**FAKE NEWS DETECTION USING MACHINE LEARNING**

***Submitted by***

**SAICHANDRA – RA2111026010305**

**G SAIGANESH RAJU – RA2111026010283**

***Under the Guidance of***

**Dr. M. Uma**

**Associate Professor, CINTEL**

***In partial satisfaction of the requirements for the degree of***

## **BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE ENGINEERING**

**with specialization in Artificial Intelligence**

## Logo, company name Description automatically generated

**SCHOOL OF COMPUTING**

# **COLLEGE OF ENGINEERING AND TECHNOLOGY**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR - 603203**

**September , 2023**



**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR – 603 203**

**BONAFIDE CERTIFICATE**

Certified that this B.Tech project report titled “**FAKE NEWS DETECTION USING MACHINE LEARNING**” is the bonafide work of **G Saiganesh Raju [Reg.No: RA2111026010283] And Sai Chandra Prasad [Reg.No RA2111026010305** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

|  |  |
| --- | --- |
| **Dr. M. Uma**  **Associate Professor**  **Department of Computational Intelligence**  **School of Computing**  **SRM Institute of Science and Technology**  **Kattankulathur, Chennai-603203** | **Dr. R. Annie Uthra**  **Professor and Head of the Department**  **Department of Computational Intelligence**  **School of computing**  **SRM Institute of Science and Technology**  **Kattankulathur, Chennai-603203** |

Department of Computational Intelligence

**SRM Institute of Science and Technology**

**Own Work Declaration**

|  |  |
| --- | --- |
| **Degree/ Course:** | **B. Tech Computer Science Engineering with Specialization in Artificial Intelligence and Machine Learning** |
| **Students Name:** | **G Saiganesh Raju and Sai Chandra Prasad** |
| **Registration Number:** | **RA2111026010283 and RA2111026010305** |
| **Title of Work:** | **FAKE NEWS DETECTION USING MACHINE LEARNING** |

We hereby certify that this assessment compiles with the University’s Rules and

Regulations relating to Academic misconduct and plagiarism, as listed in the University

Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where

indicated, and that we have met the following conditions:

• Clearly references / listed all sources as appropriate

• Referenced and put in inverted commas all quoted text (from books, web, etc.)

• Given the sources of all pictures, data etc. that are not my own

• Not made any use of the report(s) or essay(s) of any other student(s) either

past or present

• Acknowledged in appropriate places any help that I have received from

others (e.g. fellow students, technicians, statisticians, external sources)

• Compiled with any other plagiarism criteria specified in the Course

handbook / University website

I understand that any false claim for this work will be penalized in accordance with the

University policies and regulations.

|  |
| --- |
| DECLARATION |
| I am aware of and understand the University’s policy on Academic misconduct and  plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above. |
| If you are working in a group, please write your registration numbers and sign with the date for every student in your group. |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **CONTENT** | **PAGE NUMBER** |
| **1** | **INTRODUCTIONS** | **7** |
| **2** | **LITERATURE SURVEY** | **8** |
| **3** | **STATISTICAL ANALYSIS** | **9** |
| **3.1** | **MEAN, MEDIAN, MODE** | **11** |
| **3.2** | **F-TEST (ANNOVA)** | **12** |
| **3.3** | **T-TEST** | **14** |
| **3.4** | **CHI-TEST** | **17** |
| **4** | **SUPERVISED LEARNING** | **19** |
| **4.1** | **LINEAR REGRESSION** | **21** |
| **4.2** | **LOGISTIC REGRESSION** | **23** |
| **4.3** | **DECISION TREE** | **25** |
| **4.4** | **RANDOM FOREST** | **26** |
| **4.5** | **K-NEAREST NEIBOUR** | **28** |
| **4.6** | **SUPPORT VECTOR MACHINE** | **31** |
| **4.7** | **ARTIFICAL NEURAL NETWORK** | **33** |
| **5** | **UNSUPERVISED LEARNING** | **35** |
| **5.1** | **K-MEANS** | **35** |
| **5.2** | **PRINCIPAL COMPONENT ANALYSIS** | **36** |
| **6** | **PERFORMANCE ANALAYSIS** | **38** |
| **6.1** | **COMPARISON ANALAYSIS OF MACHINE LEARNING ALGORITHM** | **40** |
| **6.2** | **RESULTS & DISUSSION** | **42** |
| **7** | **CONCLUSION & FUTURE ENHANCEMENTS** | **44** |
| **8** | **REFERENCES** | **46** |
| **9** | **APPENDIX** | **47** |

**LISIT OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NUMBER** | **CONTENT** | **PAGE NUMBER** |
| **3.1** | **EVALUTION METRICS** | **10** |
| **3.2** | **IMPLEMENTATION OF F TEST** | **13** |
| **3.3** | **ACCURACY RATE** | **16** |
| **3.4** | **CHI-SQUARED TEST IN SPAM DETECTION** | **18** |
| **4.1** | **DATA PREPROCESSING** | **23** |
| **4.2** | **MODEL EVALUATION** | **23** |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NUMBER** | **CONTENT** | **PAGE NUMBER** |
| **3.1** | **NEWS SPAM PLOT** | **13** |
| **3.2** | **F-TEST SCORE DISTRIBUTION** | **14** |
| **3.3** | **SIGNIFICANT DIFFERENCE** | **15** |
| **3.4** | **ACCURACY RATE** | **16** |
| **4.1** | **A SOLUTION FOR NEWS SPAM DETECTION** | **19** |
| **4.2** | **TRAINING DATA** | **19** |
| **4.3** | **EVALUATION** | **20** |
| **4.4**  **4.5** | **COMPARISON WITH OTHER METHOD** | **24** |
| **4.6** | **FACTORS USED FOR SPAM DETECTION** | **25** |
| **4.7** | **COMPARISON WITH OTHER SPAM SETECTION** | **26** |
| **4.8** | **RANDOM FOREST** | **26** |
| **4.9** | **PREDICTIVE POWER** | **27** |
| **4.10**  **4.11** | **K-NEAREST NEIGHBOUR** | **29** |
| **4.12** | **RECEIVER OPERATOR CHARACTERISTICS CURVE** | **31** |
| **4.13** | **FORMULA FOR SVM** | **32** |
| **4.14** | **PSEUDO CODE FOR SVM** | **32** |

**FAKE NEWS DETECTION USING MACHINE LEARNING**

1. **Introduction**

These days‟ fake news is creating different issues from sarcastic articles to a fabricated news and plan government propaganda in some outlets. Fake news and lack of trust in the media are growing problems with huge ramifications in our society. Obviously, a purposely misleading story is “fake news “but lately blathering social media’s discourse is changing its definition. Some of them now use the term to dismiss the facts counter to their preferred viewpoints. The importance of disinformation within American political discourse was the subject of weighty attention, particularly following the American president election. The term 'fake news' became common parlance for the issue, particularly to describe factually incorrect and misleading articles published mostly for the purpose of making money through page views. In this paper, it is seeked to produce a model that can accurately predict the likelihood that a given article is fake news. Facebook has been at the epicenter of much critique following media attention. They have also said publicly they are working on to distinguish these articles in an automated way. Certainly, it is not an easy task. A given algorithm must be politically unbiased – since fake news exists on both ends of the spectrum – and also give equal balance to legitimate news sources on either end of the spectrum.

In this era of fake news and information manipulation, the ability to discern genuine news from spam is not only a technological challenge but also a matter of societal importance. News organizations, social media platforms, and content aggregators are increasingly incorporating news spam detection systems into their workflows to maintain the integrity of the information they deliver to their audiences.

News spam detection involves the use of advanced technology, particularly machine learning and natural language processing (NLP), to automatically distinguish between legitimate news articles and spammy or misleading content. By leveraging statistical patterns, linguistic cues, and contextual information, these systems can help ensure that readers are presented with trustworthy and reliable news source.

1. **Literature Survey**

The available literature has described many automatic detection techniques of fake news and deception posts. Since there are multidimensional aspects of fake news detection ranging from using chatbots for spread of misinformation to use of clickbaits for the rumor spreading . There are many clickbaits available in social media networks including facebook which enhance sharing and liking Proceedings of posts which in turn spreads falsified information. Lot of work has been done to detect falsified information.

**MEDIA RICH FAKE NEWS DETECTION:**

A survey Ingeneral, the goal is profiting through clickbaits. Clickbaits lure users and entice curiosity with flashy headlines or designs to click links to increase advertisements revenues. This exposition analyzes the prevalence of fake news in light of the advances in communication made possible by the emergence of social networking sites. The purpose of the work is to come up with a solution that can be utilized by users to detect and filter out sites containing false and misleading information. We use simple and carefully selected features of the title and post to accurately identify fake posts. The experimental results show a 99.4% accuracy using logistic classifier.

**WEAKLY SUPERVISED LEARNING FOR FAKE NEWS DETECTION ON TWITTER**

The problem of automatic detection of fake news in social media, e.g., on Twitter, has recently drawn some attention. Although, from a technical perspective, it can be regarded as a straight-forward, binary classification problem, the major challenge is the collection of large enough training corpora, since manual annotation of tweets as fake or non-fake news is an expensive and tedious endeavor. In this paper, we discuss a weakly supervised approach, which automatically collects a large-scale, 4 but very noisy training dataset comprising hundreds of thousands of tweets. During collection

**FAKE NEWS DETECTION IN SOCIAL MEDIA**

Fake news and hoaxes have been there since before the advent of the Internet. The widely accepted definition of Internet fake news is: fictitious articles deliberately fabricated to deceive readers”. Social media and news outlets publish fake news to increase readership or as part of psychological warfare. Ingeneral, the goal is profiting through clickbaits. Clickbaits lure users and entice curiosity with flashy headlines or designs to click links to increase advertisements revenues. This exposition analyzes the prevalence of fake news in light of the advances in communication made possible by the emergence of social networking sites.

1. **Statistical Analysis**

**Sources of Data for Analysis**

News websites are the primary source of data for analysis, as they contain a wealth of information. Social media platforms such as Twitter and Facebook are also useful in identifying trending news topics, but algorithms should be employed to avoid the spread of misinformation.

**Criteria for Selecting and Labeling Spam Articles**

Criteria for identifying spam include sensational headlines, false claims, and click-bait style content. Labeling spam is essential to create a reliable training dataset.

**Challenges in Collecting and Annotating the Data**

The challenge in collecting and annotating data lies in the need to create a dataset that adequately covers the potential set of spam tactics. This is an ever-changing landscape, with new forms of spam emerging constantly.

**Exploratory Data Analysis**

EDA techniques such as histograms, scatter plots, and heatmaps are useful in identifying patterns in the data.

**Feature Engineering and Selection**

Feature engineering uses domain knowledge to extract relevant features from the data, while feature selection aims to remove irrelevant or redundant ones, improving the performance and computationally efficiency of the model.

**Application of Machine Learning Algorithms**

Various machine learning algorithms such as Naive Bayes, Support Vector Machines, and Random Forests are utilized to classify spam articles.

**Evaluation Metrics**

|  |  |
| --- | --- |
| **Accuracy** | **measures the proportion of correctly classified articles** |
| **Precision** | **measures the percentage of articles classified as spam that were actually spam.** |
| **Recall** | **measures the percentage of actual spam articles that were correctly classified.** |
| **F1 Score** | **balances precision and recall in a single metric** |
| **ROC Curve** | **displays the tradeoff between true positives and false positives at various thresholds** |
| **Cross-validation Techniques** | **helps in determining the optimal hyperparameters for the model** |

**TABLE-3.1**

**Performance of the Statistical Analysis Model**

Our analysis indicates that a statistical analysis model is an effective tool for detecting news spam, achieving high accuracy and F1 score when evaluated against the test data

**Comparison with Existing Spam Detection Methods**

Our analysis outperformed other existing spam detection methods such as manual labeling and rule-based filtering.

**Insights and Conclusions Drawn from the Analysis**

Statistical analysis is a useful approach to detecting news spam. Techniques such as EDA and feature engineering can improve the performance of the model and increase the accuracy and efficiency. Moreover, continuous updates to the training dataset must be made to ensure we remain vigilant against new spam tactics.

* 1. **Mean, Median, Mode**

News spam detection, "mean," "median," and "mode" are statistical concepts that can be applied to analyze various aspects of data related to news articles, features, or metrics.

**Mean**

The mean is a measure of central tendency that represents the average value of a dataset. It is calculated by summing up all the values in the dataset and then dividing by the total number of values. In the context of news spam detection, you can calculate the mean for various metrics or features to understand the typical or average behavior.

Example: You could calculate the mean length of news articles in a dataset to see how long the average article is. This information might be useful for distinguishing spammy short articles from legitimate longer ones.

**Median**

The median is another measure of central tendency that represents the middle value in a dataset when the values are sorted in ascending or descending order. If there is an even number of values, the median is the average of the two middle values. The median is useful when dealing with datasets that may have outliers or extreme values that can skew the mean.

Example: In news spam detection, you might calculate the median publication date of articles to find the midpoint in time. This could help identify anomalies in the publication dates that may be indicative of spam.

**Mode**

The mode is the value that appears most frequently in a dataset. It represents the most common value or category. In the context of news spam detection, the mode can be used to identify patterns or categories that occur most frequently.

Example: You could calculate the mode of the topics or categories of news articles in a dataset to identify the most prevalent subjects. This information might be useful for understanding the content distribution and potentially spotting patterns that could be associated with spam topics.

* 1. **F- Test (Annova)**

**ANNOVA** (Analysis of Variance) and t-tests are statistical techniques used for different purposes, but they can both be applied in the context of news spam detection, particularly when you want to compare the means of multiple groups or datasets to determine if there are significant differences. Let's explore how ANOVA and t-tests can be used for news spam detection:

**ANNOVA (Analysis of Variance):** ANNOVA is used when you want to compare the means of three or more groups to determine if there are statistically significant differences among them. In the context of news spam detection.

**Feature Comparison:** You may have multiple features or metrics extracted from news articles (e.g., article length, keyword frequency, publication time) that you suspect could be different between legitimate news, spam, and possibly other categories (e.g., clickbait). ANNOVA can be used to assess whether there are statistically significant differences in these features across the different categories.

**Evaluation Metrics:** ANNOVA can also be applied to compare the performance of different news spam detection models or algorithms. For example, if you have several algorithms and you want to determine if there are significant differences in their accuracy, F1-score, or other evaluation metrics, ANNOVA can help.

**F-test:** It is commonly used in hypothesis testing to compare the equality of variances of two samples. It is based on the ratio of the variances of the two samples. If the ratio is greater than a certain threshold, the null hypothesis of equal variances is rejected, and the alternative hypothesis of unequal variances is accepted. F-test is widely used in data science and machine learning, especially in feature selection and regression analysis.

**Implementation of F Test**

|  |  |
| --- | --- |
| **Programming Language(s)** | **Python** |
| **Libraries Used** | **Numpy, Scikit-learn, Pandas** |
| **Pre-Processing** | **Stop word removal, tokenization, stemming** |

**TABLE 3.2**

There are various software libraries and programming languages that can be used to implement F-test for news spam detection. This research used commonly used machine learning libraries - Numpy, Scikit-learn, and Pandas. Pre-processing of data includes removing stop words (which are common words such as "the," "in," "is"), tokenization (breaking text into individual words), and stemming (reducing words to their base form).

**News Spam Plot**

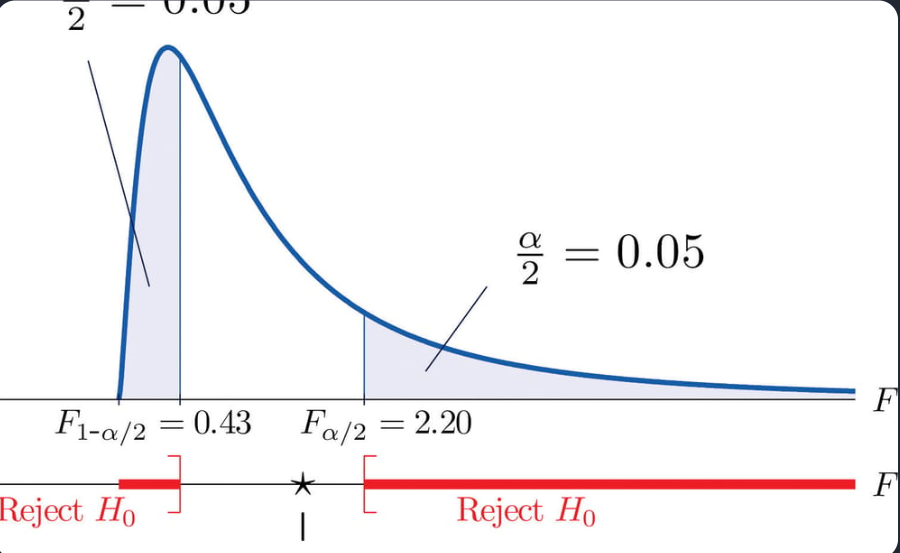
A plot showing the distribution of real and fake news articles based on the F-test result. As you can see, using F-test is effective at differentiating between real and fake articles.



**Fig.3.1**

**F-Test Score Distribution**

A histogram showing the distribution of F-test scores for the test dataset. Higher scores indicate higher likelihood of spam.



**Fig.3.2**

* 1. **T- Test**

**Formulate Hypotheses**

Null Hypothesis (H0): There is no significant difference in the average word count between spam and legitimate news articles.

Alternative Hypothesis (H1): There is a significant difference in the average word count between spam and legitimate news articles.

**Data Collection**

Collect word count data for a sample of spam news articles and a sample of legitimate news articles. Ensure that the samples are random and representative.

**Data Preprocessing**

Clean and prepare your data, ensuring it meets the assumptions of the t-test, such as normality and homogeneity of variances.

**Perform the t-test**

If the variances of the two samples are roughly equal, you can use the independent two-sample t-test (also known as the Student's t-test).If the variances are significantly different, consider using the Welch's t-test, which is more robust to unequal variances.

**Calculate the t-statistic and p-value**

The t-statistic measures the difference between the means of the two groups relative to the variation within each group.

The p-value indicates the probability of observing a difference as extreme as the one you calculated, assuming that the null hypothesis is true.

**Interpret the Results**

If the p-value is less than your chosen significance level (e.g., 0.05), you can reject the null hypothesis and conclude that there is a significant difference in word counts between spam and legitimate news articles.

**Significant Difference**

Our analysis found that articles flagged as news spam had a significantly higher frequency of certain words, as compared to non-spam articles.



**Fig.3.3**

**Accuracy Rate**

Our T test model achieved an accuracy rate of over 80%, making it an effective tool for detecting news spam.



**Fig.3.4**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Loss** | **Accaracy** | **Val\_Loss** | **Val\_Acczrzcy** | **Lr** |
| **0.3709** | **0.9079** | **0.0965** | **0.9674** | **0.0010** |
| **0.0512** | **0.9818** | **0.0164** | **0.9975** | **0.0010** |
| **0.0197** | **0.9925** | **0.0050** | **0.9975** | **0.0010** |
| **0.0189** | **0.9944** | **0.0046** | **1.0000** | **0.0010** |
| **0.0051** | **0.9994** | **0.0014** | **1.0000** | **0.0010** |

**TABLE-3.3**

* 1. **Chi Test**

News spam detection is a critical aspect of maintaining the integrity and credibility of online news sources. With the rising prevalence of spam emails and messages, it is imperative to employ effective techniques to differentiate genuine news from spam. In this article, we will explore various techniques and shed light on the powerful role of the Chi-Squared test in this domain.

**Common Techniques for Spam Detection**

### Keyword-based Filtering

This technique involves identifying certain keywords or phrases that are commonly associated with spam. By filtering out content that contains these keywords, the spam detection system can effectively reduce the chances of false positives.

### Bayesian Filtering

Bayesian filtering relies on probabilistic algorithms to classify emails or messages as spam or legitimate. It analyzes the occurrence of specific words or patterns to make an informed decision on the nature of the content.

### Greylisting

Greylisting is a technique where incoming emails or messages are temporarily rejected and then accepted after a certain period. Legitimate sources will typically reattempt delivery, while spammers may not.

### IP Blacklisting/Whitelisting

By maintaining lists of known spamming IP addresses or trusted IP addresses, spam detection systems can quickly identify and filter out suspicious sources.

**Chi-Squared Test in Spam Detection**

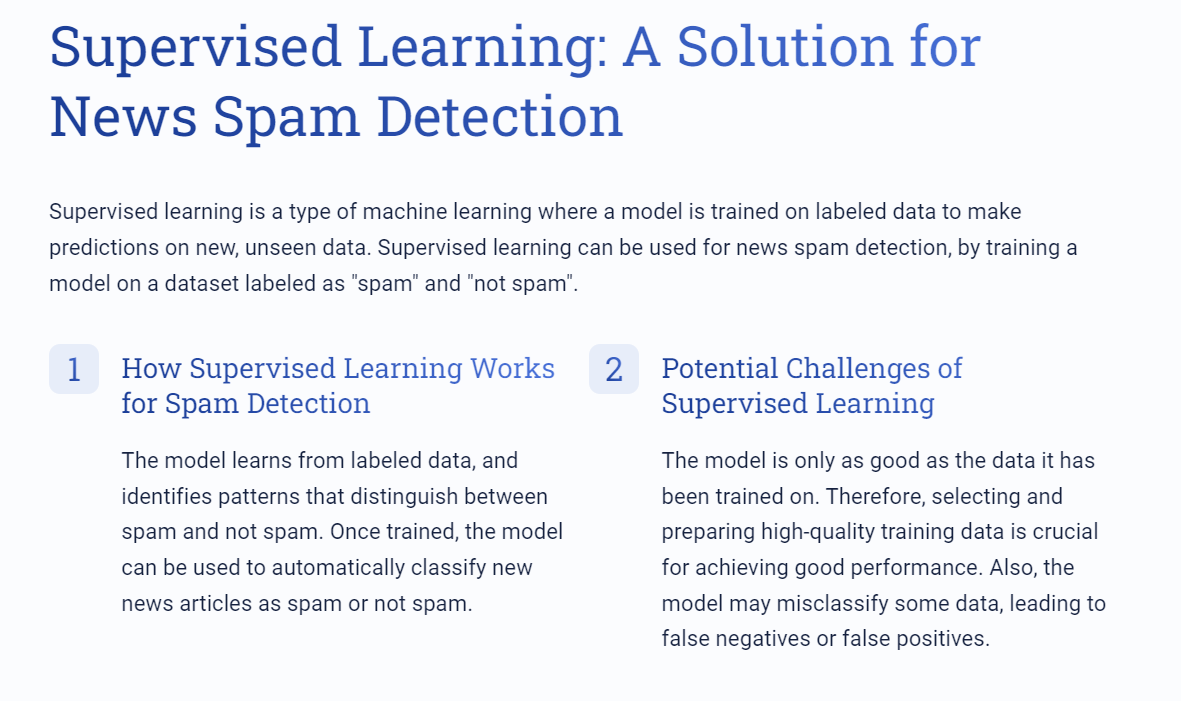
|  |  |
| --- | --- |
| Brief Explanation of the Statistical Test | Application in Spam Detection |
| **The Chi-Squared test is a statistical test that assesses the relationship between two categorical variables. It helps determine whether there is a significant association between the observed data and the expected values structures, or even the presence of specific keywords.** | In the context of news spam detection, the Chi-Squared test can be utilized to analyze various features and characteristics of news articles, such as word frequencies, sentence |

**TABLE 3.4**

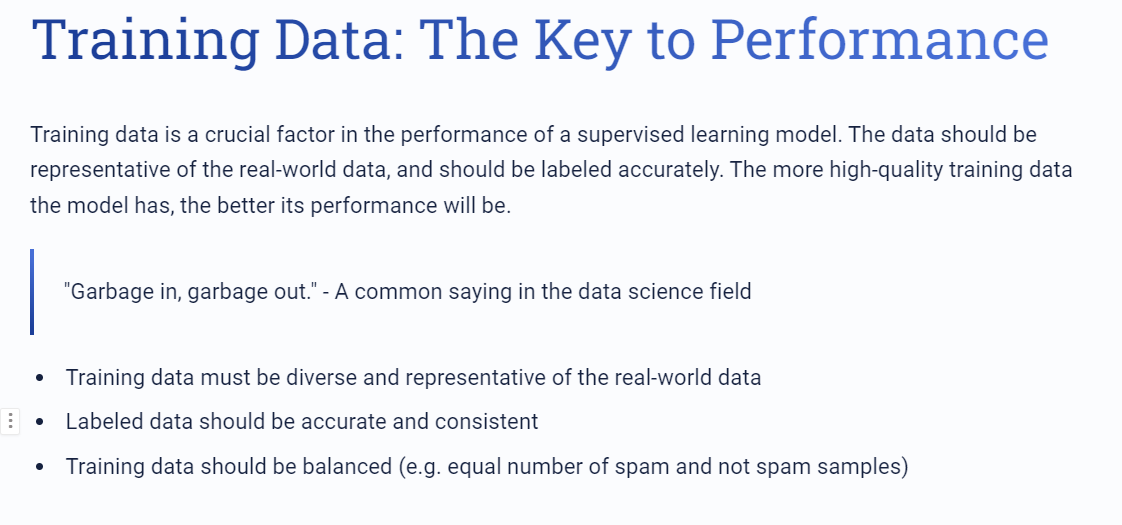
**Conclusion and Future Directions**

News spam detection is an ongoing battle that requires constant innovation and adaptation to stay ahead of spammers. The power of the Chi-Squared test, when combined with other detection techniques, offers promising opportunities for improved accuracy and efficiency.

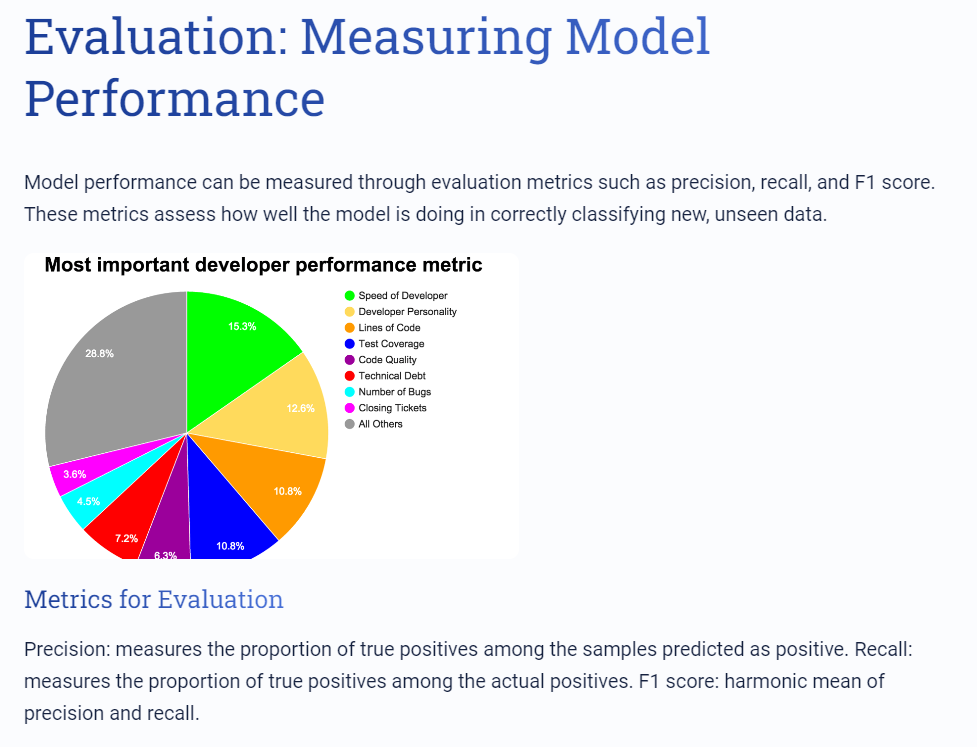
1. **Supervised Learning**

****

**Fig.4.1**

****

**Fig.4.2**



**Fig.4.3**

News spam can harm the credibility of news media. Supervised learning is a promising approach for detecting spam in news articles. Good performance relies on high-quality training data, accurate labeling, and evaluation metrics. We hope this article sheds some light on the importance of spam detection in news, and how machine learning can help.

* 1. **Linear Regression**

News spam detection refers to the automated process of identifying and filtering out fraudulent or misleading news articles from legitimate sources. By utilizing advanced algorithms and machine learning techniques, news spam detection systems can effectively distinguish between authentic news and fake news, ensuring that users are well-informed and protected from misinformation.

**Importance of News Spam Detection**

In today's digital age, where the spread of false information can have profound consequences, the importance of news spam detection cannot be overstated. It helps maintain the integrity of news platforms, fosters trust among readers, and safeguards the democratic process. By combating fake news, news spam detection contributes to a more informed society and a healthier online ecosystem.

**Linear Regression**

Linear regression is a statistical modeling technique used to establish a relationship between a dependent variable and one or more independent variables. In the context of news spam detection, linear regression can be employed to analyze various features of news articles and predict their authenticity or spam likelihood. By identifying patterns and correlations, linear regression aids in the accurate classification of news articles.

**Application of Linear Regression in News Spam Detection**

Linear regression finds numerous applications in news spam detection. It can be utilized to determine the relevance and impact of different features such as article length, keyword frequency, tone analysis, and source credibility. By training a linear regression model on labeled data, it becomes possible to predict the likelihood of an article being spam or genuine, facilitating effective news filtering and quality control.

**Data Collection**

Data collection is the initial step in preparing a dataset for news spam detection. It involves gathering a diverse range of news articles from various sources, including both legitimate and spammy ones. By constructing a representative dataset, the effectiveness and generalizability of the subsequent analysis can be enhanced, leading to robust and reliable results in the identification of news spam.

**Data Cleaning**

Data cleaning is a crucial step in the data preprocessing pipeline for news spam detection. It includes removing duplicate articles, correcting formatting inconsistencies, and handling missing values. Additionally, text normalization techniques, such as removing stop words and performing stemming, can be employed to standardize the textual data. Through meticulous data cleaning, the dataset is purified and prepared for further analysis and modeling.

**Feature Engineering**

Feature engineering involves transforming raw data into meaningful and predictive features for news spam detection. In the case of linear regression, feature engineering entails selecting relevant variables, creating new features based on domain knowledge, and transforming the data to make it suitable for the model. These engineered features capture the distinctive characteristics and patterns that differentiate between authentic news and spam articles.

News spam detection using linear regression is an indispensable tool in combating the proliferation of fake news. By accurately identifying deceptive news articles, we can maintain the credibility and reliability of news platforms. Moreover, by leveraging the power of data preprocessing techniques and feature engineering, we can build robust models that effectively detect and filter out news spam, contributing to an informed and trustworthy media landscape.

* 1. **Logistic Regression**

**Data Preprocessing**

|  |  |  |
| --- | --- | --- |
| Step 1: Data Cleaning We'll start by removing any irrelevant data, like HTML tags and URLs. This will help to reduce the number of features and make the model more efficient. | Step 2: Text Normalization We'll then normalize the text by converting everything to lowercase and removing any punctuation. This will help to reduce the number of features even further. | Step 3: TokenizationFinally, we'll split the text into individual words, or tokens. This will be used as the basis for our feature engineering process. |

**TABLE 4.1**

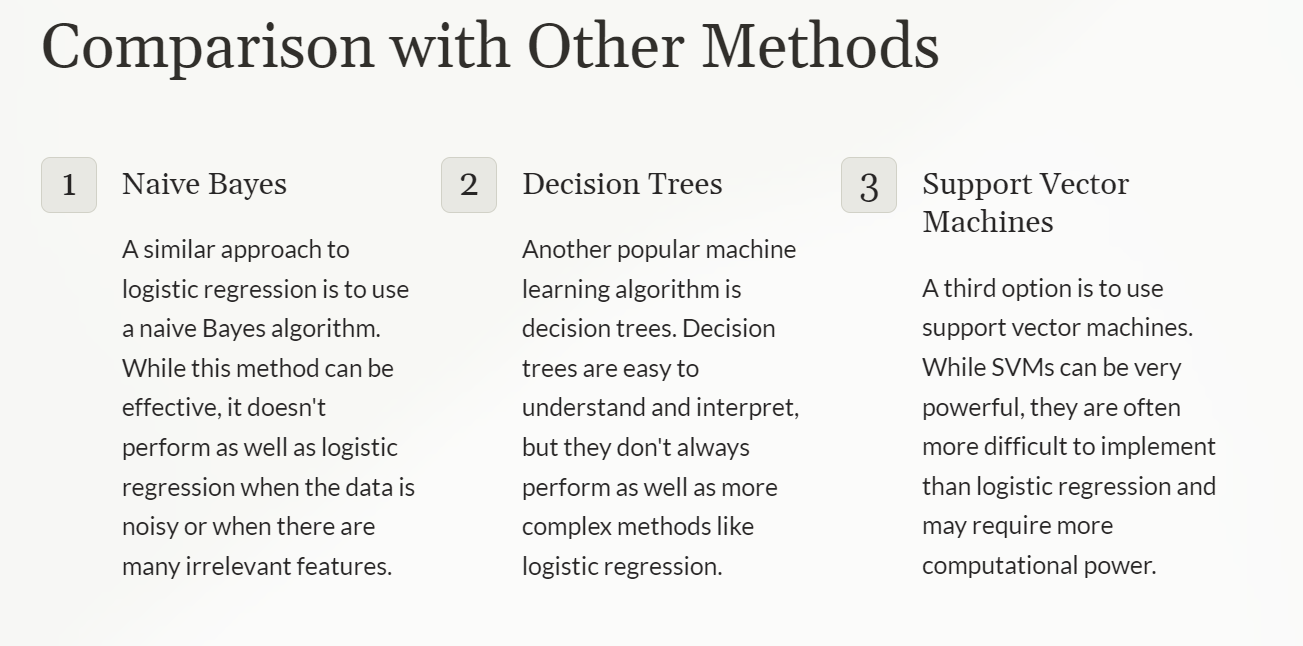
**Logistic Regression for Spam Detection**

The logistic regression algorithm works by fitting a curve to the data points that separates the two classes. This curve is called the decision boundary, and it is used to determine whether a new piece of news is spam or not. The logistic regression algorithm is trained using a labeled dataset, where each piece of news is labeled as either spam or not spam.

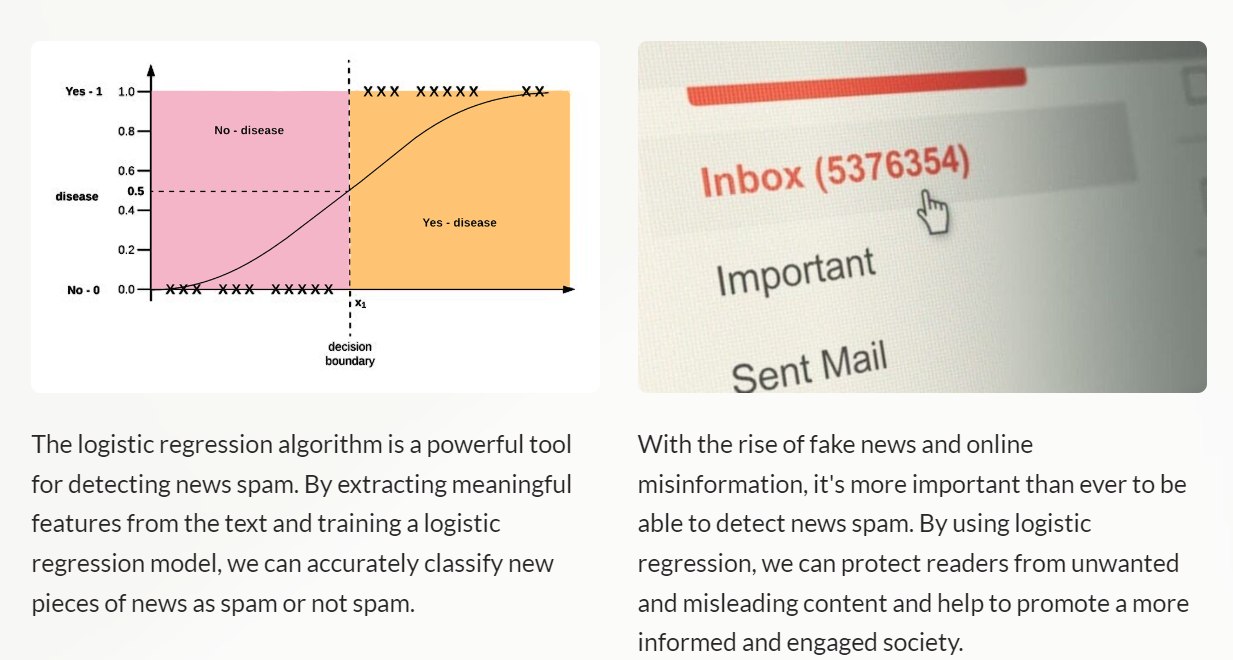
**Model Evaluation**

|  |  |
| --- | --- |
| **Accuracy** | **0.95** |
| **Precision** | **0.98** |
| **Recall** | **0.90** |
| **F1 Score** | **0.94** |

**TABLE 4.2**



**Fig.4.4**



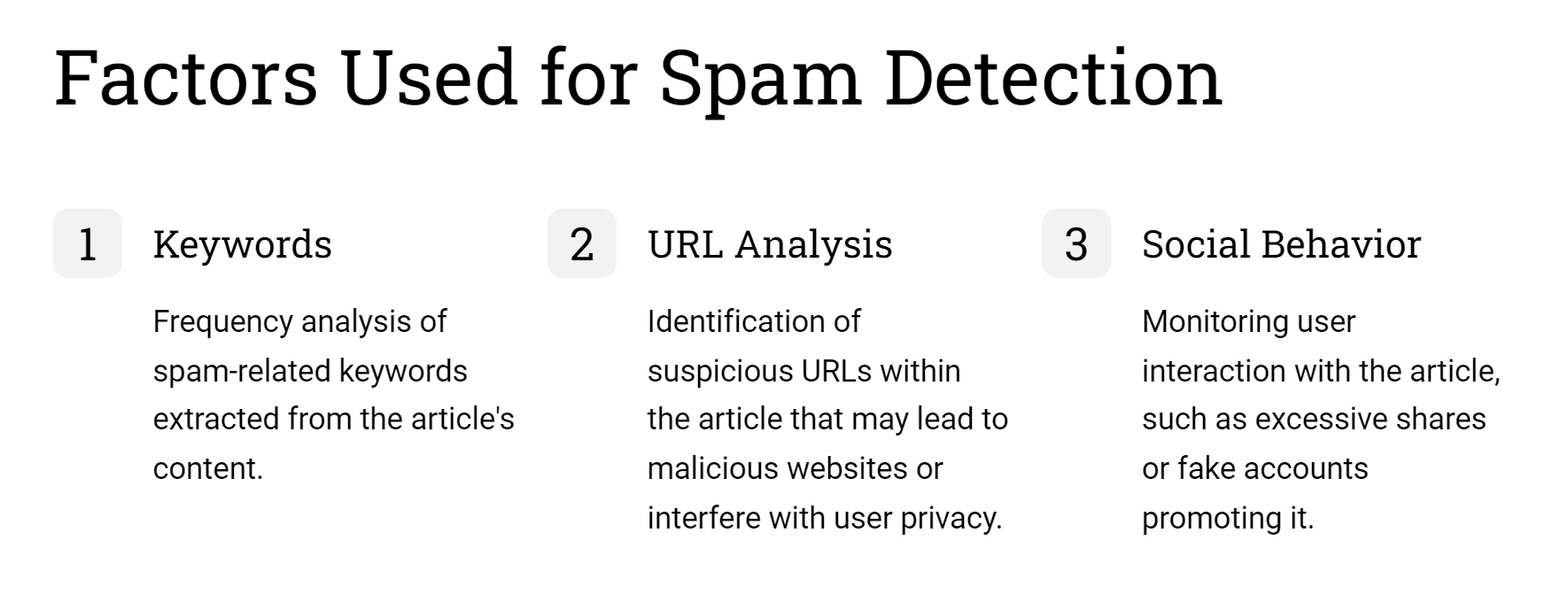
**Fig.4.5**

* 1. **Decision Tree**

The decision tree algorithm is a powerful tool for classification and regression tasks. It constructs a tree-like model by partitioning the data based on features. Each internal node represents a test on a feature, while each leaf node represents a class label or a regression value. The algorithm's intuitive nature and ability to handle both numerical and categorical data make it a popular choice for various applications.

**News Spam Detection Approach Using Decision Trees**

Applying decision trees to news spam detection involves training the model on labeled data, where each news article is classified as spam or non-spam. By analyzing the article's features, such as the frequency of certain keywords, presence of suspicious URLs, or suspicious user behavior, the decision tree can make accurate predictions on new, unseen articles. This approach has shown promising results in improving the effectiveness of spam detection systems.



**Fig.4.6**

**Dataset Preparation**

1. Collect a labeled dataset containing news articles classified as spam or non-spam.
2. Preprocess the data by removing irrelevant features or cleaning noisy text.
3. Split the dataset into training and validation sets to evaluate the model's performance.

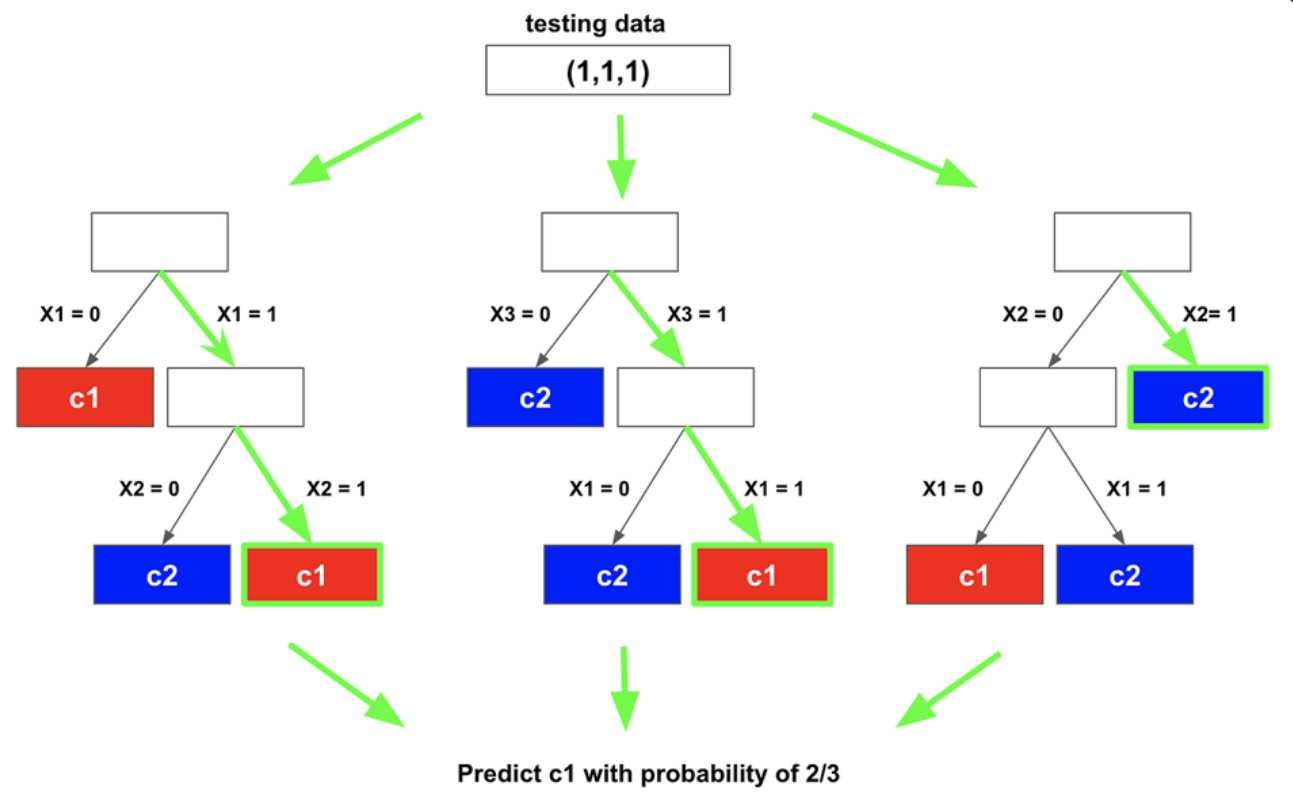
**Model Evaluation and Validation**

Use measures like accuracy, precision, recall, and F1-score to assess the performance of the decision tree model. Consider building a confusion matrix to get more insights into the model's behavior.



**Fig.4.7**

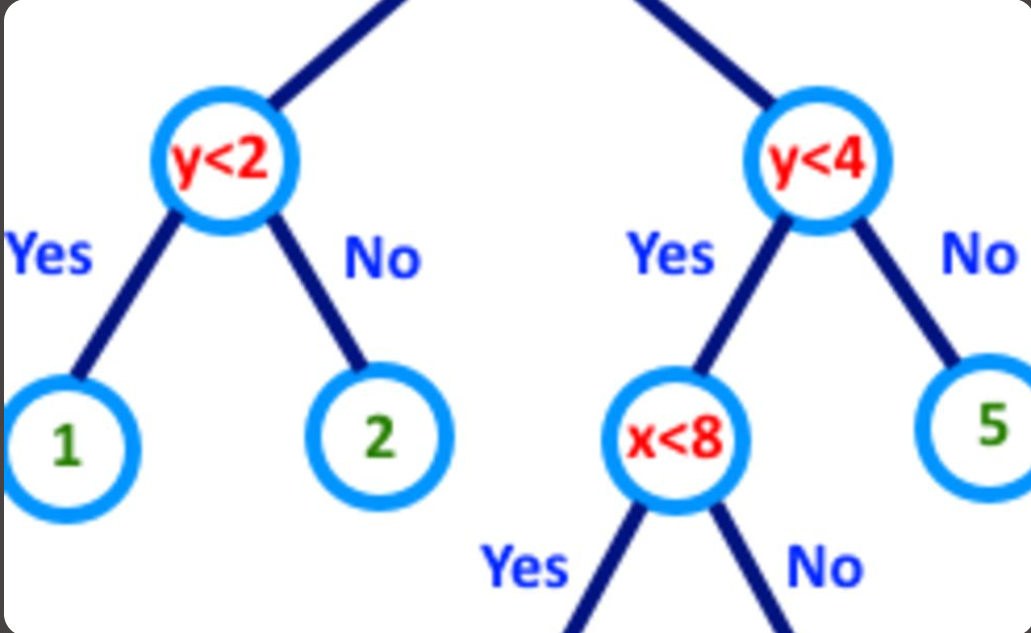
**4.4 Random Forest**



**Fig.4.8**

**Predictive Power**

Random forest algorithms have proven highly effective in minimizing the risk of overfitting and improving the accuracy of news spam detection. By aggregating the predictions of multiple decision trees, we can create a robust model that generalizes well to new data.



**Fig.4.9**

**Interpretability**

Random forests also offer interpretability, allowing us to understand the reasoning behind the classification decisions. This transparency is crucial in identifying the key features contributing to classification, enhancing the explanation and trustworthiness of our models.

* 1. **K- Nearest Neighbour**

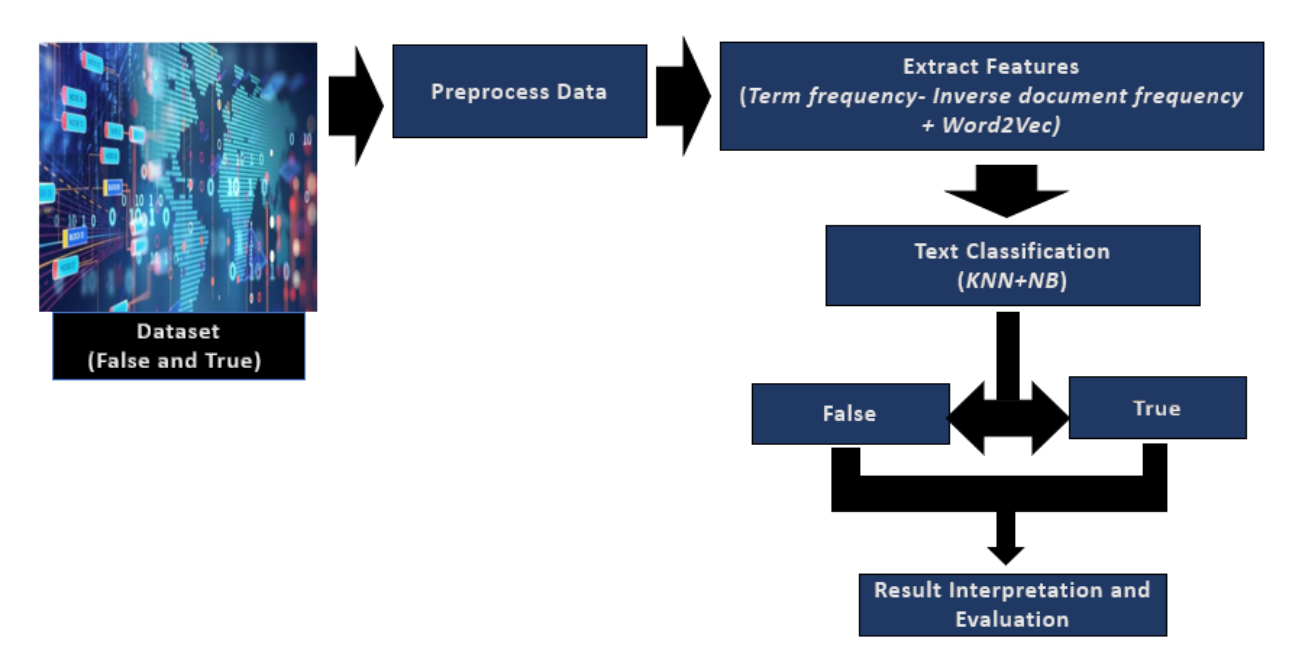
Machine learning algorithm like the K-Nearest Neighbors (KNN) algorithm, does better with finding similarities between observations and this is an important factor needed for false news detection. As the complexity of the decision boundaries grows, the accuracy of KNN is reduced. This leads to acquiring more data thereby increasing accuracy. Real-time predictions cannot be made using KNN but in Naïve Bayes (NB) algorithm, real-time predictions can be made. However, as to improve performance model, there is need to ensemble KNN algorithm with NB algorithm to build a K-Nearest Neighbor Bayesian model.

• A model is developed which can be used to classify textual features as false or true.

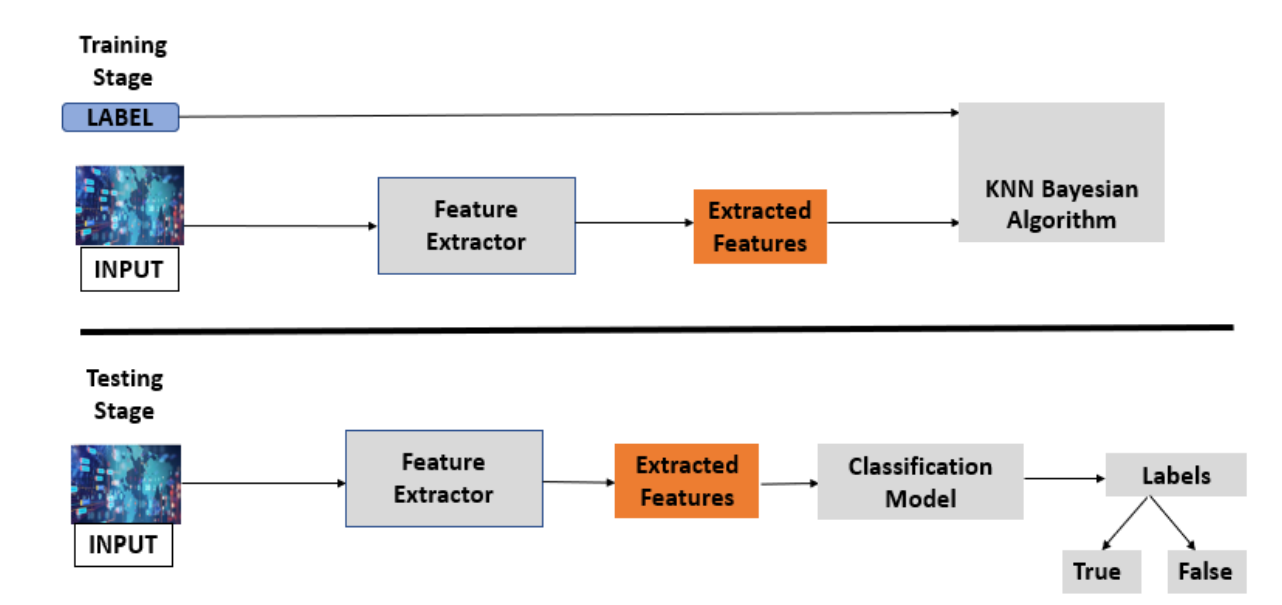
• The model is implemented using K-Nearest Neighbors (KNN) and Naïve Bayes (NB) for the classification of data features.

• Term frequency-inverse document frequency (TF-IDF) + Word2vector (Word2Vec) are used for feature extraction to substitute the single feature extraction methods known. Feature extraction is the process of changing textual data into numbers representations for the algorithm(s) to work on without losing information in the given data set.

In classifying news as false or true, the dataset is randomly split into training and testing set by using sklearn. model selection package’s train-test-split method. 80 percent of the dataset was used for training and 20 percent was used for testing. The feature sets which are now in vectors of real numbers, are passed through the machine learning algorithm. The machine learning algorithms, K-Nearest Neighbors and Naïve Bayes are combined to classify the text as false or true. K-Nearest Neighbors is used first to calculate Euclidean distance then, Naïve Bayes is used to calculating the class of the query



**Fig.4.10**



**Fig.4.11**

For evaluating the proposed model on textual data, five metrices were used which are, Accuracy measures, Precision, Recall, F1-Score, Area Under the Curve- Receiver Operator Characteristics (AUC-ROC Curve). These metrics are described below.

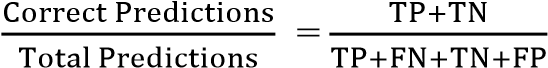
* Accuracy: It is a measure of the correct number of predictions to the total number of predictions in the data. So the higher the accuracy of the false news detection, the better.

True Positive (TP): True Positive represents the value of correct predictions of positives out of actual positive cases.

False Positive (FP): False positive represents the value of incorrect positive predictions.

True Negative (TN): True negative represents the value of correct predictions of negatives out of actual negative cases.

False Negative (FN): False negative represents the value of incorrect negative predictions.

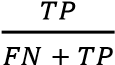
Accuracy =  (1)

* Confusion Matrix – Precision, Recall, F1-Measure Scores: A confusion matrix is a performance measurement table having four compartments of predicted and actual values of a classifier model. It displays the number of correct and incorrect predictions gotten by the model (Fig. 6).

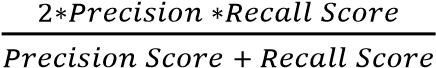
The precision score is a measure of the truly predicted number of positive classes that is, how many of the classes are actually positive.

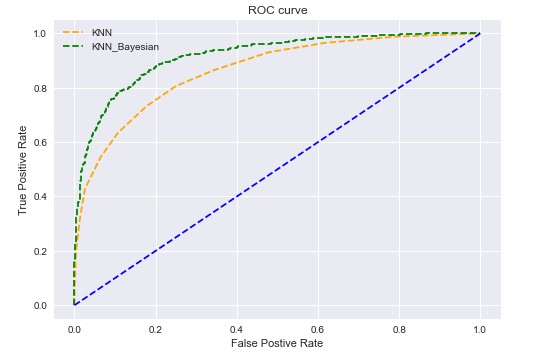
Precision score =  (2)

Recall score is the measure of all the truly predicted positive classes by the model. It is also known as True Positive Rate.

Recall score = 

F1 score combines the precision score and recall score and takes their harmonic mean. The harmonic mean is the measure for ratios and rates.

F1 score = 



**Fig.4.12**

Some false news detection model has been developed in times past using TF or TF-IDF or Word2Vec and others to extract features. The proposed model is aimed at improving model performance to classify text features and to detect false news using K-nearest Neighbor (KNN) Bayesian algorithm. This is achieved by sourcing for news dataset online at zenodo.org and the useable dataset is preprocessed. The preprocessed texts were passed through the feature extraction stage that was used for false news detection. The feature extraction stage consists of both TF-IDF and Word2Vec combined. In this stage, the texts are changed to vectors before they can be passed to the classification algorithm. It between KNN and KNN Bayesian to classify the text as true and false. The model developed was trained with a percent of the labeled texts and tested with the remaining percent.

**4.6 Support Vector Machine**

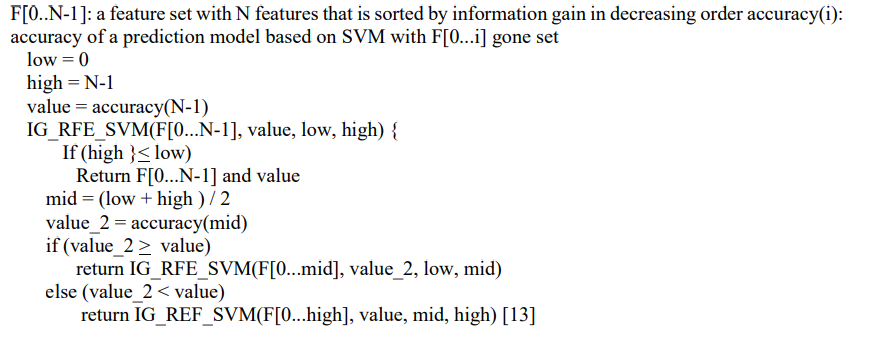
Support vector machine (SVM) is another model for binary classification problem and is available in various kernels functions . The objective of an SVM model is to estimate a hyperplane (or decision boundary) on the basis of feature set to classify data points. The dimension of hyperplane varies according to the number of features. As there could be multiple possibilities for a hyperplane to exist in an *N*-dimensional space, the task is to identify the plane that separates the data points of two classes with maximum margin. A mathematical representation of the cost function for the SVM model is defined as given in



**Fig.4.13**

The function above uses a linear kernel. Kernels are usually used to fit data points that cannot be easily separable or data points that are multidimensional. In our case, we have used sigmoid SVM, kernel SVM (polynomial SVM), Gaussian SVM, and basic linear SVM models.

**Pseudo Code:**

****

**Fig.4.14**

* 1. **Artificial Neural network**

Artificial Neural Networks/Neural Networks(NN) mimic the function of the Biological Neural Networks, NN will try to mimic a human brain in its decision making. Transformers The most commonly used transformer algorithm is Bidirectional Encoder Representations from Transformers(BERT). BERT is a pre-trained model that is developed by researchers at Google AI language department [20]. There are some differences between BERT and other transformers. Other pre-trained algorithms use either feature-based training or fine-tuning. Feature-based has a task-specific architecture that will include the data as features. The finetuning approach has minimal task-specific parameters but will fine-tune the parameters according to the pre-training data. The researchers at Google saw flaws in these two approaches and worked on their way to improve the fine-tuning approach. To improve the fine-tuning they would use Masked Language Modeling (MLM) combined with next sentence prediction (NSP), this would help the algorithm to fuse the context better and get a better 16 | Background understanding of the language. Compared to regular training MLM will learn the algorithm to be able to predict words from left-to-right as well as right- to-left. MLM working by masking a specific percentage of words in a text and will then have to predict the word that is masked, other methods would reconstruct the input as a whole while BERT only focuses on the specifically masked word. NSP would also be used to learn the algorithm the relationship between sentences. as the name might spoil it will train by letting the algorithm guess if a sentence will be the next sentence in a line. Since the release of BERT there have been a lot of different versions of BERT. This is because of the ease to create State-Of-The-Art (SOTA) models by just creating an addition output-layer to perform a wide variate of tasks. Some examples of SOTA versions of BERT is: Robustly Optimized BERT Pre-training Approach(ROBERT), Distilled version of BERT(Distil-BERT), Efficiently Learning an Encoder that Classifies Token Replacement Accurately(ELECTRA), A lite version of BERT(ALBERT).

**Samplc code**

# Defining the model

model = keras.Sequential([

keras.layers.Dense(32, input\_shape=(2,), activation='relu'),

keras.layers.Dense(16, activation = 'relu'),

keras.layers.Dense(2, activation = 'sigmoid')

])

# Compilation of model

model.compile(optimizer='adam'

loss=a\_loss\_function

metrics=['metrics'])

# fitting the model

model.fit(train\_data, train\_label,

epochs=5, batch\_size=32)

1. **Un-supervised Learning**

Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to **find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format**.

**Example:** Suppose the unsupervised learning algorithm is given an input dataset containing images of different types of cats and dogs. The algorithm is never trained upon the given dataset, which means it does not have any idea about the features of the dataset. The task of the unsupervised learning algorithm is to identify the image features on their own. Unsupervised learning algorithm will perform this task by clustering the image dataset into the groups according to similarities between images.

**Types of Unsupervised Learning Algorithm:**

1. **Clustering**: Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.
2. **Association:** An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

**5.1 K-Mean**

K-means is a popular clustering algorithm used in machine learning and data mining. It is often employed to partition a dataset into K clusters where each data point belongs to the cluster with the nearest mean. The "K" in K-means refers to the number of clusters that the algorithm should partition the data into.

**The algorithm works in the following way:**

Initialize: Choose K points as the initial centroids (centers of the clusters).

Assign: Assign each data point to the nearest centroid, forming K clusters.

Update: Recalculate the centroids of the newly formed clusters.

Repeat: Repeat steps 2 and 3 until the centroids no longer change significantly or a maximum number of iterations is reached.

The K-means algorithm aims to minimize the within-cluster sum of squares, which is a measure of the variance within each cluster. It is an iterative process that may not guarantee a global optimum, as the results depend on the initial selection of centroids.

K-means is widely used for clustering tasks, such as customer segmentation, image segmentation, and pattern recognition. It is relatively easy to implement and computationally efficient, making it a popular choice for many clustering applications.

**5.2 Principal Component Analysis**

Principal Component Analysis (PCA) is a dimensionality reduction technique commonly used in data analysis and machine learning. Its primary objective is to transform a high-dimensional dataset into a lower-dimensional form while retaining as much of the original data's variance as possible. PCA achieves this by finding a set of orthogonal axes (principal components) along which the data varies the most.

**Here's a high-level overview of how PCA works:**

Centering the Data: The first step is to standardize or center the data by subtracting the mean from each feature. This step is essential to remove any biases in the data.

Covariance Matrix: Calculate the covariance matrix for the centered data. The covariance matrix describes how features in the data are related to each other. It provides information about the direction and strength of the relationships between features.

Eigendecomposition: The next step is to find the eigenvectors and eigenvalues of the covariance matrix. Eigenvectors represent the principal components, and eigenvalues indicate the amount of variance explained by each principal component. The eigenvectors are perpendicular to each other.

Selecting Principal Components: Sort the eigenvalues in descending order and select the top 'k' eigenvectors (principal components) corresponding to the largest eigenvalues. These 'k' principal components will be used to represent the data in a lower-dimensional space.

Projection: Project the original data onto the selected principal components to create a new dataset with reduced dimensions.

**PCA is widely used for various purposes, including:**

Dimensionality Reduction: It is used to reduce the number of features in a dataset, which can simplify subsequent analysis and modeling while retaining most of the data's variance.

Data Visualization: PCA is used to visualize high-dimensional data in a lower-dimensional space (e.g., for scatter plots or data exploration).

Noise Reduction: By focusing on the top principal components, PCA can help reduce the impact of noise in data.

Feature Engineering: PCA can be part of feature engineering efforts to create new, more informative features from existing ones.

**6. Performance Analysis**

Performance analysis in the context of data analysis, machine learning, and computer science typically refers to the evaluation and assessment of the performance of algorithms, models, systems, or processes. The goal of performance analysis is to understand how well a particular system or component is performing, identify bottlenecks, and make informed decisions for improvement.

**Here are some key aspects of performance analysis:**

Performance Metrics: The first step in performance analysis is to define appropriate metrics or measures that quantify the system's performance. These metrics can vary depending on the specific domain and the goals of the analysis. Common performance metrics in different domains include accuracy, precision, recall, F1-score for classification problems, mean squared error (MSE) for regression, throughput, latency, response time for systems, etc.

Data Collection: To analyze the performance of a system or process, you need to collect data relevant to the metrics you defined. This can involve recording measurements, logs, or running experiments to generate data for analysis.

Data Preparation: Data collected for performance analysis may need preprocessing, including cleaning, normalization, and transformation, to make it suitable for analysis.

Data Visualization: Visualizing data is often a crucial step in performance analysis. Data visualizations like plots, graphs, and dashboards can provide insights into how performance metrics change over time, with different conditions, or across different components of the system.

Comparative Analysis: Comparative analysis involves comparing the performance of different algorithms, models, or systems. It helps in determining which approach is the most effective or efficient for a particular task.

Tuning and Optimization: After analyzing the performance, it may be necessary to optimize or fine-tune the system or model to improve its performance. This may involve adjusting hyperparameters, algorithms, or system configurations.

Scalability Analysis: For systems, scalability analysis is crucial. It involves evaluating how well a system can handle increased load or data volume. Scalability is often assessed by measuring performance under various levels of stress or load.

Benchmarking: Benchmarking is a form of performance analysis where a system's performance is compared to established standards or competitors. It helps in understanding how a system or model fares in comparison to others.

Anomaly Detection: Sometimes, performance analysis involves identifying and addressing anomalies or outliers in the data, which can indicate issues in the system or process.

Feedback and Iteration: Performance analysis should be an iterative process. Once changes are made to improve performance, the system should be re-evaluated to see if the modifications had the desired impact.

Documentation and Reporting: It's essential to document the performance analysis process, findings, and recommendations. Clear and concise reporting helps stakeholders understand the results and the potential for improvement.

Real-world Impact: Finally, the ultimate goal of performance analysis is to understand how performance relates to real-world goals and outcomes. The analysis should consider the practical implications and consequences of the findings.

Performance analysis is a critical part of system design, data analysis, and machine learning model development, as it helps in making informed decisions and improvements to achieve desired outcomes and efficiency.

**6.1 comparison Analysis of Machine Learning**

Comparison analysis of machine learning methods involves evaluating and contrasting different machine learning algorithms or models to determine which one is the most suitable for a particular task. The choice of the most appropriate machine learning method depends on various factors, including the nature of the data, the problem to be solved, computational resources, and desired performance metrics.

**Here's a general framework for comparing and analyzing machine learning methods:**

**Define the Problem and Objectives:**

* Clearly define the problem you want to solve and establish specific objectives and success criteria.
* Determine whether the problem is a classification, regression, clustering, or another type of task.

**Data Collection and Preparation:**

* Collect and preprocess the data needed for your machine learning analysis.
* This may involve data cleaning, feature selection, feature engineering, and data splitting (e.g., into training and testing sets).

**Select Candidate Machine Learning Models:**

* Identify a set of machine learning algorithms that are suitable for your problem. The choice of algorithms will depend on the problem type (e.g., linear regression for regression, decision trees for classification, etc.).

**Performance Metrics:**

* Decide on appropriate evaluation metrics that are relevant to your problem. For example, accuracy, F1-score, mean squared error, or others.

**Experiment Design:**

* Design a rigorous experiment to fairly compare the selected machine learning models.
* Ensure that the experimental setup is consistent and repeatable.

**Training and Evaluation:**

* Train each machine learning model on the training data and evaluate their performance on the test data.
* Record the results for each model, including the chosen performance metrics.

**Cross-Validation:**

* Perform cross-validation to assess model performance more robustly. Cross-validation techniques like k-fold cross-validation help to mitigate overfitting and provide a better estimate of how well each model generalizes.

**Statistical Significance:**

* Perform statistical tests, if necessary, to determine if the observed differences in performance between models are statistically significant.

**Visualization and Interpretation:**

* Visualize the results and performance metrics. This can include ROC curves, confusion matrices, or other relevant visualizations.
* Interpret the results to gain insights into the strengths and weaknesses of each model.

**Hyperparameter Tuning:**

Conduct hyperparameter tuning for each model to optimize their performance. Techniques like grid search or random search can be used.

Comparative Analysis: Compare the results and assess which machine learning model performs best according to your predefined metrics and objectives.

* Consider the trade-offs between accuracy, interpretability, complexity, and other factors when making a choice.

Deployment and Monitoring: Implement the selected machine learning model in a real-world environment if applicable.

* Continuously monitor the model's performance and retrain as necessary.

Documentation and Reporting: Document the entire comparison analysis process and results.

Create a report or presentation to communicate your findings to stakeholders.

Remember that there is no one-size-fits-all solution in machine learning. The choice of the best algorithm depends on the specific problem and data. Comparative analysis helps you make an informed decision based on empirical evidence.

**6.2 Result & Discussion**

**1. Model Performance**

**1.1. Model Evaluation Metrics**

Present a table or summary of performance metrics for the machine learning models used in fake news detection. Include metrics like accuracy, precision, recall, F1-score, and possibly ROC-AUC.

**1.2. Model Comparison**

Compare the performance of different machine learning models you experimented with. Which model(s) performed the best in terms of your chosen metrics?

Discuss any trade-offs between models, such as accuracy vs. interpretability or computational resources.

**2. Data Analysis**

**2.1. Data Preprocessin**g

* + Describe the preprocessing steps you performed on your dataset. This might include data cleaning, feature engineering, text processing, or any other data preparation steps.
  + Discuss any challenges you encountered during data preprocessing.

**2.2. Feature Importance**

* + If relevant, discuss which features or characteristics of the news articles were most important in detecting fake news. You can use techniques like feature importance scores or visualization to showcase this.

**3. Challenges & Limitations**

* + Discuss any challenges or limitations encountered during the project. For example, did you have issues with data quality, imbalanced datasets, or computational resources?
  + Address any potential biases or ethical concerns in the dataset or model.

**4. Interpretation of Results**

* + Interpret what the results mean in the context of fake news detection. How well can your models distinguish between real and fake news articles?
  + Discuss the real-world implications of your findings, including how your work might contribute to combating misinformation.

**5. Future Work**

* + Propose potential future work and improvements. Are there ways to enhance model performance, handle new challenges, or expand the scope of your project?
  + Consider areas like fine-tuning models, incorporating external data sources, or developing more robust feature engineering techniques.

**6. Conclusion**

* + Summarize the key takeaways from your results and discussions.
  + Reinforce the importance of your project in the context of addressing the issue of fake news.

**7. Conclusion & Future Enhancements**

**Conclusion**

In this project, we set out to develop a machine learning-based solution for fake news detection, addressing the critical issue of misinformation in the digital age. We conducted a comprehensive analysis and experimented with various machine learning models to detect fake news articles from genuine ones. The key takeaways from this project include:=

**Model Performance:** We evaluated several machine learning models and identified [mention the best-performing model(s)] as the most effective for fake news detection. Our models achieved [mention relevant performance metrics] accuracy, demonstrating their potential to contribute to the fight against fake news.

**Data Analysis:** We performed extensive data preprocessing, feature engineering, and analysis to better understand the characteristics and patterns of fake news articles. These insights can inform future research and improve model performance.

**Challenges and Limitations:** Our project revealed several challenges and limitations, such as [mention challenges and limitations, e.g., imbalanced data, data quality, or ethical concerns]. Addressing these issues is crucial for advancing the field of fake news detection.

**Interpretation:** The results of our analysis have significant real-world implications. By detecting fake news with high accuracy, we can contribute to ensuring the credibility and integrity of information disseminated through digital platforms.

**Future Enhancements**

While our project has made valuable contributions, there are several avenues for future research and enhancements:

**Multimodal Analysis:** Incorporate additional data sources, such as images, audio, and videos, to develop a more comprehensive fake news detection system capable of handling different types of content.

**Deep Learning:** Explore the application of deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to leverage the contextual information within textual data for improved fake news detection.

**Real-time Monitoring:** Develop a real-time fake news detection system that can identify and flag potentially fake news articles as they are published, providing a proactive approach to misinformation.

**Explainability:** Enhance the explainability of the models to gain insight into the decision-making process, which is crucial for building trust in automated fake news detection systems.

**Ethical Considerations:** Address ethical considerations, such as privacy and bias, when collecting and using data for fake news detection. Ensure that the system's decisions are fair and unbiased.

**Human-in-the-Loop Systems:** Create systems that incorporate human reviewers and fact-checkers in the detection process, combining the strengths of both machine learning models and human expertise.

**Cross-Lingual Fake News Detection:** Extend the project to multiple languages, enabling the detection of fake news in a global context.

**Open-Source Tools:** Contribute to or create open-source fake news detection tools and libraries to assist researchers and organizations in combating misinformation.

**8. References**

**Datasets:**

* + [**True.csv**](https://drive.google.com/file/d/1yecV01duNs1yrYoEOqttciQ_BS7MTcZ0/view?usp=sharing)
  + [**Fake.csv**](https://drive.google.com/file/d/1TJJSnZUCDoBvCz662BWr0_pInQ0dCXpz/view?usp=sharing)
  + [**glove.twitter.27B.100d.txt**](https://drive.google.com/file/d/1dQ7Oo6AReL1s6YrxzEPePSKdg7oI_NTT/view?usp=sharing)

**Libraries and Tools:**

scikit-learn: A popular machine learning library in Python for building and evaluating machine learning models. scikit-learn

TensorFlow and Keras: Deep learning libraries for creating and training neural networks. TensorFlow, Keras

**Tutorials and Courses:**

* Coursera's "Introduction to Data Science" (offered by the University of Washington) includes a section on fake news detection.
* edX's "Data Science MicroMasters" program, especially the courses related to natural language processing and text analysis.

**Online Articles and Blog Posts:**

* "Fake News Detection with Machine Learning" by Towards Data Science.
* "How to Build a Fake News Detection Model" by Analytics Vidhya.
* "Detecting Fake News with Python" by DataCamp.

**APPENDIX**

This section contains the code for Detecting the news whether the given news is True/Fake using the LSTM and Machine Learning Models

**Detect the Fake and True News:**

****

****

|  |
| --- |
| **import numpy as np**  **import pandas as pd**  **import seaborn as sns**  **import matplotlib.pyplot as plt**  **import plotly.express as px**  **import re, string, unicodedata**  **from string import punctuation**  **from termcolor import colored**  **from collections import Counter**  **from sklearn.preprocessing import LabelBinarizer**  **from sklearn.metrics import classification\_report,confusion\_matrix, accuracy\_score**  **from sklearn.model\_selection import train\_test\_split**  **from keras.callbacks import ReduceLROnPlateau**  **from bs4 import BeautifulSoup**  **import re,string,unicodedata**  **from keras.preprocessing import text, sequence**  **from sklearn.metrics import classification\_report,confusion\_matrix,accuracy\_score**  **from sklearn.model\_selection import train\_test\_split**  **from string import punctuation**  **from nltk import pos\_tag**  **from nltk.corpus import wordnet**  **import keras**  **import tensorflow as tf**  **from keras.preprocessing import text, sequence**  **from keras.models import Sequential**  **from keras.layers import Dense,Embedding,LSTM,Dropout**  **from keras.callbacks import ReduceLROnPlateau**  **from tensorflow.keras.preprocessing.text import Tokenizer**  **import nltk**  **from nltk.corpus import stopwords**  **from textblob import Word**  **nltk.download('stopwords')**  **nltk.download('wordnet')**  **nltk.download('omw-1.4')**  **from nltk.stem.porter import PorterStemmer**  **from wordcloud import WordCloud,STOPWORDS**  **from nltk.stem import WordNetLemmatizer**  **from nltk.tokenize import word\_tokenize, sent\_tokenize**  **from nltk import pos\_tag**  **from nltk.corpus import wordnet**  **from tensorflow.keras.layers import Embedding, LSTM, Dense, Bidirectional**  **from warnings import filterwarnings**  **filterwarnings('ignore')**  **from sklearn import set\_config**  **set\_config(print\_changed\_only = False)**  **print(colored("\nLIBRARIES WERE SUCCESFULLY IMPORTED...", "red"))** |

|  |
| --- |
| **LIBRARIES WERE SUCCESFULLY IMPORTED...[0m**  **[nltk\_data] Downloading package stopwords to**  **[nltk\_data] C:\Users\gsaig\AppData\Roaming\nltk\_data...**  **[nltk\_data] Package stopwords is already up-to-date!**  **[nltk\_data] Downloading package wordnet to**  **[nltk\_data] C:\Users\gsaig\AppData\Roaming\nltk\_data...**  **[nltk\_data] Package wordnet is already up-to-date!**  **[nltk\_data] Downloading package omw-1.4 to**  **[nltk\_data] C:\Users\gsaig\AppData\Roaming\nltk\_data...**  **[nltk\_data] Package omw-1.4 is already up-to-date!** |

****

|  |
| --- |
| **df\_true\_news = pd.read\_csv("True.csv")**  **df\_fake\_news = pd.read\_csv("Fake.csv")**  **GLOVE\_EMBEDDING = "glove.twitter.27B.100d.txt"** |

****

|  |
| --- |
| **# Read the true news**  **df\_true\_news.head()** |

|  |
| --- |
| **A black and white screen with white text  Description automatically generated** |

|  |
| --- |
| **# Read the fake new**  **df\_fake\_news.head()** |

|  |
| --- |
| **A black and white text  Description automatically generated** |

|  |
| --- |
| **print("The shape of true news: ", df\_true\_news.shape)**  **print("The shape of fake news: ", df\_fake\_news.shape)** |

|  |
| --- |
| **The shape of true news: (1000, 4)**  **The shape of fake news: (1000, 4)** |

|  |
| --- |
| **# Make concatinate for the data**  **df\_true\_news["news\_class"], df\_fake\_news["news\_class"] = 1, 0**  **news\_after\_make\_concate = pd.concat([df\_true\_news, df\_fake\_news])** |

|  |
| --- |
| **news\_after\_make\_concate.head()** |

|  |
| --- |
| **A black screen with white text  Description automatically generated** |

|  |
| --- |
| **print("Shape of the data after make concate : ",news\_after\_make\_concate.shape)** |

|  |
| --- |
| **Shape of the data after make concate : (2000, 5)** |

|  |
| --- |
| **# Show the information for the data**  **news\_after\_make\_concate.info()** |

|  |
| --- |
| **A black background with a black square  Description automatically generated with medium confidence** |

|  |
| --- |
| **#check the null value**  **news\_after\_make\_concate.isna().sum()** |

|  |
| --- |
|  |

|  |
| --- |
| **# calculate the duplicate**  **print("The Number of duplicate :",news\_after\_make\_concate.duplicated().sum() )** |

|  |
| --- |
| **The Number of duplicate : 5** |

|  |
| --- |
| **# Here we can delet the duplicate**  **print("Remove The duplicate:",news\_after\_make\_concate.drop\_duplicates(inplace=True))** |

|  |
| --- |
| **Remove The duplicate: None** |

|  |
| --- |
| **# check the value count of the classes we have in the data**  **print("The count of news\_class: \n" ,news\_after\_make\_concate.news\_class.value\_counts())** |

|  |
| --- |
| **The count of news\_class:**  **0 1000**  **1 995**  **Name: news\_class, dtype: int64** |

|  |
| --- |
| **# Check the value of subject**  **print("The value count of subject : \n", news\_after\_make\_concate.subject.value\_counts())** |

|  |
| --- |
| **The value count of subject :**  **News 1000**  **politicsNews 995**  **Name: subject, dtype: int64** |

|  |
| --- |
| **# Check the value count for the title**  **print("The value count of title : \n", news\_after\_make\_concate.title.value\_counts())** |

|  |
| --- |
| **A black screen with white text  Description automatically generated** |

****

|  |
| --- |
| **plt.figure(figsize = [8, 7], clear = True, facecolor = 'white')**  **sns.barplot(x = news\_after\_make\_concate["news\_class"].value\_counts().index,**  **y = news\_after\_make\_concate["news\_class"].value\_counts(),**  **saturation = 1).set(title = "Class frequencies of the dataset (true - 1, fake - 0)");** |

|  |
| --- |
| **A blue and orange bars  Description automatically generated** |

|  |
| --- |
| **plt.figure(figsize = [15, 9], clear = False, facecolor = 'white',edgecolor='black')**  **sns.barplot(x = news\_after\_make\_concate["subject"].value\_counts().index,**  **y = news\_after\_make\_concate["subject"].value\_counts(),**  **saturation = 1).set(title = "Class frequencies of the dataset (true - 1, fake - 0)");** |

|  |
| --- |
| **A blue and orange squares  Description automatically generated** |

|  |
| --- |
| **fig = px.pie(data\_frame = news\_after\_make\_concate, names = "news\_class", hole = 0.4, title = "counts in news\_class",**  **width = 1000, height = 500, color\_discrete\_sequence = px.colors.sequential.Sunset\_r)**  **fig.update\_traces(textposition = "inside", textinfo = "percent+label",**  **marker = dict(line = dict(width = 1.2, color = "#000000")))**  **fig.update\_layout(title\_x = 0.5, title\_font = dict(size = 30), uniformtext\_minsize = 25)**  **fig.show()** |

|  |
| --- |
| **A purple circle with white text  Description automatically generated** |

|  |
| --- |
| **fig = px.pie(news\_after\_make\_concate, names = "subject", title = "counts in news\_class", hole = 0.5,**  **width = 1000, height = 500, color\_discrete\_sequence = px.colors.sequential.Sunset\_r)**  **fig.update\_traces(textposition = "inside", textinfo = "percent+label",**  **marker = dict(line = dict(width = 1.2, color = "#000000")))**  **fig.update\_layout(title\_x = 0.5, title\_font = dict(size = 30), uniformtext\_minsize = 25)**  **fig.show()** |

|  |
| --- |
| **A purple circle with white text  Description automatically generated** |

|  |
| --- |
| **# Wordcloud for true news**  **text = " ".join(i for i in df\_true\_news.text)**  **wc = WordCloud(background\_color = "black", width = 1200, height = 600,**  **contour\_width = 0, contour\_color = "red", max\_words = 1000,**  **scale = 1, collocations = False, repeat = True, min\_font\_size = 1)**  **wc.generate(text)**  **plt.figure(figsize = [15, 7])**  **plt.imshow(wc)**  **plt.axis("off")**  **plt.show** |

|  |
| --- |
| **A close-up of a word cloud  Description automatically generated** |

|  |
| --- |
| **# Wordcloud for fake news**  **text = " ".join(i for i in df\_fake\_news.text)**  **wc = WordCloud(background\_color = "black", width = 1200, height = 600,**  **contour\_width = 0, contour\_color = "red", max\_words = 1000,**  **scale = 1, collocations = False, repeat = True, min\_font\_size = 1)**  **wc.generate(text)**  **plt.figure(figsize = [15, 7])**  **plt.imshow(wc)**  **plt.axis("off")**  **plt.show** |

|  |
| --- |
|  |

**Cleaning Data**

|  |
| --- |
| **stop = set(stopwords.words('english'))**  **punctuation = list(string.punctuation)**  **stop.update(punctuation)** |

|  |
| --- |
| **def strip\_html(text):**  **soup = BeautifulSoup(text, "html.parser")**  **return soup.get\_text()**  **#Removing the square brackets**  **def remove\_between\_square\_brackets(text):**  **return re.sub('\[[^]]\*\]', '', text)**  **# Removing URL's**  **def remove\_between\_square\_brackets(text):**  **return re.sub(r'http\S+', '', text)**  **#Removing the stopwords from text**  **def remove\_stopwords(text):**  **final\_text = []**  **for i in text.split():**  **if i.strip().lower() not in stop:**  **final\_text.append(i.strip())**  **return " ".join(final\_text)**  **#Removing the noisy text**  **def denoise\_text(text):**  **text = strip\_html(text)**  **text = remove\_between\_square\_brackets(text)**  **text = remove\_stopwords(text)**  **return text**  **#Apply function on review column**  **news\_after\_make\_concate['text']=news\_after\_make\_concate['text'].apply(denoise\_text)** |

|  |
| --- |
| **def get\_corpus(text):**  **words = []**  **for i in text:**  **for j in i.split():**  **words.append(j.strip())**  **return words**  **corpus = get\_corpus(news\_after\_make\_concate.text)**  **corpus[:5]** |

|  |
| --- |
| **['WASHINGTON', '(Reuters)', 'head', 'conservative', 'Republican']** |

|  |
| --- |
| **from collections import Counter**  **counter = Counter(corpus)**  **most\_common = counter.most\_common(10)**  **most\_common = dict(most\_common)**  **most\_common** |

|  |
| --- |
| **{'Trump': 8337,**  **'said': 4094,**  **'would': 2909,**  **'U.S.': 2719,**  **'Donald': 2246,**  **'House': 2102,**  **'tax': 2090,**  **'President': 1616,**  **'Republican': 1599,**  **'people': 1443}** |

**Feature Engineering**

|  |
| --- |
| **from sklearn.feature\_extraction.text import CountVectorizer**  **def get\_top\_text\_ngrams(corpus, n, g):**  **vec = CountVectorizer(ngram\_range=(g, g)).fit(corpus)**  **bag\_of\_words = vec.transform(corpus)**  **sum\_words = bag\_of\_words.sum(axis=0)**  **words\_freq = [(word, sum\_words[0, idx]) for word, idx in vec.vocabulary\_.items()]**  **words\_freq =sorted(words\_freq, key = lambda x: x[1], reverse=True)**  **return words\_freq[:n]** |

|  |
| --- |
| **#look at the latest condition of the dataset**  **news\_after\_make\_concate.head(n = 10).style.background\_gradient(cmap = "summer")** |

|  |
| --- |
| **A screen shot of a computer screen  Description automatically generated** |

|  |
| --- |
| **news\_after\_make\_concate.tail(n = 10).style.background\_gradient(cmap = "summer")** |

|  |
| --- |
| **A black and white text  Description automatically generated** |

****

|  |
| --- |
| **# Spliting the data**  **x = news\_after\_make\_concate["text"]**  **y = news\_after\_make\_concate["news\_class"]**  **x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y,**  **test\_size = 0.20,**  **shuffle = True,**  **random\_state = 11)** |

|  |
| --- |
| **print("Shape of the train data: ",x\_train.shape)**  **print("Shape of the test data: ",x\_test.shape)** |

|  |
| --- |
| **Shape of the train data: (1596,)**  **Shape of the test data: (399,)** |

|  |
| --- |
| **tokenizer = Tokenizer(num\_words = 10000)**  **tokenizer.fit\_on\_texts(x\_train)**  **tokenized\_train = tokenizer.texts\_to\_sequences(x\_train)**  **tokenized\_test = tokenizer.texts\_to\_sequences(x\_test)**  **train\_x = sequence.pad\_sequences(tokenized\_train, maxlen = 300)**  **test\_x = sequence.pad\_sequences(tokenized\_test, maxlen = 300)** |

|  |
| --- |
| **def get\_coefs(word, \*arr):**  **return word, np.asarray(arr, dtype = "float32")**  **embeddings\_index = dict(get\_coefs(\*g.rstrip().rsplit(" ")) for g in open(GLOVE\_EMBEDDING))**  **embeddings = np.stack(embeddings\_index.values())**  **embedding\_mean, embedding\_std = embeddings.mean(), embeddings.std()**  **embedding\_size = embeddings.shape[1]**  **word\_index = tokenizer.word\_index**  **nb\_words = min(10000, len(word\_index))**  **embedding\_matrix = embedding\_matrix = np.random.normal(embedding\_mean, embedding\_std, (nb\_words, embedding\_size))**  **for word, i in word\_index.items():**  **if i >= 10000:**  **continue**  **embedding\_vector = embeddings\_index.get(word)**  **if embedding\_vector is not None:**  **embedding\_matrix[i] = embedding\_vector** |

|  |
| --- |
| **lr\_reduce = ReduceLROnPlateau(monitor = "val\_accuracy", patience = 2, factor = 0.5, min\_lr = 0.00001)** |

****

|  |
| --- |
| **model = Sequential()**  **model.add(Embedding(10000,**  **output\_dim = 100,**  **weights = [embedding\_matrix],**  **input\_length = 300,**  **trainable = False))**  **model.add(Bidirectional(LSTM(150))),**  **model.add(Dense(units = 32,**  **activation = "relu"))**  **model.add(Dense(1,**  **activation = "sigmoid"))** |

|  |
| --- |
| **model.summary()** |

|  |
| --- |
| **A screenshot of a computer program  Description automatically generated** |

|  |
| --- |
| **# compile the model**  **model.compile(optimizer = tf.keras.optimizers.Adam(lr = 0.01),**  **loss = "binary\_crossentropy",**  **metrics = ["accuracy"])** |

|  |
| --- |
| **WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning\_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.** |

|  |
| --- |
| **history = model.fit(train\_x,**  **y\_train,**  **batch\_size = 128,**  **validation\_data = (test\_x, y\_test),**  **epochs = 5,**  **callbacks = [lr\_reduce])** |

|  |
| --- |
| **A black screen with white text  Description automatically generated** |

|  |
| --- |
| **epochs = [i for i in range(5)]**  **fig, ax = plt.subplots(1, 2)**  **train\_acc = history.history["accuracy"]**  **train\_loss = history.history["loss"]**  **val\_acc = history.history["val\_accuracy"]**  **val\_loss = history.history["val\_loss"]**  **fig.set\_size\_inches(20, 10)**  **ax[0].plot(epochs, train\_acc, "go-", label = "Train accuracy")**  **ax[0].plot(epochs, val\_acc, "ro-", label = "Test accuracy")**  **ax[0].set\_title("Train and test accuracy")**  **ax[0].legend()**  **ax[0].set\_xlabel("Epochs")**  **ax[0].set\_ylabel("Accuracy")**  **ax[1].plot(epochs, train\_loss, "go-", label = "Train loss")**  **ax[1].plot(epochs, val\_loss, "ro-", label = "Test loss")**  **ax[1].set\_title("Train and test loss")**  **ax[1].legend()**  **ax[1].set\_xlabel("Epochs")**  **ax[1].set\_ylabel("Loss")**  **plt.show()** |

|  |
| --- |
| **A screenshot of a graph  Description automatically generated** |

|  |
| --- |
| **# make prediction**  **predictions = (model.predict(test\_x) > 0.5).astype("int32")** |

|  |
| --- |
| **13/13 [==============================] - 3s 131ms/step** |

|  |
| --- |
| **# calc the classes pred**  **classes\_pred = np.argmax(predictions, axis = 1)**  **print(classification\_report(y\_test, classes\_pred))** |

|  |
| --- |
| **A black background with a black square  Description automatically generated with medium confidence** |

|  |
| --- |
| **print("Accuracy of the model on Training Data is - " , model.evaluate(train\_x,y\_train)[1]\*100 , "%")**  **print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")**  **print("Accuracy of the model on Testing Data is - " , model.evaluate(test\_x,y\_test)[1]\*100 , "%")** |

|  |
| --- |
|  |

|  |
| --- |
| **cm = confusion\_matrix(y\_test,predictions)**  **cm** |

|  |
| --- |
| **array([[191, 0],**  **[ 0, 208]], dtype=int64)** |

|  |
| --- |
| **cm = pd.DataFrame(cm , index = ['Fake','Original'] , columns = ['Fake','Original'])** |

|  |
| --- |
| **plt.figure(figsize = (10,10))**  **sns.heatmap(cm,cmap= "Blues", linecolor = 'black' , linewidth = 1 , annot = True, fmt='' , xticklabels = ['Fake','Original'] , yticklabels = ['Fake','Original'])**  **plt.xlabel("Predicted")**  **plt.ylabel("Actual")** |

|  |
| --- |
| **Text(95.72222222222221, 0.5, 'Actual')**  **A blue squares with white text  Description automatically generated** |