearn.metrics import confusion\_matrix confusion\_matrix = confusion\_matrix(y\_test,y\_pred) confusion\_matrix**  **array([[ 571, 61],**  **[ 35, 1787]],**  **dtype=int64)**  **#checking accuracy score from sklearn.metrics import recall\_score recall\_score(y\_test,y\_pred)**  **-> 0.9807903402854007**  **from sklearn.metrics import precision\_score precision\_score(y\_test,y\_pred)**  **-> 0.966991341991342**  **from sklearn.metrics import fbeta\_score fbeta\_score(y\_test,y\_pred,beta=1)**  **-> 0.973841961852861**  **# Checking the training model with custom input data positive\_message = "dont care if it is good" negative\_message = "the hotel was bad and the staff was rude" data = [positive\_message] vect = cv.transform(data).toarray() my\_prediction = spam\_detect\_model.predict(vect) my\_prediction\_prob = spam\_detect\_model.predict\_proba(vect) if my\_prediction==1:  print("Positive") else:  print("Negative")**  **-> Negative** |

**BagOfWords.ipynb**

|  |
| --- |
| **from sklearn.feature\_extraction.text import CountVectorizer import pandas as pd example\_data = ['This is first document.','This is the second document.','And this is the third'] example\_vec = CountVectorizer() example\_vec\_fit = example\_vec.fit(example\_data).get\_feature\_names() # my list of features from the data print(example\_vec\_fit)**  **-> ['and', 'document', 'first', 'is', 'second', 'the', 'third', 'this']**  **example\_vec\_transform = example\_vec.transform(example\_data).toarray() print(example\_vec\_transform)**  **-> [[0 1 1 1 0 0 0 1]**  **[0 1 0 1 1 1 0 1]**  **[1 0 0 1 0 1 1 1]]**  **# examine the vocabulary and document-term matrix together pd.DataFrame(example\_vec\_transform, columns=example\_vec\_fit)** |

**Style.css**

**body{**

**font:15px/1.5 Arial, Helvetica,sans-serif;**

**padding: 0px;**

**background-color:#f4f3f3;**

**}**

**.container{**

**width:100%;**

**margin: auto;**

**overflow: hidden;**

**}**

**header{**

**background:#03A9F4;**

**border-bottom:#448AFF 3px solid;**

**height:120px;**

**width:100%;**

**padding-top:30px;**

**}**

**.main-header{**

**text-align:center;**

**background-color: blue;**

**height:100px;**

**width:100%;**

**margin:0px;**

**}**

**#brandname{**

**float:left;**

**font-size:30px;**

**color: #fff;**

**margin: 10px;**

**}**

**header h2{**

**text-align:center;**

**color:#fff;**

**}**

**.padding-large {**

**padding: 10px 5px;**

**}**

**.btn-info {background-color: #2196F3;**

**height:40px;**

**width:100px;} /\* Blue \*/**

**.btn-info:hover {background: #0b7dda;}**

**.results{**

**padding: 20px;**

**width: 70%;**

**margin-inline: auto;**

**}**

**requirements.txt**

**Flask**

**Werkzeug**

**gunicorn**

**itsdangerous**

**Jinja2**

**MarkupSafe**

**scikit-learn**

**numpy**

**pandas**

**style.css**

|  |
| --- |
| **body{  font:15px/1.5 Arial, Helvetica,sans-serif;  padding: 0px;  background-color:#f4f3f3; }  .container{  width:100%;  margin: auto;  overflow: hidden; }  header{  background:#03A9F4;  border-bottom:#448AFF 3px solid;  height:120px;  width:100%;  padding-top:30px;  }  .main-header{  text-align:center;  background-color: blue;  height:100px;  width:100%;  margin:0px; } #brandname{  float:left;  font-size:30px;  color: #fff;  margin: 10px; }  header h2{  text-align:center;  color:#fff;  }  .padding-large {  padding: 10px 5px; }   .btn-info {background-color: #2196F3;  height:40px;  width:100px;} /\* Blue \*/ .btn-info:hover {background: #0b7dda;}   .results{  padding: 20px;   width: 70%;  margin-inline: auto; }** |

**w3.css**

**/\* W3.CSS 4.15 December 2020 by Jan Egil and Borge Refsnes \*/**

**html{box-sizing:border-box}\*,\*:before,\*:after{box-sizing:inherit}**

**/\* Extract from normalize.css by Nicolas Gallagher and Jonathan Neal git.io/normalize \*/**

**.w3-animate-fading{animation:fading 10s infinite}@keyframes fading{0%{opacity:0}50%{opacity:1}100%{opacity:0}}**

**.w3-animate-opacity{animation:opac 0.8s}@keyframes opac{from{opacity:0} to{opacity:1}}**

**.w3-animate-top{position:relative;animation:animatetop 0.4s}@keyframes animatetop{from{top:-300px;opacity:0} to{top:0;opacity:1}}**

**.w3-animate-left{position:relative;animation:animateleft 0.4s}@keyframes animateleft{from{left:-300px;opacity:0} to{left:0;opacity:1}}**

**.w3-animate-right{position:relative;animation:animateright 0.4s}@keyframes animateright{from{right:-300px;opacity:0} to{right:0;opacity:1}}**

**.w3-animate-bottom{position:relative;animation:animatebottom 0.4s}@keyframes animatebottom{from{bottom:-300px;opacity:0} to{bottom:0;opacity:1}}**

**.w3-animate-zoom {animation:animatezoom 0.6s}@keyframes animatezoom{from{transform:scale(0)} to{transform:scale(1)}}**

**.w3-animate-input{transition:width 0.4s ease-in-out}.w3-animate-input:focus{width:100%!important}**

**.w3-opacity,.w3-hover-opacity:hover{opacity:0.60}.w3-opacity-off,.w3-hover-opacity-off:hover{opacity:1}**

**.w3-opacity-max{opacity:0.25}.w3-opacity-min{opacity:0.75}**

**.w3-greyscale-max,.w3-grayscale-max,.w3-hover-greyscale:hover,.w3-hover-grayscale:hover{filter:grayscale(100%)}**

**.w3-greyscale,.w3-grayscale{filter:grayscale(75%)}.w3-greyscale-min,.w3-grayscale-min{filter:grayscale(50%)}**

**.w3-sepia{filter:sepia(75%)}.w3-sepia-max,.w3-hover-sepia:hover{filter:sepia(100%)}.w3-sepia-min{filter:sepia(50%)}**

**.w3-tiny{font-size:10px!important}.w3-small{font-size:12px!important}.w3-medium{font-size:15px!important}.w3-large{font-size:18px!important}**

**.w3-xlarge{font-size:24px!important}.w3-xxlarge{font-size:36px!important}.w3-xxxlarge{font-size:48px!important}.w3-jumbo{font-size:64px!important}**

**.w3-left-align{text-align:left!important}.w3-right-align{text-align:right!important}.w3-justify{text-align:justify!important}.w3-center{text-align:center!important}**

**.w3-border-0{border:0!important}.w3-border{border:1px solid #ccc!important}**

**.w3-border-top{border-top:1px solid #ccc!important}.w3-border-bottom{border-bottom:1px solid #ccc!important}**

**.w3-border-left{border-left:1px solid #ccc!important}.w3-border-right{border-right:1px solid #ccc!important}**

**.w3-topbar{border-top:6px solid #ccc!important}.w3-bottombar{border-bottom:6px solid #ccc!important}**

**.w3-leftbar{border-left:6px solid #ccc!important}.w3-rightbar{border-right:6px solid #ccc!important}**

**.w3-section,.w3-code{margin-top:16px!important;margin-bottom:16px!important}**

**.w3-margin{margin:16px!important}.w3-margin-top{margin-top:16px!important}.w3-margin-bottom{margin-bottom:16px!important}**

**.w3-margin-left{margin-left:16px!important}.w3-margin-right{margin-right:16px!important}**

**.w3-padding-small{padding:4px 8px!important}.w3-padding{padding:8px 16px!important}.w3-padding-large{padding:12px 24px!important}**

**.w3-padding-16{padding-top:16px!important;padding-bottom:16px!important}.w3-padding-24{padding-top:24px!important;padding-bottom:24px!important}**

**.w3-padding-32{padding-top:32px!important;padding-bottom:32px!important}.w3-padding-48{padding-top:48px!important;padding-bottom:48px!important}**

**.w3-padding-64{padding-top:64px!important;padding-bottom:64px!important}**

**.w3-padding-top-64{padding-top:64px!important}.w3-padding-top-48{padding-top:48px!important}**

**.w3-padding-top-32{padding-top:32px!important}.w3-padding-top-24{padding-top:24px!important}**

**.w3-left{float:left!important}.w3-right{float:right!important}**

**.w3-button:hover{color:#000!important;background-color:#ccc!important}**

**.w3-transparent,.w3-hover-none:hover{background-color:transparent!important}**

**.w3-hover-none:hover{box-shadow:none!important}**

**/\* Colors \*/**

**.w3-amber,.w3-hover-amber:hover{color:#000!important;background-color:#ffc107!important}**

**.w3-aqua,.w3-hover-aqua:hover{color:#000!important;background-color:#00ffff!important}**

**.w3-blue,.w3-hover-blue:hover{color:#fff!important;background-color:#2196F3!important}**

**.w3-light-blue,.w3-hover-light-blue:hover{color:#000!important;background-color:#87CEEB!important}**

**.w3-brown,.w3-hover-brown:hover{color:#fff!important;background-color:#795548!important}**

**.w3-cyan,.w3-hover-cyan:hover{color:#000!important;background-color:#00bcd4!important}**

**.w3-blue-grey,.w3-hover-blue-grey:hover,.w3-blue-gray,.w3-hover-blue-gray:hover{color:#fff!important;background-color:#607d8b!important}**

**.w3-green,.w3-hover-green:hover{color:#fff!important;background-color:#4CAF50!important}**

**.w3-light-green,.w3-hover-light-green:hover{color:#000!important;background-color:#8bc34a!important}**

**.w3-indigo,.w3-hover-indigo:hover{color:#fff!important;background-color:#3f51b5!important}**

**.w3-khaki,.w3-hover-khaki:hover{color:#000!important;background-color:#f0e68c!important}**

**.w3-lime,.w3-hover-lime:hover{color:#000!important;background-color:#cddc39!important}**

**.w3-orange,.w3-hover-orange:hover{color:#000!important;background-color:#ff9800!important}**

**.w3-deep-orange,.w3-hover-deep-orange:hover{color:#fff!important;background-color:#ff5722!important}**

**.w3-pink,.w3-hover-pink:hover{color:#fff!important;background-color:#e91e63!important}**

**.w3-purple,.w3-hover-purple:hover{color:#fff!important;background-color:#9c27b0!important}**

**.w3-deep-purple,.w3-hover-deep-purple:hover{color:#fff!important;background-color:#673ab7!important}**

**.w3-red,.w3-hover-red:hover{color:#fff!important;background-color:#f44336!important}**

**.w3-sand,.w3-hover-sand:hover{color:#000!important;background-color:#fdf5e6!important}**

**.w3-teal,.w3-hover-teal:hover{color:#fff!important;background-color:#009688!important}**

**.w3-yellow,.w3-hover-yellow:hover{color:#000!important;background-color:#ffeb3b!important}**

**.w3-white,.w3-hover-white:hover{color:#000!important;background-color:#fff!important}**

**.w3-black,.w3-hover-black:hover{color:#fff!important;background-color:#000!important}**

**.w3-grey,.w3-hover-grey:hover,.w3-gray,.w3-hover-gray:hover{color:#000!important;background-color:#9e9e9e!important}**

**.w3-light-grey,.w3-hover-light-grey:hover,.w3-light-gray,.w3-hover-light-gray:hover{color:#000!important;background-color:#f1f1f1!important}**

**.w3-dark-grey,.w3-hover-dark-grey:hover,.w3-dark-gray,.w3-hover-dark-gray:hover{color:#fff!important;background-color:#616161!important}**

**.w3-pale-red,.w3-hover-pale-red:hover{color:#000!important;background-color:#ffdddd!important}**

**.w3-pale-green,.w3-hover-pale-green:hover{color:#000!important;background-color:#ddffdd!important}**

**.w3-pale-yellow,.w3-hover-pale-yellow:hover{color:#000!important;background-color:#ffffcc!important}**

**.w3-pale-blue,.w3-hover-pale-blue:hover{color:#000!important;background-color:#ddffff!important}**

**.w3-text-amber,.w3-hover-text-amber:hover{color:#ffc107!important}**

**.w3-text-aqua,.w3-hover-text-aqua:hover{color:#00ffff!important}**

**.w3-text-blue,.w3-hover-text-blue:hover{color:#2196F3!important}**

**.w3-text-light-blue,.w3-hover-text-light-blue:hover{color:#87CEEB!important}**

**.w3-text-brown,.w3-hover-text-brown:hover{color:#795548!important}**

**.w3-text-cyan,.w3-hover-text-cyan:hover{color:#00bcd4!important}**

**.w3-text-blue-grey,.w3-hover-text-blue-grey:hover,.w3-text-blue-gray,.w3-hover-text-blue-gray:hover{color:#607d8b!important}**

**.w3-text-green,.w3-hover-text-green:hover{color:#4CAF50!important}**

**.w3-text-light-green,.w3-hover-text-light-green:hover{color:#8bc34a!important}**

**.w3-text-indigo,.w3-hover-text-indigo:hover{color:#3f51b5!important}**

**.w3-text-khaki,.w3-hover-text-khaki:hover{color:#b4aa50!important}**

**.w3-text-lime,.w3-hover-text-lime:hover{color:#cddc39!important}**

**.w3-text-orange,.w3-hover-text-orange:hover{color:#ff9800!important}**

**.w3-text-deep-orange,.w3-hover-text-deep-orange:hover{color:#ff5722!important}**

**.w3-text-pink,.w3-hover-text-pink:hover{color:#e91e63!important}**

**.w3-text-purple,.w3-hover-text-purple:hover{color:#9c27b0!important}**

**.w3-text-deep-purple,.w3-hover-text-deep-purple:hover{color:#673ab7!important}**

**.w3-text-red,.w3-hover-text-red:hover{color:#f44336!important}**

**.w3-text-sand,.w3-hover-text-sand:hover{color:#fdf5e6!important}**

**.w3-text-teal,.w3-hover-text-teal:hover{color:#009688!important}**

**.w3-pink,.w3-hover-pink:hover{color:#fff!important;background-color:#e91e63!important}**

**.w3-purple,.w3-hover-purple:hover{color:#fff!important;background-color:#9c27b0!important}**

**.w3-deep-purple,.w3-hover-deep-purple:hover{color:#fff!important;background-color:#673ab7!important}**

**.w3-red,.w3-hover-red:hover{color:#fff!important;background-color:#f44336!important}**

**.w3-sand,.w3-hover-sand:hover{color:#000!important;background-color:#fdf5e6!important}**

**.w3-teal,.w3-hover-teal:hover{color:#fff!important;background-color:#009688!important}**

**.w3-yellow,.w3-hover-yellow:hover{color:#000!important;background-color:#ffeb3b!important}**

**.w3-white,.w3-hover-white:hover{color:#000!important;background-color:#fff!important}**

**.w3-black,.w3-hover-black:hover{color:#fff!important;background-color:#000!important}**

**.w3-grey,.w3-hover-grey:hover,.w3-gray,.w3-hover-gray:hover{color:#000!important;background-color:#9e9e9e!important}**

**.w3-light-grey,.w3-hover-light-grey:hover,.w3-light-gray,.w3-hover-light-gray:hover{color:#000!important;background-color:#f1f1f1!important}**

**.w3-dark-grey,.w3-hover-dark-grey:hover,.w3-dark-gray,.w3-hover-dark-gray:hover{color:#fff!important;background-color:#616161!important}**

**.w3-pale-red,.w3-hover-pale-red:hover{color:#000!important;background-color:#ffdddd!important}**

**.w3-pale-green,.w3-hover-pale-green:hover{color:#000!important;background-color:#ddffdd!important}**

**.w3-pale-yellow,.w3-hover-pale-yellow:hover{color:#000!important;background-color:#ffffcc!important}**

**.w3-pale-blue,.w3-hover-pale-blue:hover{color:#000!important;background-color:#ddffff!important}**

**.w3-text-amber,.w3-hover-text-amber:hover{color:#ffc107!important}**

**.w3-text-aqua,.w3-hover-text-aqua:hover{color:#00ffff!important}**

**.w3-text-blue,.w3-hover-text-blue:hover{color:#2196F3!important}**

**.w3-text-light-blue,.w3-hover-text-light-blue:hover{color:#87CEEB!important}**

**.w3-text-brown,.w3-hover-text-brown:hover{color:#795548!important}**

**.w3-text-cyan,.w3-hover-text-cyan:hover{color:#00bcd4!important}**

**.w3-text-blue-grey,.w3-hover-text-blue-grey:hover,.w3-text-blue-gray,.w3-hover-text-blue-gray:hover{color:#607d8b!important}**

**.w3-text-green,.w3-hover-text-green:hover{color:#4CAF50!important}**

**.w3-text-light-green,.w3-hover-text-light-green:hover{color:#8bc34a!important}**

**.w3-text-indigo,.w3-hover-text-indigo:hover{color:#3f51b5!important}**

**.w3-text-khaki,.w3-hover-text-khaki:hover{color:#b4aa50!important}**

**.w3-text-lime,.w3-hover-text-lime:hover{color:#cddc39!important}**

**.w3-text-orange,.w3-hover-text-orange:hover{color:#ff9800!important}**

**.w3-text-deep-orange,.w3-hover-text-deep-orange:hover{color:#ff5722!important}**

**.w3-text-pink,.w3-hover-text-pink:hover{color:#e91e63!important}**

**.w3-text-purple,.w3-hover-text-purple:hover{color:#9c27b0!important}**

**.w3-text-deep-purple,.w3-hover-text-deep-purple:hover{color:#673ab7!important}**

**.w3-text-red,.w3-hover-text-red:hover{color:#f44336!important}**

**.w3-text-sand,.w3-hover-text-sand:hover{color:#fdf5e6!important}**

**.w3-text-teal,.w3-hover-text-teal:hover{color:#009688!important}**

**.w3-text-yellow,.w3-hover-text-yellow:hover{color:#d2be0e!important}**

**.w3-text-white,.w3-hover-text-white:hover{color:#fff!important}**

**.w3-text-black,.w3-hover-text-black:hover{color:#000!important}**

**.w3-text-grey,.w3-hover-text-grey:hover,.w3-text-gray,.w3-hover-text-gray:hover{color:#757575!important}**

**.w3-text-light-grey,.w3-hover-text-light-grey:hover,.w3-text-light-gray,.w3-hover-text-light-gray:hover{color:#f1f1f1!important}**

**.w3-text-dark-grey,.w3-hover-text-dark-grey:hover,.w3-text-dark-gray,.w3-hover-text-dark-gray:hover{color:#3a3a3a!important}**

**.w3-border-amber,.w3-hover-border-amber:hover{border-color:#ffc107!important}**

**.w3-border-aqua,.w3-hover-border-aqua:hover{border-color:#00ffff!important}**

**.w3-border-blue,.w3-hover-border-blue:hover{border-color:#2196F3!important}**

**.w3-border-light-blue,.w3-hover-border-light-blue:hover{border-color:#87CEEB!important}**

**.w3-border-brown,.w3-hover-border-brown:hover{border-color:#795548!important}**

**.w3-border-cyan,.w3-hover-border-cyan:hover{border-color:#00bcd4!important}**

**.w3-border-blue-grey,.w3-hover-border-blue-grey:hover,.w3-border-blue-gray,.w3-hover-border-blue-gray:hover{border-color:#607d8b!important}**

**.w3-border-green,.w3-hover-border-green:hover{border-color:#4CAF50!important}**

**.w3-border-light-green,.w3-hover-border-light-green:hover{border-color:#8bc34a!important}**

**.w3-border-indigo,.w3-hover-border-indigo:hover{border-color:#3f51b5!important}**

**.w3-border-khaki,.w3-hover-border-khaki:hover{border-color:#f0e68c!important}**

**.w3-border-lime,.w3-hover-border-lime:hover{border-color:#cddc39!important}**

**.w3-border-orange,.w3-hover-border-orange:hover{border-color:#ff9800!important}**

**.w3-border-deep-orange,.w3-hover-border-deep-orange:hover{border-color:#ff5722!important}**

**.w3-border-pink,.w3-hover-border-pink:hover{border-color:#e91e63!important}**

**.w3-border-purple,.w3-hover-border-purple:hover{border-color:#9c27b0!important}**

**.w3-border-deep-purple,.w3-hover-border-deep-purple:hover{border-color:#673ab7!important}**

**.w3-border-red,.w3-hover-border-red:hover{border-color:#f44336!important}**

**.w3-border-sand,.w3-hover-border-sand:hover{border-color:#fdf5e6!important}**

**.w3-border-teal,.w3-hover-border-teal:hover{border-color:#009688!important}**

**.w3-border-yellow,.w3-hover-border-yellow:hover{border-color:#ffeb3b!important}**

**.w3-border-white,.w3-hover-border-white:hover{border-color:#fff!important}**

**.w3-border-black,.w3-hover-border-black:hover{border-color:#000!important}**

**.w3-border-grey,.w3-hover-border-grey:hover,.w3-border-gray,.w3-hover-border-gray:hover{border-color:#9e9e9e!important}**

**.w3-border-light-grey,.w3-hover-border-light-grey:hover,.w3-border-light-gray,.w3-hover-border-light-gray:hover{border-color:#f1f1f1!important}**

**.w3-border-dark-grey,.w3-hover-border-dark-grey:hover,.w3-border-dark-gray,.w3-hover-border-dark-gray:hover{border-color:#616161!important}**

**.w3-border-pale-red,.w3-hover-border-pale-red:hover{border-color:#ffe7e7!important}.w3-border-pale-green,.w3-hover-border-pale-green:hover{border-color:#e7ffe7!important}**

**.w3-border-pale-yellow,.w3-hover-border-pale-yellow:hover{border-color:#ffffcc!important}.w3-border-pale-blue,.w3-hover-border-pale-blue:hover{border-color:#e7ffff!important}**

**3.6 MODEL DEPLOYMENT**

The Model is deployed using Flask Framework.

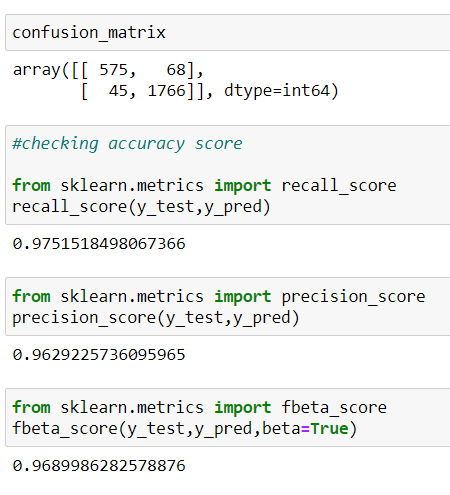
**POSITIVE REVIEW**

|  |
| --- |
|  |
|  |

**NEGATIVE REVIEWS**

|  |
| --- |
|  |
|  |

**CHAPTER 4**

**PERFORMANCE METRICS**

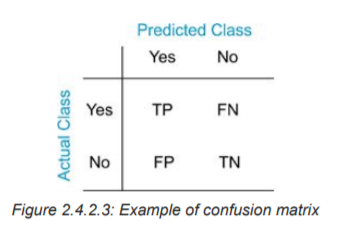
To evaluate our classifier, We use two methods: the Fbeta score and a confusion matrix. Fbeta-measure is a configurable single-score metric for evaluating a binary classification model based on the predictions made for the positive class. The Fbeta-measure is calculated using precision and recall.

Before we can dive into the Fbeta-measure, we must review the basics of the precision and recall metrics used to evaluate the predictions made by a classification model.

### Confusion Matrix

A [confusion matrix](https://machinelearningmastery.com/ufaqs/what-is-a-confusion-matrix/) summarizes the number of predictions made by a model for each class and the classes to which those predictions belong. It helps to understand the types of prediction errors made by a model. The simplest confusion matrix is for a two-class classification problem, with negative (class 0) and positive (class 1) classes.

In this type of confusion matrix, each cell in the table has a specific and well-understood name, summarized as follows:



The precision and recall metrics are defined in terms of the cells in the confusion matrix, specifically terms like true positives and false negatives.

**Precision** is a metric that quantifies the number of correct positive predictions made. It is calculated as the ratio of correctly predicted positive examples divided by the total number of positive examples that were predicted.

* Precision = TruePositives / (TruePositives + FalsePositives)

The result is a value between 0.0 for no precision and 1.0 for full or perfect precision.

**Recall** is a metric that quantifies the number of correct positive predictions made out of all positive predictions that could have been made. It is calculated as the ratio of correctly predicted positive examples divided by the total number of positive examples that could be predicted.

* Recall = TruePositives / (TruePositives + FalseNegatives)

The result is a value between 0.0 for no recall and 1.0 for full or perfect recall.

### F-BETA SCORE

The F-measure balances the precision and recall.

On some problems, we might be interested in an F-measure with more attention put on precision, such as when false positives are more important to minimize, but false negatives are still important.

On other problems, we might be interested in an F-measure with more attention put on recall, such as when false negatives are more important to minimize, but false positives are still important.

The solution is the Fbeta-measure.

The Fbeta-measure measure is an abstraction of the F-measure where the balance of precision and recall in the calculation of the [harmonic mean](https://en.wikipedia.org/wiki/Harmonic_mean) is controlled by a coefficient called *beta*.

* Fbeta = ((1 + beta^2) \* Precision \* Recall) / (beta^2 \* Precision + Recall)

The choice of the beta parameter will be used in the name of the Fbeta-measure.

For example, a beta value of 2 is referred to as F2-measure or F2-score. A beta value of 1 is referred to as the F1-measure or the F1-score.

Three common values for the beta parameter are as follows:

* **F0.5-Measure** (beta=0.5): More weight on precision, less weight on recall.
* **F1-Measure** (beta=1.0): Balance the weight on precision and recall.
* **F2-Measure** (beta=2.0): Less weight on precision, more weight on recall

The impact on the calculation for different beta values is not intuitive, at first.

### F1-Measure

The F-measure discussed in the previous section is an example of the Fbeta-measure with a *beta* value of 1.

Specifically, F-measure and F1-measure calculate the same thing; for example:

* F-Measure = ((1 + 1^2) \* Precision \* Recall) / (1^2 \* Precision + Recall)
* F-Measure = (2 \* Precision \* Recall) / (Precision + Recall)

Consider the case where we have 50 percept precision and perfect recall. We can manually calculate the F1-measure for this case as follows:

* F-Measure = (2 \* Precision \* Recall) / (Precision + Recall)
* F-Measure = (2 \* 0.5 \* 1.0) / (0.5 + 1.0)
* F-Measure = 1.0 / 1.5
* F-Measure = 0.666

We can confirm this calculation using the [fbeta\_score() function](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.fbeta_score.html) in scikit-learn with the “*beta*” argument set to 1.0.

### F0.5-Measure

The F0.5-measure is an example of the Fbeta-measure with a *beta* value of 0.5.

It has the effect of raising the importance of precision and lowering the importance of recall.

If maximizing precision minimizes false positives, and maximizing recall minimizes false negatives, then the **F0.5-measure puts more attention on minimizing false positives** than minimizing false negatives.

The F0.5-Measure is calculated as follows:

* F0.5-Measure = ((1 + 0.5^2) \* Precision \* Recall) / (0.5^2 \* Precision + Recall)
* F0.5-Measure = (1.25 \* Precision \* Recall) / (0.25 \* Precision + Recall)

Consider the case where we have 50 percent precision and perfect recall. We can manually calculate the F0.5-measure for this case as follows:

* F0.5-Measure = (1.25 \* Precision \* Recall) / (0.25 \* Precision + Recall)
* F0.5-Measure = (1.25 \* 0.5 \* 1.0) / (0.25 \* 0.5 + 1.0)
* F0.5-Measure = 0.625 / 1.125
* F0.5-Measure = 0.555

We would expect that a beta value of 0.5 would result in a lower score for this scenario given that precision has a poor score and the recall is excellent.

### F2-Measure

The F2-measure is an example of the F Beta-measure with a *beta* value of 2.0.

It has the effect of lowering the importance of precision and increase the importance of recall.

If maximizing precision minimizes false positives, and maximizing recall minimizes false negatives, then the **F2-measure puts more attention on minimizing false negatives** than minimizing false positives.

The F2-measure is calculated as follows:

* F2-Measure = ((1 + 2^2) \* Precision \* Recall) / (2^2 \* Precision + Recall)
* F2-Measure = (5 \* Precision \* Recall) / (4 \* Precision + Recall)

Consider the case where we have 50 percent precision and perfect recall.

We can manually calculate the F2-measure for this case as follows:

* F2-Measure = (5 \* Precision \* Recall) / (4 \* Precision + Recall)
* F2-Measure = (5 \* 0.5 \* 1.0) / (4 \* 0.5 + 1.0)
* F2-Measure = 2.5 / 3.0
* F2-Measure = 0.833

We would expect that a *beta* value of 2.0 would result in a higher score for this scenario given that recall has a perfect score, which will be promoted over that of the poor performance of precision.

**CONCLUSION AND FUTURE ENHANCEMENT**

Nowadays, sentiment analysis or opinion mining is a hot topic in machine learning. We are still far from detecting the sentiments of the corpus of texts very accurately because of the complexity in the English language and even more if we consider other languages such as Chinese.

In this project, we tried to show the basic way of classifying hotel reviews into a positive or negative category using two algorithms such as Naive Bayes and as a baseline, and how language models are related to the algorithm and can produce better results.

We could further improve our classifier by trying to extract more features from the reviews, trying different kinds of feature extraction methods, collecting more datasets, and tuning the parameters of the Naive Bayes classifier and or applying may be a more complicated deep learning model.

\

**REFERENCE**

* [**https://www.cse.ust.hk/~rossiter/independent\_studies\_projects/twitter\_emotion\_analysis/twitter\_emotion\_analysis.pdf**](https://www.cse.ust.hk/~rossiter/independent_studies_projects/twitter_emotion_analysis/twitter_emotion_analysis.pdf)
* [**http://www.ijstr.org/final-print/may2020/A-Literature-Survey-On-Sentiment-Analysis-Techniques-Involving-Social-Media-And-Online-Platforms.pdf**](http://www.ijstr.org/final-print/may2020/A-Literature-Survey-On-Sentiment-Analysis-Techniques-Involving-Social-Media-And-Online-Platforms.pdf)
* **Alexander Pak, Patrick Paroubek. 2010, Twitter as a Corpus for Sentiment Analysis and Opinion Mining.**
* **Alec Go, Richa Bhayani, Lei Huang. Twitter Sentiment Classification using Distant Supervision.**
* **Jin Bai, Jian­Yun Nie. Using Language Models for Text Classification.**
* **Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, Rebecca Passonneau. Sentiment Analysis of Twitter Data.**
* **Fuchun Peng. 2003, Augmenting Naive Bayes Classifiers with Statistical Language Models**