

Machine Learning

MOD006562

Faculty: Science and Engineering

School: Computing and Information Science

SID Number: 2217939

Academic Year: 2023/24

Trimester: 2

Contents

[Introduction 4](#_Toc164015086)

[Data Preparation and Statistical Analysis 4](#_Toc164015087)

[Initial Exploratory Data Analysis 4](#_Toc164015088)

[Data Preparation 7](#_Toc164015089)

[Data Pre-Processing. 8](#_Toc164015090)

[Infrequent Classes and Down Sampling 10](#_Toc164015091)

[Text Normalization and Cleaning 11](#_Toc164015092)

[Train-Test Split 14](#_Toc164015093)

[Cross-Validation 14](#_Toc164015094)

[Vectorization of Data 14](#_Toc164015095)

[Bag of Words Vectorization 15](#_Toc164015096)

[TF-IDF Vectorization 15](#_Toc164015097)

[Hashing Vectorization 15](#_Toc164015098)

[Training and Evaluating Models 16](#_Toc164015099)

[Multinomial Naive Bayes (MNB) 16](#_Toc164015100)

[Model Description 16](#_Toc164015101)

[Evaluation Design 17](#_Toc164015102)

[Outcomes 17](#_Toc164015103)

[XGBoost 18](#_Toc164015104)

[Model Description 18](#_Toc164015105)

[Evaluation Design 19](#_Toc164015106)

[Outcomes 19](#_Toc164015107)

[Support Vector Classifier 20](#_Toc164015108)

[Model Description 20](#_Toc164015109)

[Evaluation Design 20](#_Toc164015110)

[Outcomes 20](#_Toc164015111)

[Results/Evaluation Analysis 21](#_Toc164015112)

[References 23](#_Toc164015113)

[Appendix 25](#_Toc164015114)

[Appendix A: Word2Vec Comparison 25](#_Toc164015115)

[Appendix B: Comparison Multinomial NB over Gaussian NB 26](#_Toc164015116)

[Appendix C: Multinomial Naïve Bayes Model Results 28](#_Toc164015117)

[Appendix D: XGBoost Model Results 31](#_Toc164015118)

[Appendix E: SVC Hyperparameter Tuning 34](#_Toc164015119)

[Appendix F: SVC Model Results 36](#_Toc164015120)

# Introduction

The project aims to create machine models which can classify new into the distinct categories. The initial dataset consists of over 200,000 headlines and 41 different categories. In order to classify news, special Natural Language Processing (NLP) techniques were used, such as, data preparation, text pre-processing, down sampling, feature extraction, and others. Consequently, three machine learning models were selected and fit. The performance of each model was evaluated, and outcome produced.

# Data Preparation and Statistical Analysis

## Initial Exploratory Data Analysis

Before pre-processing, an exploratory data analysis (EDA) was done. It aided with understanding of the dataset structure and its balance. The "countplot" function of "seaborn" library was used to create a plot and "matplotlib" was used to represent plot in more clear way (Figure 2). The plot visualized all categories, distribution of articles, and demonstrated huge class imbalance. In addition, manual exploration of excel file was conducted, in order to understand data quality and text format. The dataset initially consists of over 200,000 headlines and 41 categories. The first step of data preparation was a clear overview of the dataset. In order to commit an overview, Pandas 'head', 'tail', 'info', and 'nunique' methods were used (Figure 1, Figure 3, Figure 4, Figure 5).

A screenshot of a computer

Description automatically generated

Figure 1: Demonstrates head method output

A graph of numbers and names

Description automatically generated with medium confidence

Figure 2: Shows initial news distribution

A screenshot of a black and white list

Description automatically generated

Figure 3: Shows output of the tail method

A screenshot of a computer

Description automatically generated

Figure 4: Illustrates information table of the dataset

A black and white text on a black background

Description automatically generated

Figure 5: Shows categories namely and their total number

## Data Preparation

Records that had any column empty were dropped, and any duplicates were dropped (Figure 5). Moreover, after careful consideration, some columns were considered irrelevant and dropped, such as ‘Unnamed: 0’, ‘link’, 'date', and ‘authors’ (Figure 7). ‘Unnamed: 0’ is a column which automatically generated since the Pandas 'read\_excel’ is used to import dataset, while others do not contribute to the category classification.

A screenshot of a computer

Description automatically generated

Figure 6: Demonstrates cleaning from missing values and duplicates

A screenshot of a computer program

Description automatically generated

Figure 7: Demonstrates drop of irrelevant features

# Data Pre-Processing.

In the phase of pre-processing, the categories that shared themes were grouped together into distinct categories. This helped simplify the classification task by reducing the number of target classes this decision aimed to enhance model accuracy by decreasing the scarcity of separate categories (Figure 8). After grouping categories only 24 distinct categories left (Figure 9).

The dataset underwent further cleaning to create a 'full\_text' feature by concatenating 'headline' and 'short\_description'. The decision was done since headline and description can be considered as parts of full text and it is easier to train NLP model on single column.

A screenshot of a computer program

Description automatically generated

Figure 8: Demonstrates category mapping and column combining

A graph with colorful bars

Description automatically generated

Figure 9: Shows combined categories

## Infrequent Classes and Down Sampling

In order to address the issue of class representation without losing the variety of categories the dataset was adjusted by removing categories with less than 6000 samples. At the time categories with an excess of instances were down sampled to 10,000 records each. Down sampling is important to create a distribution among classes preventing any imbalances, slightly reduce training time, and reduce biases (Figure 10, Figure 11) (RITHP, 2023).

A screenshot of a computer program

Description automatically generated

Figure 10: Shows downsampling stage

A graph with different colored bars

Description automatically generated

Figure 11: Demostrates result of downsampling

## Text Normalization and Cleaning

The last step, in the dataset preparation for feature extraction and modelling is cleaning and normalizing the text. Initially all URLs were removed. Then the lemmatization with the spacy library was used to remove noise and decrease complexity. This approach is particularly effective due to spacy's extensive lexicographical resources, so it is accurate across different word forms (spaCy, 2016). Then stop words were removed, which are typically does not add any sentimental meaning and adding huge noise. Additionally, the characters and punctuation were removed, and all text was converted to lowercase for consistency. It is important to mention that stemming was not used on purpose since spacy's lemmatization capabilities produce desirable result (Figure 12). Consequently, the plot of most frequent words and distribution of word count was created to comprehend normalization and cleaning results (Figure 13, Figure 14).

A screen shot of a computer program

Description automatically generated

Figure 12: Shows cleaning process

A graph of a number of words

Description automatically generated

Figure 13: Shows 50 most frequent words in the cleaned dataset

A graph of a number of blue and black bars

Description automatically generated

Figure 14: Illustrates distribution of word counts in the dataset

## Train-Test Split

The dataset was divided into two parts with 80% set for training and the remaining 20%. This split was decided upon to give the model data for learning while allowing keeping a significant portion for an unbiased evaluation of its predictive performance (Khanna, 2023). As a result, over 57000 articles were selected for training and over 14000 ones for testing and evaluation (Figure 15).

A screenshot of a computer program

Description automatically generated

Figure 15: Shows train-test split

## Cross-Validation

While cross validation is not used during the train test split it is essential, for evaluating produced model. During training the cross validation will be used to test how well the model performs on subsets of the training data. This method helps to confirm that the model performs reliable and robust (SciKit-Learn, 2009). The practical usage of cross validation and its outcomes will be described in the "Training and Evaluating Models" section.

# Vectorization of Data

In the project, three different feature extraction techniques were used, such as Bag of Words Vectorization, TF-IDF, and Hashing Vectorization. They were used transform textual data into number which are suitable for machine learning models. Additionally, the use of Word2Vec was checked as an additional approach to understand and compare with other techniques; however, its accuracy was unsatisfied (see Appendix A: Word2Vec Comparison). In addition, selected vectorization techniques were integrated into the pipeline with careful consideration of the parameters, n-grams were selected to be 1,1 (Figure 16).

## Bag of Words Vectorization

Bag of Words Vectorization works by analysing word frequencies in the text. It works well in situations where there is a link between the frequency of certain words and the themes of documents making it an ideal option, for basic classification tasks.

## TF-IDF Vectorization

TF IDF Vectorization improves the Bag of Words Vectorization by assigning weights to words according to how they appear in documents emphasizing the importance of terms, in differentiating between categories. This approach fine tunes our set of features enhancing model accuracy by giving preference to words that provide the value for specific categories.

## Hashing Vectorization

The Hashing Vectorization method was chosen because the project requires scalability and efficiency. The Hashing Vectorization demonstrates an approach, to text processing ensuring that our workflow remains flexible and efficient even as dataset sizes increase (Editor, 2024).

A screen shot of a computer program

Description automatically generated

Figure 16: Shows vectorization techniques integrated with the pipeline

# Training and Evaluating Models

## Multinomial Naive Bayes (MNB)

### Model Description

The Multinomial Naive Bayes (MNB) was chosen over Gaussian Naive Bayes due to its effectiveness, in handling data like word frequencies, which are more aligned with the type of textual data typically seen in document classification assignments. It uses the Bayes theorem with the assumption of conditional independence between every pair of features given the category. In addition, MNB was compared to the Gaussian NB in order to prove that MNB works better on discrete values (See Appendix B: Comparison Multinomial NB over Gaussian NB).

### Evaluation Design

In order to assess how well the Multinomial Naive Bayes model performs with text classification, the feature extraction techniques such Bag of Words Vectorization, TF IDF Vectorization, Hash Vectorization were used. The dataset based on the vectorization technique used into training and testing sets with an 80 20% split. The accuracy was assessed using 5-fold cross-validation to ensure the reliability and stability of the performance metrics. Additionally, the classification report provided conducted and confusion matrix created.

### Outcomes

The outcomes show that Bag of Words slightly outperforms TF-IDF, indicating word frequency's significance in category prediction. Hashing shows worse results due to compatibility issues with Multinomial Naive Bayes (Table 1). For the detailed classification reports and heatmaps for each vectorization method see Appendix C: Multinomial Naïve Bayes Model Results

|  |  |  |  |
| --- | --- | --- | --- |
| Vectorization Method | Mean Accuracy | Cross-Validation Score | Description of Results |
| TF-IDF | 78.15% | 5-fold: Approx. 78.15% | Effectiveness shown in prioritizing words according to the news category, indicating the significance for news categorization since different categories mostly operate with different vocabulary |
| Bag of Words | 79% | 5-fold: Approx. 79% | Slight improvement over TF-IDF suggests that word frequency is a dominant factor in this dataset for predicting categories. |
| Hashing | 76% | 5-fold: Approx. 76.14% | Lower performance due to Multinomial NB's inability to process negative values produced by the hashing technique. Despite, hashing vector was configured to alternate values, it may still be a point of concern (Figure 17). |

Table 1: Shows results of Multinomial NB with different vectorization techniques

A computer screen with text and images

Description automatically generated

Figure 17: Shows hashing vector sign alternation

## XGBoost

### Model Description

XGBoost is stands for eXtreme Gradient Boosting. It was chosen for its effectiveness, in handling a variety of datasets. This machine learning algorithm is based on decision trees, additionally it utilizes a boosting framework to convert weak learners into single strong learner (Nelson, 2019). When it comes to classifying text XGBoost stands out for its capability to handle data commonly found in text vectorization results. This choice was done due to the need for a robust, scalable model capable to capture diverse patterns in high-dimensional data.

### Evaluation Design

Similar to Multinomial NB, the model was evaluated using cross-validation to ensure robustness, with separate assessments across TF-IDF, Bag of Words, and Hashing Vectorizations. This approach was intended to identify the most effective data representation for optimizing classification accuracy. In addition, the heatmap and classification report for each case was created.

### Outcomes

XGBoost displayed constant result across all vectorization techniques, with mean accuracies around 75%, showing its strong adaptability to different textual data representations (See Table 2). For the detailed classification reports and heatmaps for each vectorization method see Appendix D: XGBoost Model Results

|  |  |  |  |
| --- | --- | --- | --- |
| Vectorization Method | Mean Accuracy | Cross-Validation Score | Description of Results |
| TF-IDF | 75.29% | 5-fold: Approx.  74.8% | Shows XGBoost ability to maintain accuracy with TF-IDF vectorization due to its gradient boosting mechanism. |
| Bag of Words | 75.49% | 5-fold: Approx.  75.2% | Shows that the frequency of terms aligns effectively with XGBoosts enhancement of classified samples demonstrating the models resilience to unprocessed textual characteristics.. |
| Hashing | 75.79% | 5-fold: Approx.  74.95% | Despite the reduction, in data size and possible loss of information when using Hashing Vectorization XGBoosts technique makes up for it by iterative mistake fixes during the boosting process. It demonstrates the adaptability to work with sparse and complex data sets. |

Table 2: Shows results of XGBoost across different vectorization techniques

## Support Vector Classifier

### Model Description

The Support Vector Classifier (SVC) was chosen due to its ability to work huge and sparse datasets; consequently, SVC is often chosen for the text categorization tasks. The main SVC is to find a hyperplane in an N-dimensional space that distinctly classifies the data points, which differs it from Linear Regression. The effectiveness of SVC can be easily improved using GridSearchCV for the hyperparameter tuning (www.w3schools.com, n.d.). Using GridSearchCV it was defined that in context of the project best parameters (Figure 18).

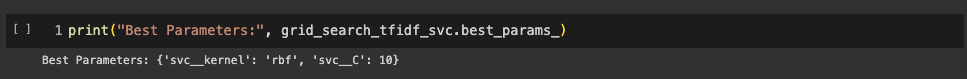


Figure 18: Shows best parameters for SVC models

### Evaluation Design

The evaluation model of the SVC model is design to verify validity of the results. The cross-validation was employed as one of the simplest and robust ways to prove accuracy of different parts of the set. As it mentioned before GridSearchCV was used to identify the best kernel (see Appendix E: SVC Hyperparameter Tuning). In addition, classification report and heatmap was produced.

### Outcomes

In the outcomes section, it is observed how the Support Vector Classifier performs with various text-to-data methods. All the models have achieved good accuracies and cross-validation. Detailed classification reports and heatmaps are included in Appendix F: SVC Model Results for further examination.

|  |  |  |  |
| --- | --- | --- | --- |
| Vectorization Method | Mean Accuracy | Cross-Validation Score | Description of Results |
| TF-IDF | 79.62% | 5-fold: Approx.  74.9% | Shows accuracy when using TF IDF vectorization taking advantage of the RBF kernels capability to deal with non-linear data distinctions. |
| Bag of Words | 79.23% | 5-fold: Approx.  78.45% | The models effectiveness with term frequencies highlights its management of unprocessed text characteristics showing stable cross validation results. |
| Hashing | 79.83% | 5-fold: Approx.  78.35% | Hashing vectorization succussed due to SVC ability to manage the reduced and transformed feature set, achieving the best mean accuracy by effectively correcting classification errors. |

# Results/Evaluation Analysis

The project proposed three different machine learning models combined with three different vectorization techniques. The Multinomial Naive Bayes (MNB) which is typically strong when it comes to analysing word frequencies, in text data did not perform well with all of vectorization techniques. It showed potential with TF IDF, Bag of Words; however, struggled with hashing vectorization probably due to incompatible vectors generated.

On the hand, XGBoost showed the consistency across all vectorization techniques. reliability and adaptability consistently delivered results regardless of the vectorization method used. Its gradient boosting mechanism showed the robustness and adaptability and allowed it to adjust well to ways of representing text.

The Support Vector Classifier (SVC) turn out to be the most appropriate model for this task, with its capability to navigate the complexity of high-dimensional spaces effectively. Through hyperparameter tuning using GridSearchCV it was discovered that an RBF kernel combined with a regularization strength of C=10 worked best for this task. This configuration shows a balance, between classifying training examples accurately and ensuring a model that generalizes well without overfitting.

Overall, the project has come up with three model with constant performance above 75% (Figure 19); outlined positive and negative of certain models, vectorization techniques, and their compatibilities.

A graph of different colored bars

Description automatically generated

Figure 19: Demonstrates accuracies for all models tested

# References

Awasthi, S. (2020). *Seven Most Popular SVM Kernels*. [online] Dataaspirant. Available at: https://dataaspirant.com/svm-kernels/.

Editor, C. (2024). *What is Hashing Vectorizer: Artificial Intelligence Explained*. [online] chatgptguide.ai. Available at: https://www.chatgptguide.ai/2024/02/26/what-is-hashing-vectorizer/#:~:text=The%20Hashing%20Vectorizer%20is%20scalable [Accessed 11 Apr. 2024].

Fatima, A. (2021). *Template for Machine Learning Assignment*.

Khanna, S. (2023). *A Comprehensive Guide to Train-Test-Validation Split in 2023*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2023/11/train-test-validation-split/.

Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space*. [online] Available at: https://arxiv.org/pdf/1301.3781.pdf.

Nelson, D. (2019). *Gradient Boosting Classifiers in Python with Scikit-Learn*. [online] Stack Abuse. Available at: https://stackabuse.com/gradient-boosting-classifiers-in-python-with-scikit-learn/.

Ray, S. (2017). *Naive Bayes Classifier Explained: Applications and Practice Problems of Naive Bayes Classifier*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/#:~:text=Gaussian%20Naive%20Bayes%3A%20gaussiannb%20is [Accessed 11 Apr. 2024].

RITHP (2023). *Choosing the Right Size: A Look at the Differences between Upsampling and Downsampling Methods*. [online] Medium. Available at: https://medium.com/@rithpansanga/choosing-the-right-size-a-look-at-the-differences-between-upsampling-and-downsampling-methods-daae83915c19#:~:text=Downsampling%20is%20a%20method%20used [Accessed 11 Apr. 2024].

SciKit-Learn (2009). *3.1. Cross-validation: evaluating estimator performance — scikit-learn 0.21.3 documentation*. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/cross\_validation.html.

spaCy (2016). *spaCy 101: Everything you need to know · spaCy Usage Documentation*. [online] spaCy 101: Everything you need to know. Available at: https://spacy.io/usage/spacy-101.

www.w3schools.com. (n.d.). *Python Machine Learning - Grid Search*. [online] Available at: https://www.w3schools.com/python/python\_ml\_grid\_search.asp.

# Appendix

## Appendix A: Word2Vec Comparison

Initially Word2Vec was selected as one of the feature extracting techniques. However, based on the provided theoretical analyses regarding word counts in the articles (Figure 14) it was assumed that Word2Vec will be limited due to documents containing an average of 15-20 words. Word2Vecs reliance on having a lot of contexts to create representations (Mikolov et al., 2013) made it less suitable for the current task. In addition, practical comparison was conducted using Support Vector Classifier as a model (Figure 20, Figure 21, Figure 22).

A screen shot of a computer program

Description automatically generated

Figure 20: Shows code snippet for Word2Vec

A screenshot of a computer screen

Description automatically generated

Figure 21: Classification report for Word2Vec

A screenshot of a computer screen

Description automatically generated

Figure 22: Shows classification report for count vectorization

## Appendix B: Comparison Multinomial NB over Gaussian NB

The Gaussian Naive Bayes model assumes that the values of features are normally distributed. This does not always match the distribution of word frequencies or TF IDF scores for the text data. This assumption may not work well when categorizing text especially when the feature values are distinct and follow a type of distribution (Ray, 2017). The classification report for TF-IDF vectorized data using Gaussian NB shows its limitations (Figure 23, Figure 24). As a result, Multinomial NB was selected as it is better suited for handling the characteristics of text.

A screenshot of a computer

Description automatically generated

Figure 23: Shows classification report for TF-IDF and Gaussian NB

A graph showing a comparison of a performance

Description automatically generated with medium confidence

Figure 24: Shows Multinomial NB accuracy comparison with Gaussian NB

## Appendix C: Multinomial Naïve Bayes Model Results

A screenshot of a computer program

Description automatically generated

Figure 25: Shows classification report and cross-validated accuracy for TF-IDF Vectorization and Multinomial Naïve Bayes

A graph with numbers and a bar chart

Description automatically generated with medium confidence

Figure 26: Demonstrates confusion matrix for TF-IDF vectorization and Multinomial Naïve Bayes

A screenshot of a computer

Description automatically generated

Figure 27: Shows classification report and cross-validated accuracy for Bag of Words Vectorization and Multinomial Naïve Bayes

A graph with blue squares

Description automatically generated

Figure 28: Shows confusion matrix for Bag of Words vectorization and Multinomial Naïve Bayes

A screenshot of a computer

Description automatically generated

Figure 29: Shows classification report and cross-validated accuracy for Hashing Vectorization and Multinomial Naïve Bayes

A graph with blue squares

Description automatically generated

Figure 30: Shows heatmap for the Hashning Vector and Multinomial Naïve Bayes

## Appendix D: XGBoost Model Results

A screenshot of a computer screen

Description automatically generated

Figure 31: Shows classification report and cross-validated accuracy for TF-IDF and XGBoost

A graph showing different types of information

Description automatically generated with medium confidence

Figure 32: Shows confusion matrix for TF-IDF and XGBoost

A screenshot of a computer program

Description automatically generated

Figure 33: Shows classification report and cross-validated accuracy for Bag of Words Vectorization and XGBoost

A graph with blue squares

Description automatically generated

Figure 34: Shows confusion matrix for Bag of Words Vectorization and XGBoost

A screenshot of a computer program

Description automatically generated

Figure 35: Shows classification report and cross-validated accuracy for Hashing Vectorization and XGBoost

A graph with numbers and a bar chart

Description automatically generated with medium confidence

Figure 36: Shows confusion matrix for Hashing Vectorization and XGBoost

## Appendix E: SVC Hyperparameter Tuning

The RBF kernel and a C (Regularization parameter) value of 10 were chose by GridSearchCV after assessing performance, across kernel functions and regularization strengths. The RBF kernel was probably chosen due to its ability to effectively model non-linear relationships frequently present in high-dimensional text data (Awasthi, 2020). The parameter is used to control margin and classification error, the higher C parameter the less biases may appear, but with risk of model overfitting. The C value of 10 indicates a preference for a model that fits the training data closely while still allowing some level of generalization (Figure 37).

A screenshot of a computer program

Description automatically generated

Figure 37: Shows hyperparameter tuning using GridSearchCV

## Appendix F: SVC Model Results

A screenshot of a computer

Description automatically generated

Figure 38: Shows classification report and cross-validated accuracy for TF-IDF Vectorization and SVC

A graph with numbers and text

Description automatically generated with medium confidence

Figure 39: Illustrates confusion matrix for TF-IDF Vectorization and SVC

A screenshot of a computer program

Description automatically generated

Figure 40: Demonstrates classification report and cross-validated accuracy for Bag of Words Vectorization and SVC

A graph with numbers and a bar chart

Description automatically generated with medium confidence

Figure 41: Shows confusion matrix for Bag of Words Vectorization and SVC

A screenshot of a computer

Description automatically generated

Figure 42: Shows classification report and cross-validated accuracy for Hashing Vectorization and SVC

A graph with numbers and a bar chart

Description automatically generated with medium confidence

Figure 43: Shows confusion matrix for Hashing Vectorization and SVC