

**Machine Learning**

**Subject code**

**20CYS215-ML**

**ASSESSMENT REPORT**

**B. TECH (II YEAR)**

**(2023‐2024)**

**Table of contents**

|  |  |  |
| --- | --- | --- |
| S.No | Contents | Page no |
| 1 | Literature Review | 3 |
| 2 | Experimentation | 7 |
| 3 | Analysis | 25 |
| 4 | References | 27 |

**A Literature Review**

**Word Vectorization Techniques in Natural Language Processing:**

Natural language processing (NLP) deals with enabling computers to understand and manipulate human language. However, raw text data poses challenges due to its unstructured nature and ambiguity. Word vectorization techniques bridge this gap by converting words or phrases into numerical vectors, allowing machines to process and analyze textual information. This review explores key concepts in word vectorization, discusses various techniques, and highlights their significance in NLP tasks.

**Key Concepts**

* **Vector Space Model:** Words are represented as points in a high-dimensional space, where dimensions correspond to features extracted from the text corpus. The distance between vectors reflects semantic similarity between words.
* **Feature Extraction:** The process of transforming textual data into numerical features suitable for vectorization. Common features include word frequency, document frequency, and co-occurrence patterns.
* **Dimensionality Reduction:** Techniques like Principal Component Analysis (PCA) can be used to reduce the dimensionality of the vector space, improving computational efficiency without significant loss of information.

**Word Vectorization Techniques**

**Bag-of-Words (BoW):** A simple yet effective technique that represents a document as a presence/frequency count of words within a vocabulary. While computationally efficient, BoW ignores word order and semantic relationships.

**TF-IDF (Term Frequency-Inverse Document Frequency):** Addresses limitations of BoW by assigning weights to words based on their frequency within a document (TF) and rarity across the corpus (IDF). Words that are frequent in a specific document but rare overall receive higher weights, capturing semantic importance.

**Word Embeddings:** Techniques like Word2Vec and GloVe capture semantic relationships between words by analyzing their co-occurrence patterns within a corpus. These methods represent words as vectors in a high-dimensional space, where similar words have similar vector representations.

**Word2Vec:** Utilizes two main architectures: Skip-gram and CBOW. Skip-gram predicts surrounding words based on a given word, while CBOW predicts a central word based on its surrounding context.

**GloVe:** Analyzes word co-occurrence statistics from a large text corpus to learn word vector representations. It leverages the ratio of co-occurrence probabilities between words to capture semantic similarities.

**Significance of Word Vectorization in NLP Tasks**

Word vectorization serves as a foundation for various NLP tasks by enabling machines to process and understand the underlying meaning within textual data. Here are some key applications:

**Sentiment Analysis:** Classifying opinions and emotions expressed within text data. Word vectors can capture the sentiment of words and sentences, aiding in sentiment classification tasks.

**Machine Translation:** Automatically translating text from one language to another. Word vectors can represent semantic relationships between words across languages, facilitating accurate translation.

**Text Classification:** Categorizing text data into predefined classes. Word vectors allow machines to identify key topics and themes within text, enabling effective text classification.

**Information Retrieval:** Finding relevant documents or information based on a user query. Word vectors can be used to match the semantic meaning of a query with documents in a corpus, improving retrieval accuracy.

**Conclusion**

Word vectorization techniques have revolutionized NLP by enabling the representation of textual data in a format amenable to machine learning algorithms. These techniques play a crucial role in various NLP tasks, allowing machines to process and understand the nuances of human language. As research progresses, new vectorization methods are likely to emerge, further enhancing the capabilities of NLP systems.

**Conventional Word Vectorization Methods in NLP**

1. **Bag-of-Words (BoW):**

**Underlying Principle:** BoW represents a document as a simple presence/frequency count of words within a predefined vocabulary. Each document is transformed into a fixed-length vector where each dimension corresponds to a word in the vocabulary. The value at each dimension represents the number of times that word appears in the document (frequency) or simply a 1 (presence).

**Applications:**

* BoW is computationally efficient and easy to implement.
* It finds use in tasks like document similarity analysis, where the focus is on word presence rather than order or semantics.
* It can be a starting point for more advanced techniques like TF-IDF.

**Limitations:**

* BoW ignores the order of words and any semantic relationships between them.
* Documents with entirely different meanings can have the same BoW representation if they share a similar set of words.

1. **TF-IDF (Term Frequency-Inverse Document Frequency):**

**Underlying Principle:**

TF-IDF builds upon BoW by assigning weights to words based on their importance within a document and across the entire corpus. It considers two factors:

* **Term Frequency (TF):** Similar to BoW, TF captures how frequently a word appears within a specific document.
* **Inverse Document Frequency (IDF):** This measures how rare a word is across the corpus. Words that appear frequently in many documents have a lower IDF weight, while those specific to a few documents receive a higher weight.

**Applications:**

TF-IDF addresses the limitations of BoW by giving more weight to words that are distinctive and informative for a particular document. It's widely used in tasks like:

* + - **Information Retrieval:** Ranking documents relevant to a user query based on the TF-IDF weight of query terms within the documents.
    - **Text Classification:** Identifying key topics within documents by analyzing the weighted terms.

**Limitations:**

While TF-IDF captures some semantic information, it doesn't directly model word relationships. Additionally, it can be sensitive to very frequent or rare words that might not be semantically significant.

1. **N-grams:**

**Underlying Principle:**

N-grams capture sequential word patterns within a text. They consider sequences of n consecutive words (unigrams for single words, bigrams for pairs, trigrams for triplets, etc.). Each document is then represented as a frequency count of these n-gram sequences.

**Applications:**

N-grams capture local word order and context, which can be helpful for tasks like:

* **Language Modeling:** Predicting the next word in a sequence based on the preceding n-grams.
* **Machine Translation:** Preserving word order and idiomatic expressions during translation.
* **Sentiment Analysis:** Identifying sentiment-bearing phrases that might not be obvious with single words.

**Limitations:**

N-grams with higher n values (trigrams and above) can lead to data sparsity, where specific n-gram sequences rarely appear in the corpus. This can negatively impact the effectiveness of the model.

**EXPERIMENTATION**

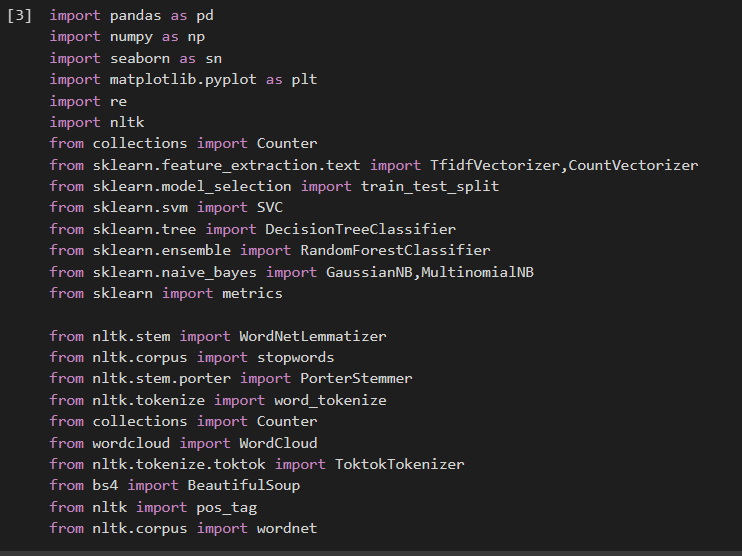
We have selected datasets created by Spam Text Message Classification of 5572 unique spam messages. This dataset consists of 2 columns, Category and Message. Message feature is classified in binary classification i.e. ham or spam.

• No. of data columns: 2

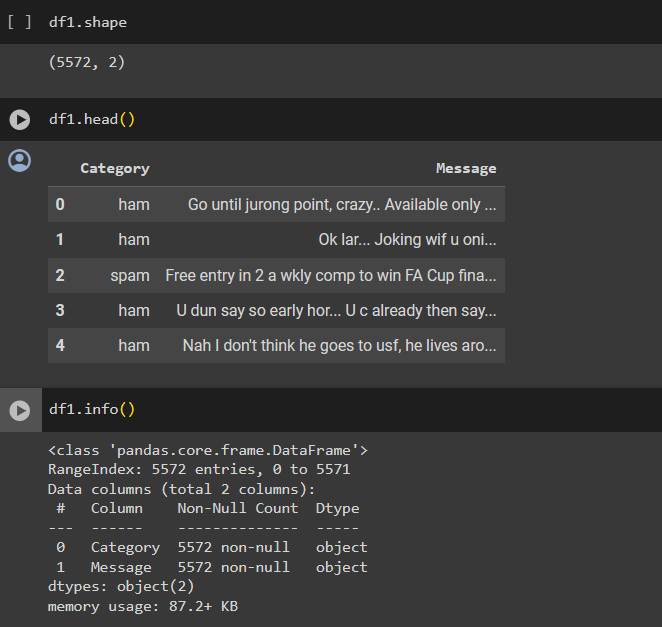
• No. of entries & range index: 5572 entries, 0 to 5571

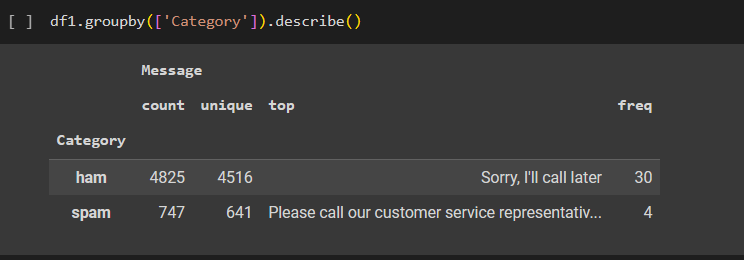
Link to dataset: [Spam Text Message Classification (kaggle.com)](https://www.kaggle.com/datasets/team-ai/spam-text-message-classification/data?select=SPAM+text+message+20170820+-+Data.csv)

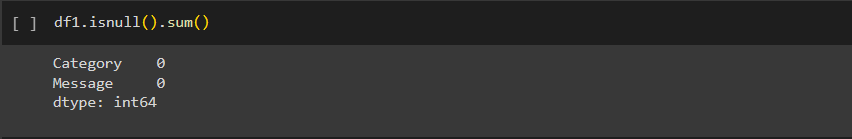
Importing the required libraries:



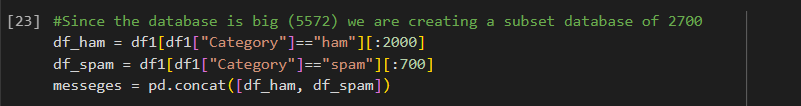
Exploring the dataset:

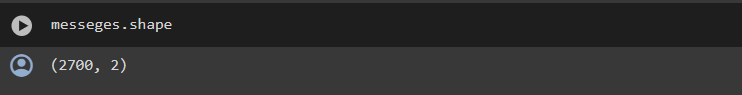




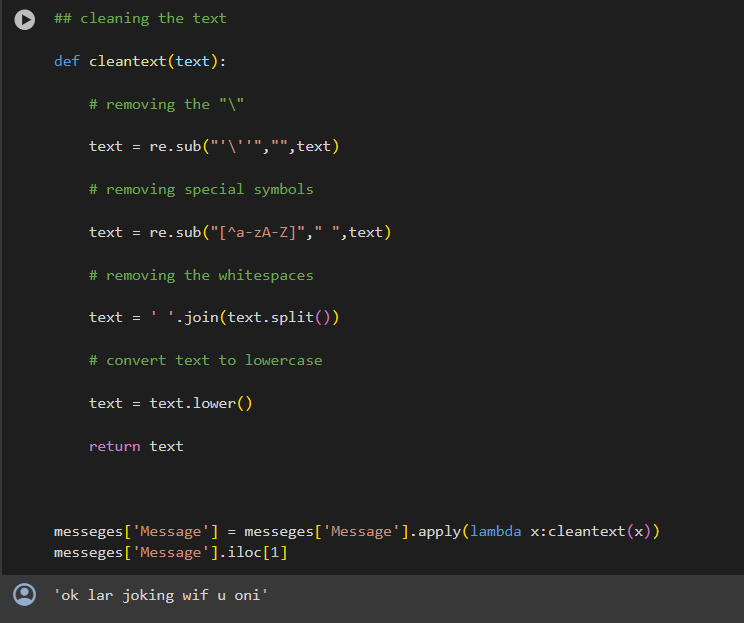


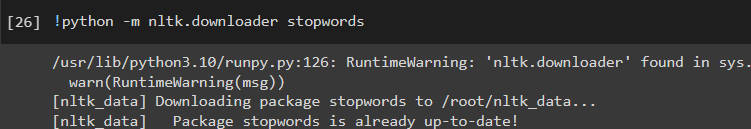
Since the dataset is big (5572) we are creating a subset database of 2700

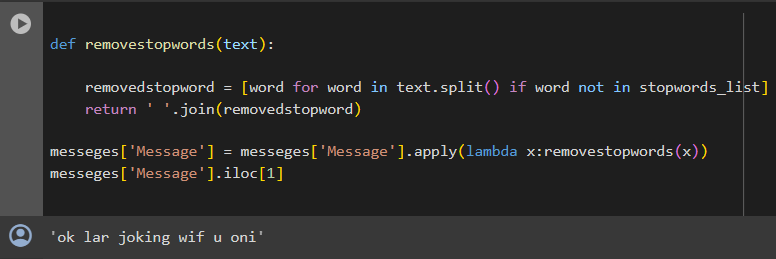


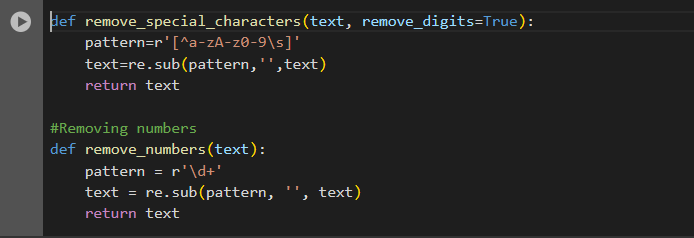


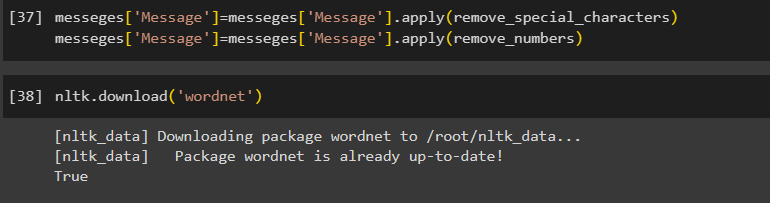
Cleaning the text

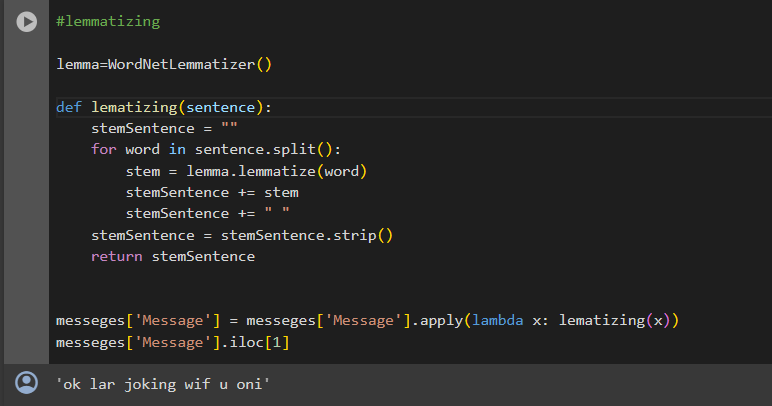


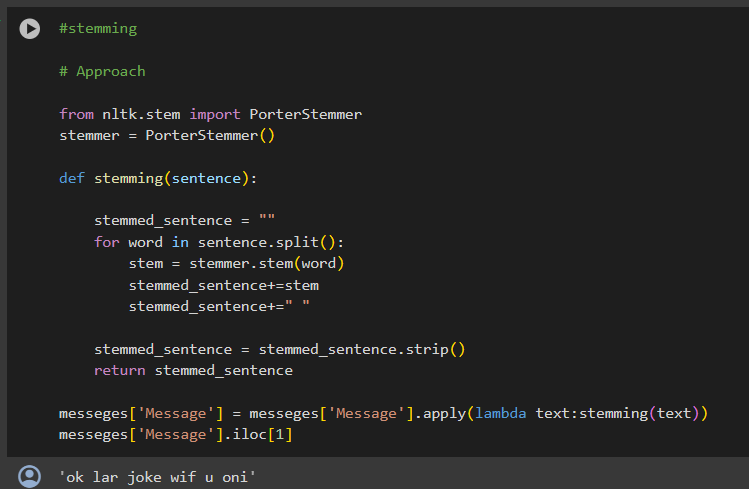


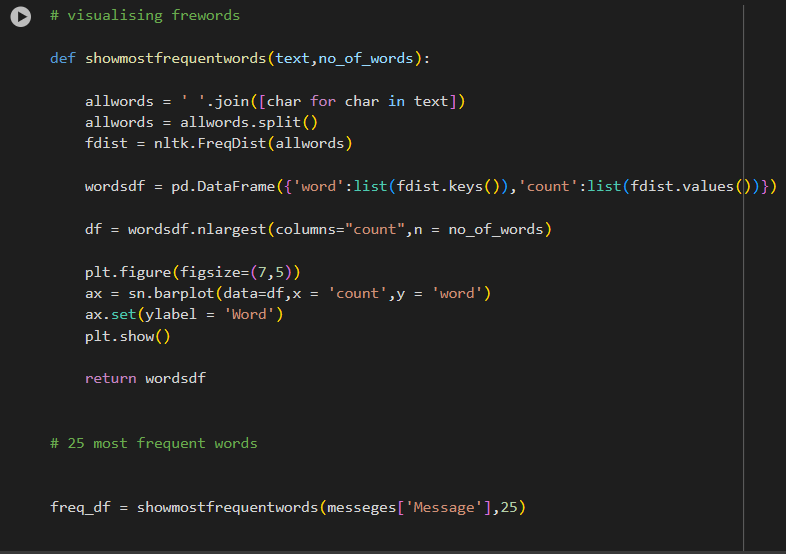


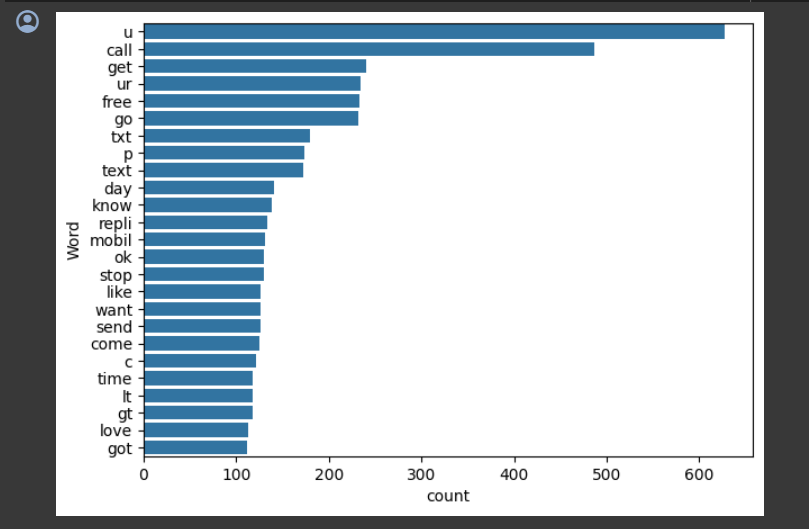








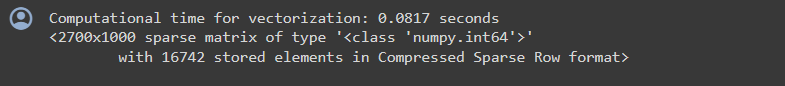


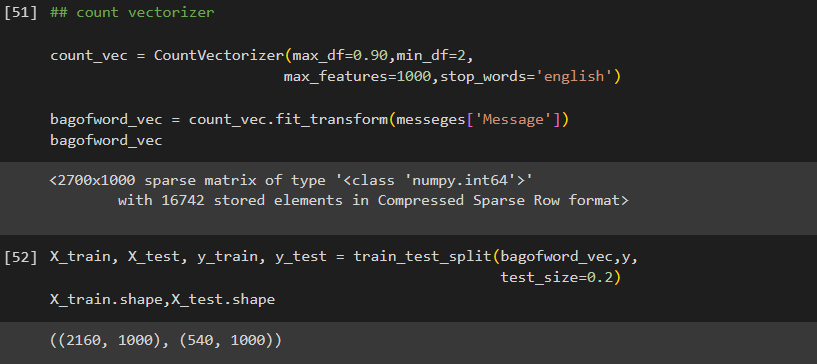




**VECTORIZATION**

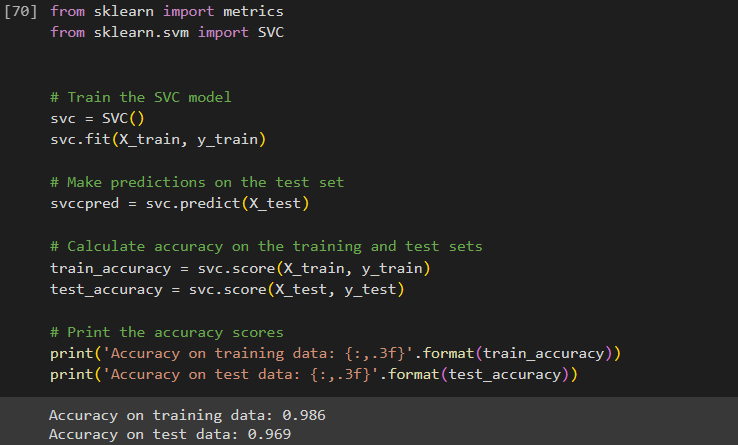
**bag of word**

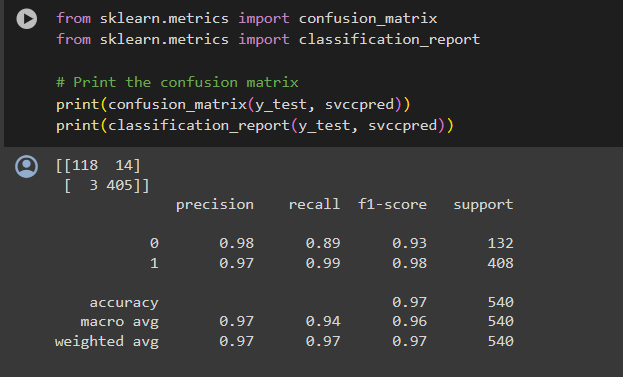




**Support Vector Machine Classification**

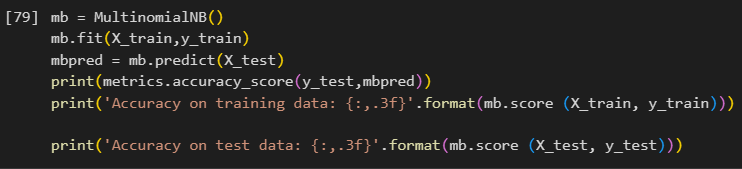
* A support vector machine (SVM) is a type of machine that employs methods to train and classify input within degrees of polarity, going beyond X/Y prediction.
* The SVM algorithm's purpose is to find the optimum line or decision boundary for categorising n-dimensional space into classes so that additional data points can be readily placed in the correct category in the future.
* A hyperplane is the name for the optimal choice boundary

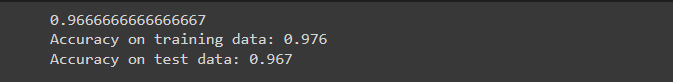
****

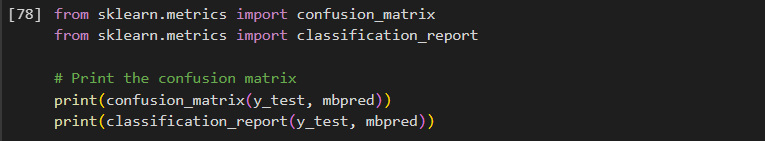
****

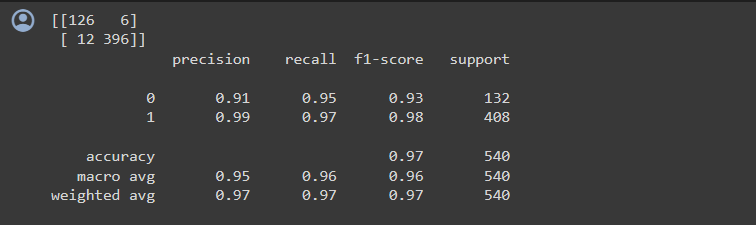
**MULTINOMIALNB**

* MNB uses Bayes' theorem to calculate the probability of a data point belonging to a class based on the presence and frequency of its features.
* MNB assumes features are independent (e.g., words in a document are unrelated), which simplifies calculations but might not always be true.
* MNB excels in tasks like spam filtering, sentiment analysis, and document classification due to its simplicity and efficiency in handling text data.



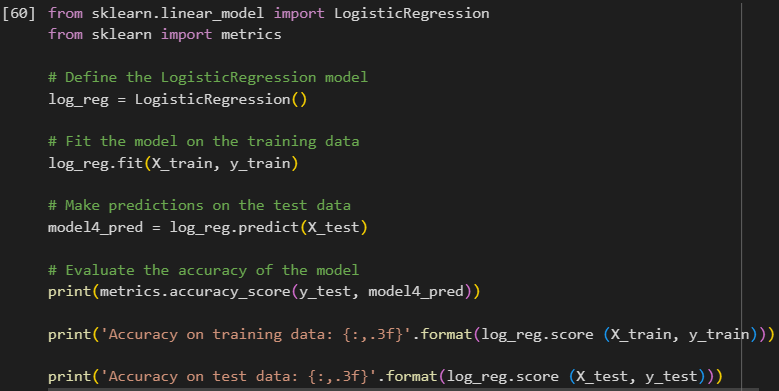


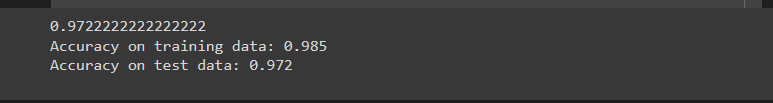


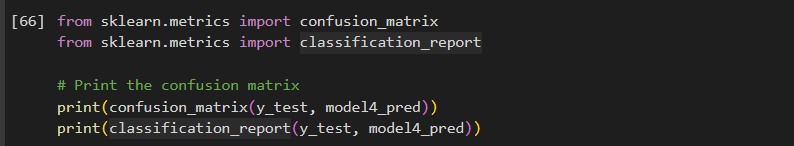


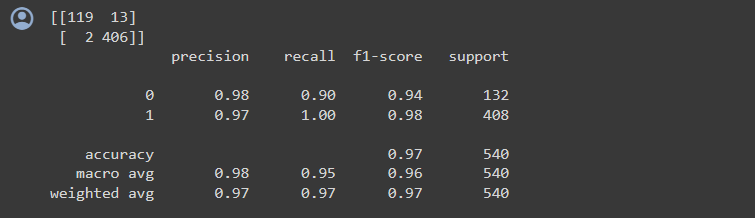
**Logistic Regression Classification**

* This algorithm is used to predict a binary outcome.
* The binary outcome is determined by analysing independent factors, with the findings falling into one of two groups.
* It is formulated as P(Y=1 | X) OR P(Y=0 | X) .
* This can be then used to calculate the probability of the variable as 0 or 1 or on a scale in between.



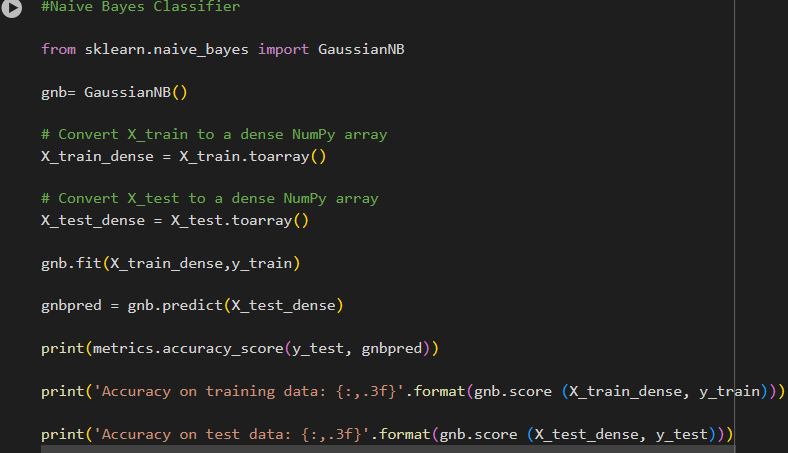


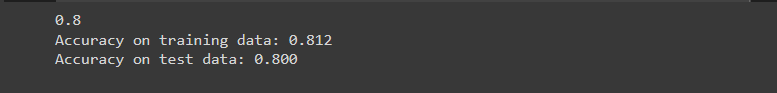


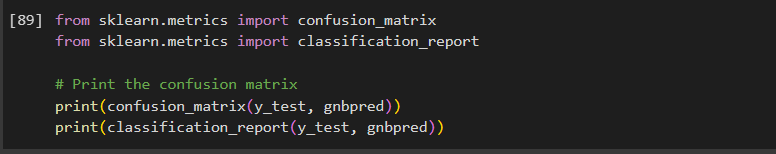


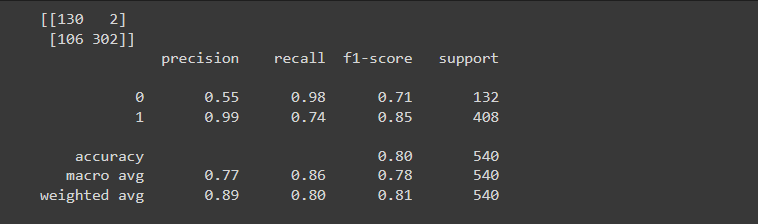
**Naive Bayes Classification**

* Every pair of features being classified is independent of each other, according to the Naive Bayes Classifier algorithm.
* The feature matrix and the response vector are the two elements of our dataset.
* It can be used in text analysis to classify words or phrases as belonging to a predefined "tag" (classification) Or Not









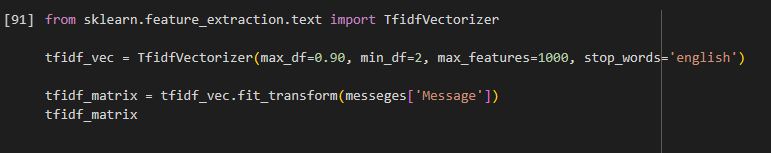
**Key Findings :**

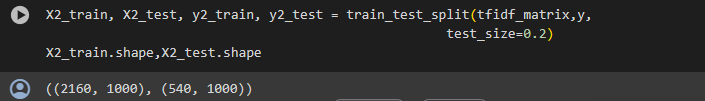
* Support Vector Machine Classifier accuracy is : 96.9%
* Naive Bayes Classifier accuracy is: 80.0 %
* Logistic Regression Classifier accuracy is : 97.2 %
* MultinomialNB classifier accuracy is: 96.7%

So Logistic Regression Classifier accuracy is giving the best accuracy on data with a value of 97.2 %

**TF-IDF (Term Frequency-Inverse Document Frequency)**

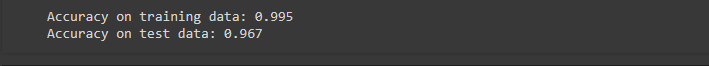
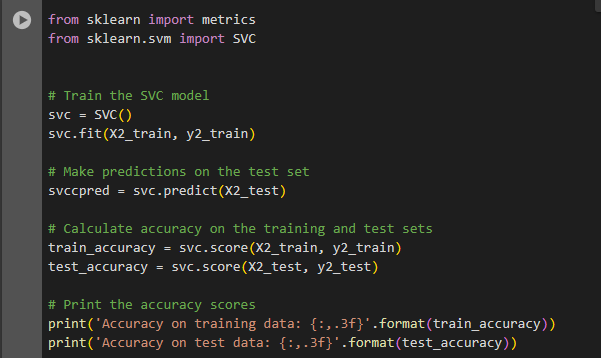


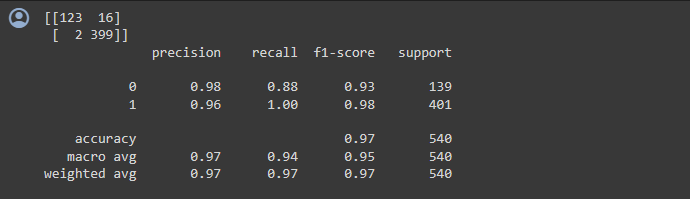
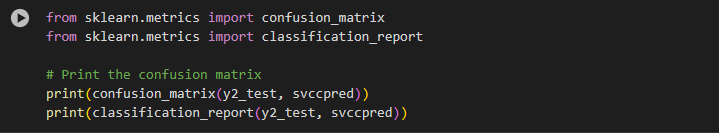




**Support Vector Machine Classification**

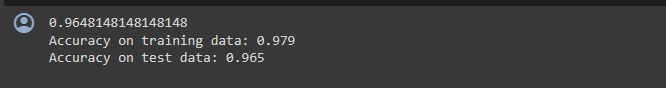
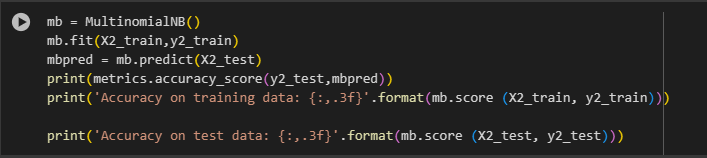
* A support vector machine (SVM) is a type of machine that employs methods to train and classify input within degrees of polarity, going beyond X/Y prediction.
* The SVM algorithm's purpose is to find the optimum line or decision boundary for categorising n-dimensional space into classes so that additional data points can be readily placed in the correct category in the future.
* A hyperplane is the name for the optimal choice boundary

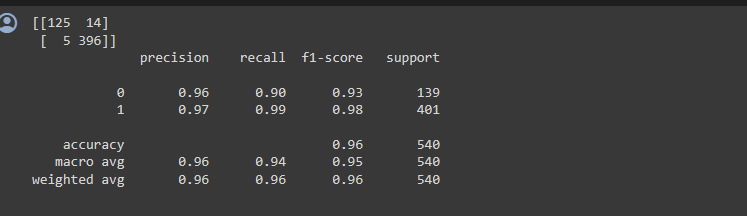
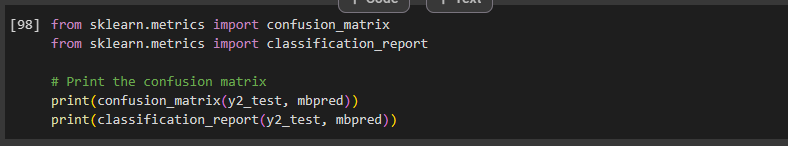
****

****

**MULTINOMIALNB**

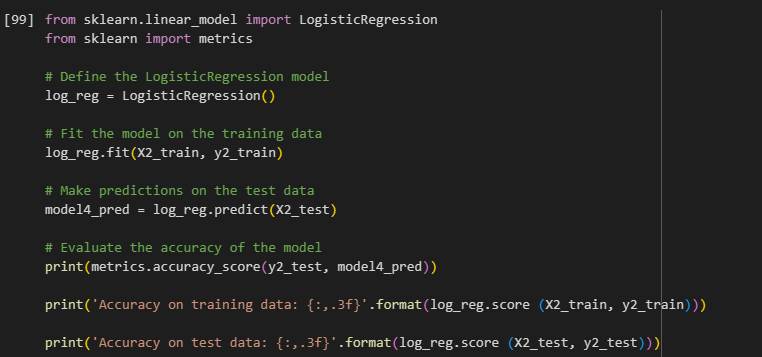
* MNB uses Bayes' theorem to calculate the probability of a data point belonging to a class based on the presence and frequency of its features.
* MNB assumes features are independent (e.g., words in a document are unrelated), which simplifies calculations but might not always be true.
* MNB excels in tasks like spam filtering, sentiment analysis, and document classification due to its simplicity and efficiency in handling text data.

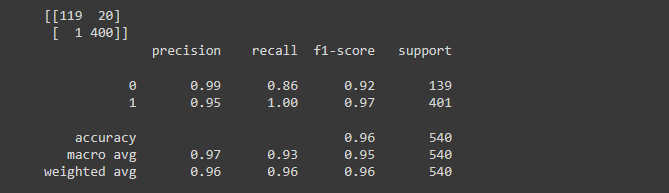
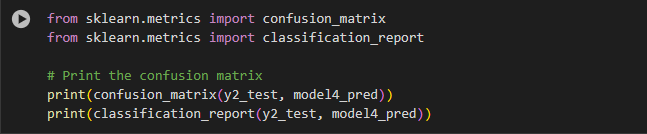
****

****

**Logistic Regression Classification**

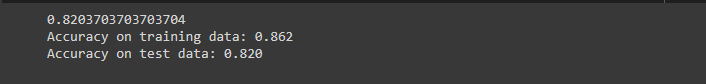
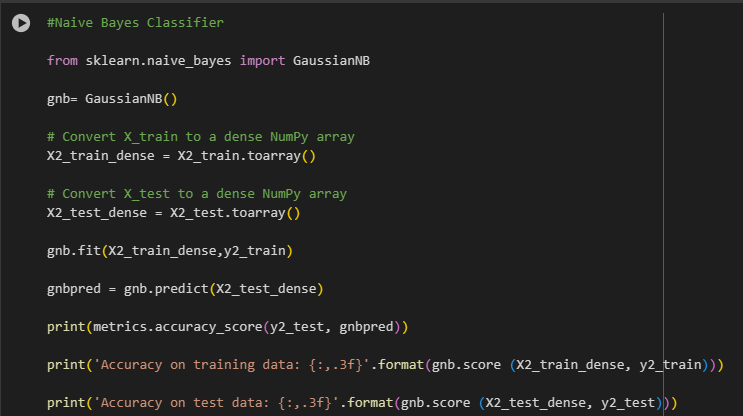
* This algorithm is used to predict a binary outcome.
* The binary outcome is determined by analysing independent factors, with the findings falling into one of two groups.
* It is formulated as P(Y=1 | X) OR P(Y=0 | X) .
* This can be then used to calculate the probability of the variable as 0 or 1 or on a scale in between.

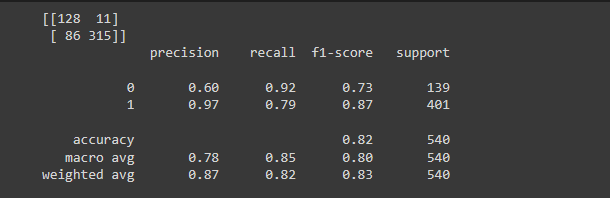
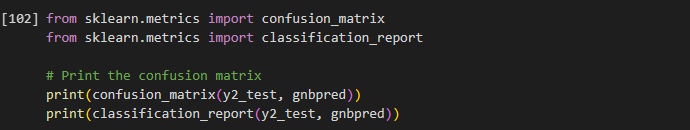
****

****

**Naive Bayes Classification**

* Every pair of features being classified is independent of each other, according to the Naive Bayes Classifier algorithm.
* The feature matrix and the response vector are the two elements of our dataset.
* It can be used in text analysis to classify words or phrases as belonging to a predefined "tag" (classification) Or Not.

****

****

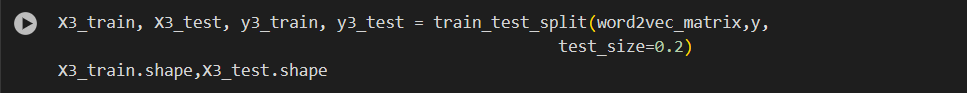
**Key Findings :**

* Support Vector Machine Classifier accuracy is : 96.6%
* Naive Bayes Classifier accuracy is: 82.0 %
* Logistic Regression Classifier accuracy is : 96.1 %
* MultinomialNB classifier accuracy is: 96.5%

So Support Vector Machine Classifier accuracy is giving the best accuracy on data with a value of 96.6 %

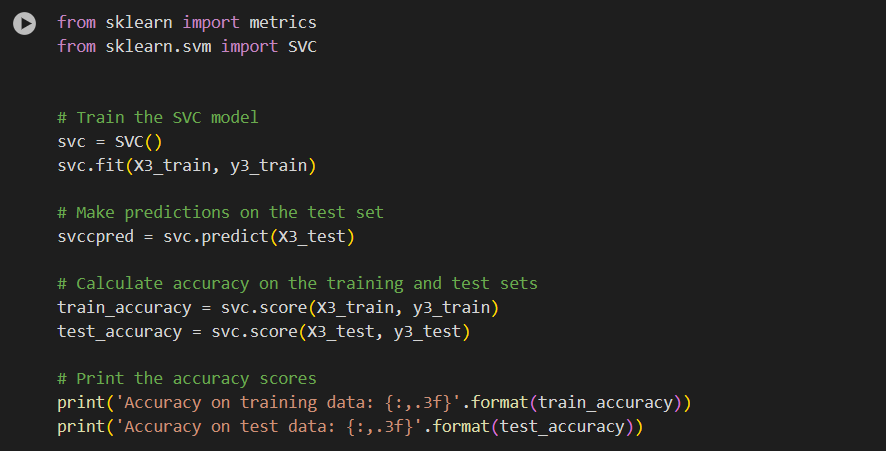
**Word2Vec**

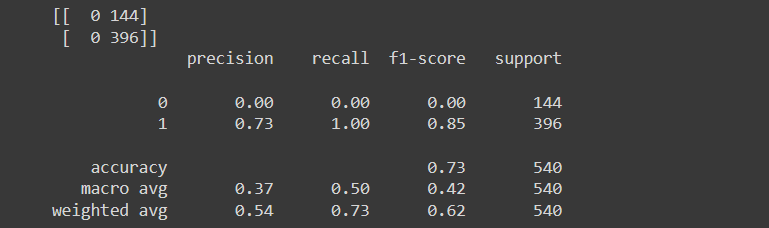
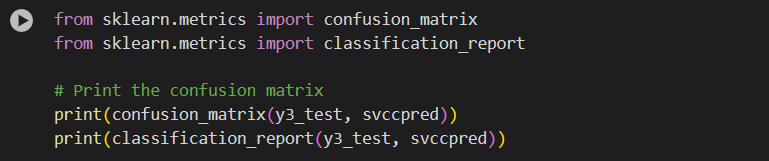
****

****

**Support Vector Machine Classification**

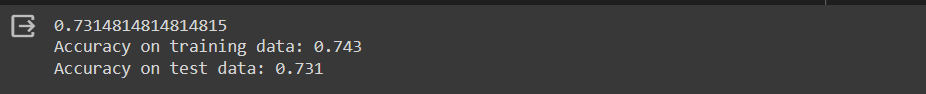
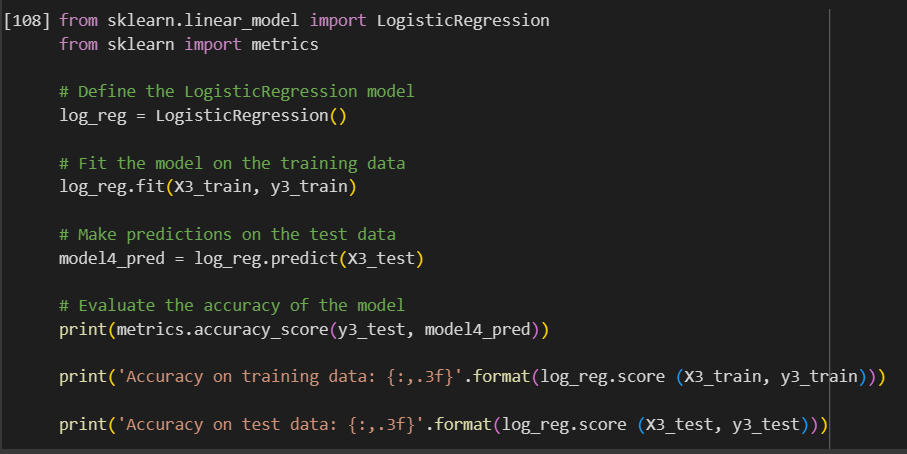
* A support vector machine (SVM) is a type of machine that employs methods to train and classify input within degrees of polarity, going beyond X/Y prediction.
* The SVM algorithm's purpose is to find the optimum line or decision boundary for categorising n-dimensional space into classes so that additional data points can be readily placed in the correct category in the future.
* A hyperplane is the name for the optimal choice boundary

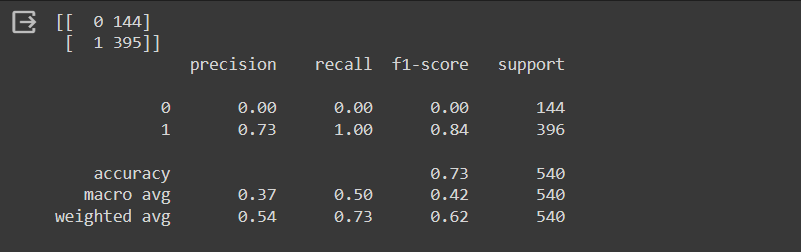
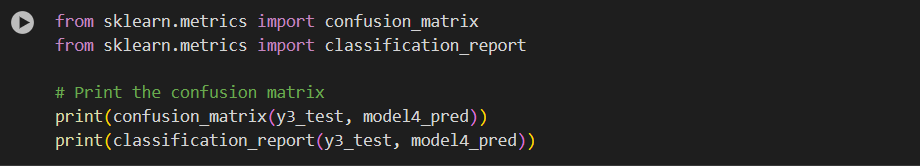
****

****

**Logistic Regression Classification**

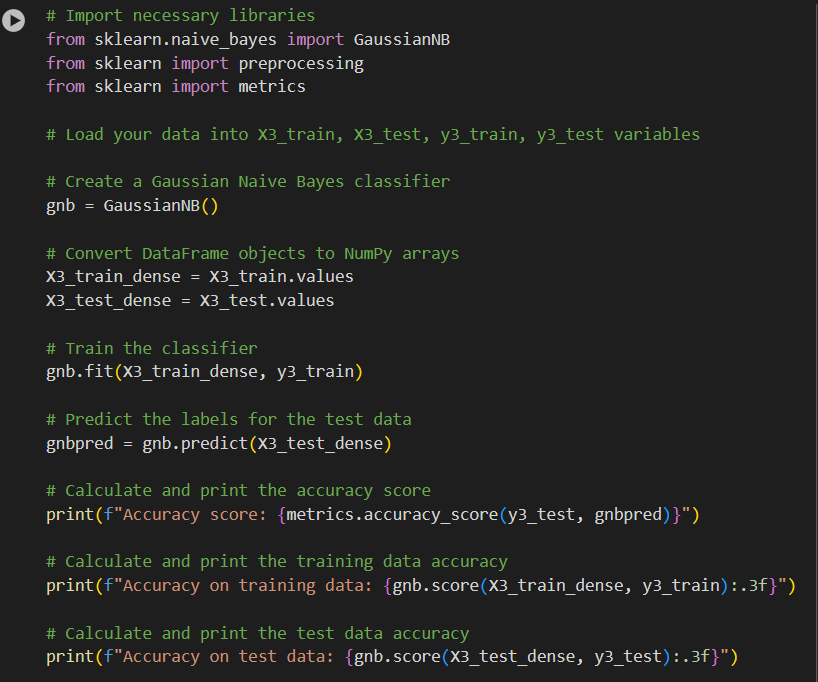
* This algorithm is used to predict a binary outcome.
* The binary outcome is determined by analysing independent factors, with the findings falling into one of two groups.
* It is formulated as P(Y=1 | X) OR P(Y=0 | X) .
* This can be then used to calculate the probability of the variable as 0 or 1 or on a scale in between.

****

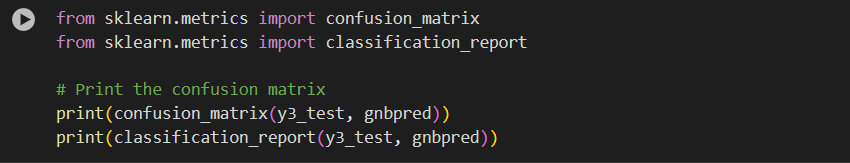
****

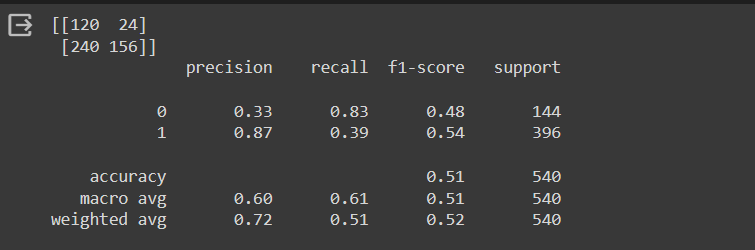
**Naive Bayes Classification**

* Every pair of features being classified is independent of each other, according to the Naive Bayes Classifier algorithm.
* The feature matrix and the response vector are the two elements of our dataset.
* It can be used in text analysis to classify words or phrases as belonging to a predefined "tag" (classification) Or Not.

****

****

****

****

**Key Findings :**

* Support Vector Machine Classifier accuracy is : 73.3%
* Naive Bayes Classifier accuracy is: 51.1 %
* Logistic Regression Classifier accuracy is : 73.1 %

So Support Vector Machine Classifier accuracy is giving the best accuracy on data with a value of 73.3 %

**Analysis**

**Bag of words:**

Logistic Regression Classifier accuracy is giving the best accuracy on data with a value of 97.2 %

**TF-IDF**:

Support Vector Machine Classifier accuracy is giving the best accuracy on data with a value of 96.6 %

**Word2Vec:**

Support Vector Machine Classifier accuracy is giving the best accuracy on data with a value of 73.3 %

|  |  |  |  |
| --- | --- | --- | --- |
| **Classifications/methods** | **Bag of Words** | **TF-IDF** | **Word2Vec** |
| **Accuracy** | 0.9722222222222222 | 0.967 | 0.733 |
| **Precession** | 0.98 | 0.98 | 0.73 |
| **Recall** | 0.90 | 0.88 | 1.00 |
| **F1 score** | 0.94 | 0.93 | 0.84 |

Inference drawn from this table

* The accuracy of the model is 0.97222222, which indicates that the model is able to correctly classify 97% of the test data.
* The precision of the model is 0.98, which indicates that when the model predicts a positive class, it is correct 98% of the time.
* The recall of the model is 0.90, which indicates that the model is able to identify 90.0% of the positive instances in the test data.
* The F1 score of the model is 0.94, which is the harmonic mean of precision and recall, and provides a balanced measure of the model's performance.
* The parameters used for the classification model include Bag of Words, TF-IDF.
* The performance metrics are calculated for both binary and multi-class classification, as indicated by the presence of both precision and recall values.
* The highest value in the table is for the F1 score of the Bag of words model, which is 9.4, indicating perfect performance for that model.
* From above data we can conclude that bag of words classification is best suited for this dataset

**Results obtained from experiment:**

Bag of words: 0.0817 seconds

TI-IDF: 0.0846 seconds

Word2Vec: 0.7867 seconds

The trade-offs between performance and time complexity in text classification methods depend on the vectorization techniques used.

**1) Words in a Bag (BOW):**

**Performance**:

Because BOW is so basic, it performs exceptionally well. Word counts are determined by looping over the manuscript and increasing counters, which are easily vectorized.  
**Time Complexity:**

BOW's document vectorization process has a linear time complexity (O(n)). This indicates that processing time increases in direct proportion to document size. This can be made much more efficient by vectorization, which processes word counts for the entire page at once.  
**Trade-off:**

BOW is quick, but by omitting context and word order, it loses semantic information.

**2) TF-IDF, or Term Frequency-Inverse Document Frequency:**

**Performance:**

TF-IDF requires extra computations for document frequency (DF) and IDF scores, building on BOW. This has a little bit more overhead than BOW. Vectorization can still be used, nevertheless, to quickly calculate TF and IDF for every word in a document.   
**Time Complexity**:

TF-IDF is still linear (O(n)) for document vectorization, but it has a little greater time complexity than BOW. When handling big datasets, vectorization keeps processing speed high.   
**Trade-off:**

By weighting terms according to their relevance, TF-IDF provides a more informative document representation than BOW, but it has a modest performance impact.

**3) Word2Vec:**

**Performance:**

Word2vec involves training a neural network, which is computationally expensive. The training phase can be slow, especially for large datasets. However, once the word embeddings are trained, generating vector representations for new documents is relatively fast.

**Time Complexity:**

Training word2vec has a higher time complexity, often ranging from O(n log n) to O(n^2) depending on the specific algorithm (e.g., Skip-gram vs. CBOW) and training parameters. Vectorization is used within the training process to optimize computations, but it doesn't significantly impact document vectorization itself (which is faster).

**Trade-off:**

Word2vec sacrifices training speed for a richer representation. The trained word embeddings capture semantic relationships between words, leading to better performance in tasks like document similarity or sentiment analysis.

**RESULT:**

From this assignment we explored word vectorization method in NLP and perform three conventional methods for word vectorization. Moreover we made and analysis of these methods based on classification parameters, computational time required. We also discussed trade-offs and time complexity of these methods.

**Summary**

This project explores the impact of word vectorization techniques on real life problems like sentiment analysis. We also perform an experiment. It utilizes a movie review dataset and implements three methods: Bag-of-Words, TF-IDF. Each method will be evaluated on its ability to convert text into useful representations for a sentiment classifier (logistic regression) while considering factors like classification accuracy, processing speed, and model generalizability. By comparing these methods, the project aims to understand the trade-offs between performance and time complexity inherent to different word vectorization approaches.