# Exploring the Use of Machine Learning and Natural Language Processing Techniques for Fake News Detection on Reddit

Ashritha Kumari , Dhrumil Shah, Prudhvi Ch, Shashank Reddy

# Index

# Index

# Abstract

# Abstract

1. Issue Recognition and Significance - Acknowledges the pervasive and disruptive nature of fake news in the digital age, emphasizing its potential fatal consequences for individuals and society.

2. Project Objective - Aims to conduct a comprehensive comparative analysis of machine learning algorithms for detecting fake news, involving data collection, preprocessing, and training/testing models on Reddit posts with features like retweets, image_text, and title.

3. Algorithmic Approach - Utilizes NLP for sentiment analysis, employs spacy for feature embedding, and trains the model using a diverse set of classification algorithms such as, Random Forests, Naïve Bayes, Decision Tree, SVM, Logistic Regression..

4. Evaluation Metrics and Model Efficiency - Assesses model efficiency using various evaluation metrics, including confusion matrix, Accuracy, F1-score,  Recall, Precision with the goal of determining the best-performing model for classifying posts.

5. Application and Future Scope  - Envisions practical application in filtering out fake news on Reddit, emphasizing the project's potential impact in preventing misinformation and suggesting future implementation for flagging misleading accounts.

# 1. Introduction

# 1.1 Project Background

1. Reddit Significance: With over 430 million monthly users and 100,000 active communities, Reddit is a globally influential social network, generating 830,000 daily posts. The prevalence of fake news on Reddit poses a significant societal risk, especially in political contexts.

2. Project Need and Importance: The project addresses the critical need to combat the escalating generation of fake news, emphasizing the potential severe consequences on a national and global scale. Unchecked, fake news can incite discord, disrupt economies, damage reputations, and erode public trust in government.

3. Proposed Model Flow: The project outlines a comprehensive model flow, encompassing data collection through Reddit's "PRAW" API, preprocessing involving natural language processing and feature embedding, and model training/testing/evaluation. Features include post-related data such as title, author, shares, likes, subreddits, and upvote ratio.

4. Diverse Classification Techniques: The model incorporates various classification techniques, including Random Forests, Naive Bayes, Decision Trees, SVM, Logistic Regression. Natural language processing and spacy embeddings enhance feature extraction.

5. Evaluation Metrics: Multiple assessment metrics, including confusion matrix, accuracy, F1-score, Recall, and Precision, will be utilized to gauge model effectiveness. The selection of the best-performing model for identifying fake Reddit posts will be based on a comprehensive comparison of these evaluation metrics.

# 1.2 Project Requirements

**Table 1**

*Table showing the variable names and description.*

| Column Name | Column Description |
|---|---|
| title | Title of the post |
| upvote_ratio | The ratio of likes to total likes and dislikes |
| num_comments | Number of comments on the post |
| url | URL provided in the post |

- Data Collection Using PRAW - Utilizes the Python Reddit API Wrapper (PRAW) to gather source and metadata information of user posts from various news-based subreddits such as "news," "Politics," "WorldNews," “nottheonion,” and "fakenews" through the Reddit API.
- Dataset Characteristics - The collected data is saved in a comma-separated file with approximately 70,000 rows and 10 columns. The chosen columns for retention include "title," "upvote_ratio," "num_comments," "url," while the remaining columns are discarded.
- Focus on Relevant Information - Emphasizes the importance of retaining specific columns such as post title, upvote ratio, image_text, self_text, and URL, streamlining the dataset to focus on key information relevant to the analysis or project goals.

# 1.2 Project Requirements (continued)

- Labeling. The collected data from Reddit are unlabeled. The collected data must be manually identified and labeled as fake or real posts. The model's prediction capabilities can be impacted by incorrect label.
- Quality of posts. Data quality should be maintained for a good-performing model. The textual data gathered from user posts on Reddit must be contextually rich. There should be no null values or unclean text in the posts, duplicates.
- Variable selection. We must ensure the required features or variables are extracted from the user posts. Needless features must be discarded from the dataset to ensure there is a clean dataset for implementation.
- Variety of posts. An extensive range of posts including those with varying degrees of engagement must be evaluated and classified effectively by the model.

# 1.3 Project Deliverables

The project performs a comparative analysis of different classification algorithms for the data at hand, we will be using agile methodology to perform project management, where the deliverables will be submitted for every two weeks, a breakdown of task assignment will also be provided. The project deliverables for this project includes the following, a careful review of the available literature on false news detection, a thorough system requirements specifications report, Documents that describe the system architecture, a report on the execution of data collection, python notebooks consisting of the model building for different machine learning algorithms and their evaluation, final project report comparing the results of the various classification algorithms. A Gantt chart and a PERT chart related to this project will also be submitted which will have the planned actions/tasks and the timeline to achieve the tasks. This will help us streamline the project.

# 1.4 Technology Survey

NLP tools like spacy are used for the preprocessing of the data. Machine learning algorithms described below will be implemented to build efficient models and validate using metrics such as confusion matrix, accuracy, F1-score, recall and precision.

1. Logistic Regression. It is a binary or multiclass classification algorithm. It considers the dataset independent of variables and predicts the dependent variable using probability.
2. Decision tree. It is a non-parametric supervised learning algorithm used for classification and regression tasks. It has a hierarchical or tree-like structure that consists of if-else conditions based on variables.
3. Random Forests. This is a commonly used machine learning algorithm that combines the output of multiple decision trees to produce an efficient predictive model. It is a supervised algorithm applied to both classification and regression problems.
4. Naïve Bayes. It is a supervised machine learning algorithm commonly used for text classification. Classifies based on Bayes' theorem. It is a generative classifier model and follows the first principle approach.
5. SVM. Support Vector Machine used for classification, regression, and outlier detection. However, it is widely used in classification tasks with the objective of classifying the data points through an n-dimensional hyperplane.

# 1.4 Technology Survey (continued)

A python script using PRAW will implement the data collection part by scraping 100 hot posts every 6 hours each day and this will be implemented through CRON jobs on an EC2 instance which will satisfy the data requirement needs.

To implement the proposed model, Google Collaboratory or Jupyter Notebook will be used. The Python libraries that are helpful for our project are scikit-learn and PyTorch.

The above models will all be constructed, evaluated, and compared in order to determine which model performs the best and provides the best results among all the models mentioned above

# 1.5 Literature Review

**Fake news detection using machine learning approaches on political news extracted from Twitter**

- Dataset: Political news extracted from Twitter, preprocessed and noise removed using NLP libraries
- Methodology: Feature extraction using word count, length, unigram and bigram features using TF-IDF n-gram features
- Results: XGBoost was found to produce the best accuracy with more than 75%, followed by SVM and Random Forest with accuracy nearly 73%

**Fake news detection on social media using machine learning techniques**

- Data: Social media news articles in text, image, and video formats
- Methodology: Deep learning discriminative classifiers (MLP, CNN, RNN), generative models (GAN, DBN), hybrid models (LSTM, ensemble-based functions), handcrafted feature extraction, and ML classifiers (SVM, Naive Bayes, KNN, Decision Trees, Linear Regression, Ensemble Classifiers, Non-linear Regressions, K-means, Gaussian, Hidden Markov Model, Neural Networks)
- Results: The paper reviews different aspects that affect the performance of machine learning techniques for fake news detection and provides a bibliometric analysis of the field, SVM performed best with Accuracy of 87% for them.

# 1.5 Literature Review

**Survey of fake news detection using machine intelligence approach**

- Dataset used in this paper consists of articles collected from the internet
- Each datapoint was tokenized, and feature extraction was performed by eliminating stop words like for, it, do, oneself, etc.
- Various machine learning algorithms like Passive Aggressive Classifier, Naïve Bayes, Logistic Regression, Decision Tree, LSTM and BERT were used
- According to their experimental results, there were algorithms that outperformed in each evaluation metric; however, it was concluded that, on average, passive aggressive classifier presented the best results.

**Fake Information Analysis and Detection on Pandemic in Twitter**

- Real-time tweets related to the pandemic were scraped from Twitter. A total of 1517 tweets were collected
- Features selected were user_followers, upon user_favorites, hashtags used in the tweets, and the polarity of the tweets
- The models applied were logistic regression, SVM, random forest, decision tree, RNN, and LSTM
- The Random Forest classifier returned the highest accuracy of 85.2% and an F1-score of 0.849

# 2. Data and Project Management

# 2.1 Data Management Plan

Data Collection

- The Python Reddit API Wrapper (PRAW) is used to collect data from six specific subreddits: news, whitepeopletwitter, politics, worldnews, nottheonion and fakenews
- The script runs every six hours, gathering 100 posts from each subreddit, resulting in 2400 new data points per day. By the end of the project, we will have collected approximately 124,800 data points

Data Storage

- The files are initially stored on EBS volumes and later transferred to S3 buckets, with only the administrator having edit access to ensure data integrity. Then the data is download from S3 buckets as CSV files

Data Pre-Processing

- It is a critical step in data science, transforming raw data into a usable format for analysis and modelling.
- It involves identifying and removing duplicates, unnecessary columns. These steps ensure data quality and consistency, leading to more accurate and reliable results.

# 2.1 Data Management Plan (continued)

The storage format used to store the data is CSV, a sample of the data collected from subreddit news on 10/22/2023 at 00:00:03 and the screenshot of the data is provided below with the explanation for all the variables collected.

20231022_000003news

| | selftext | s | author | subreddit | created_utc | upvote | num_cor | permalink | url |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Detroit synagogue president | | Thai-Reidj | news | 1697915627.0 | 0.94 | 1883 | /r/news/comments/17da1fl/detroit_synagogu | https://www.free |
| 1 | US budget gap soars to $1. | | ChocolateTsa | news | 1697911308.0 | 0.92 | 368 | /r/news/comments/17d8g9z/us_budget_gap_ | https://www.reu |
| 2 | Florida man sentenced to 1 | | davetowers64 | news | 1697914209.0 | 0.95 | 106 | /r/news/comments/17d9icw/florida_man_sent | https://apnews.c |
| 3 | Juveniles charged with dous | | Drumlin | news | 1697878703.0 | 0.96 | 689 | /r/news/comments/17cy9z9/juveniles_charge | https://apnews.c |
| 4 | Deputies find 5-year-old twi | | NickDanger3c | news | 1697893325.0 | 0.95 | 249 | /r/news/comments/17d22o9/deputies_find_5y | https://apnews.c |
| 5 | Antisemitic hate crimes in L | | Ok-Fox5767 | news | 1697924583.0 | 0.86 | 173 | /r/news/comments/17dd9k2/antisemitic_hate | https://www.the |
| 6 | Questions linger after Conne | | FreeChickenD | news | 1697895389.0 | 0.96 | 271 | /r/news/comments/17d2qqx/questions_linger | https://apnews.c |
| 7 | Czech village priest sorry fo | | nosotros_roac | news | 1697894331.0 | 0.93 | 313 | /r/news/comments/17d2ee3/czech_village_pr | https://www.bbc |
| 8 | US puts sanctions on three | | luckygrape64 | news | 1697889086.0 | 0.95 | 44 | /r/news/comments/17d0tks/us_puts_sanctior | https://www.reu |
| 9 | A masked police officer robl | | Daxl | news | 1697840547.0 | 0.9 | 509 | /r/news/comments/17cn7wm/a_masked_polic | https://www.cnr |

Figure: Showing all the variables

# 2.2 Project Development Methodology

CRISP-DM methodology is used for data mining projects, comprising six key phases

1. Business Understanding involves translating business goals into data mining problems
2. Data Understanding focuses on exploring and evaluating data quality and characteristics
3. Data Preparation ensures data is clean and suitable for modelling
4. Modelling includes selecting and building data mining models
5. Evaluation assesses model performance
6. Deployment integrates it into the business environment

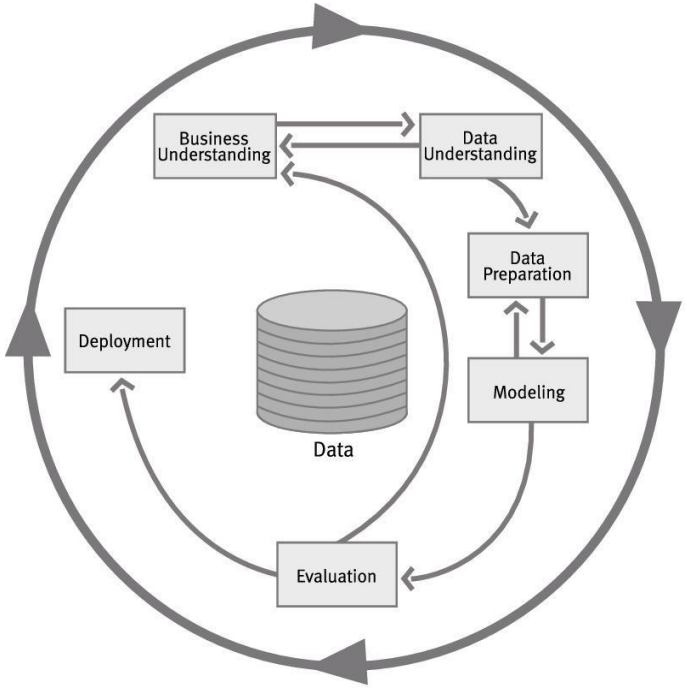



Figure: CRISP-DM methods

# 2.3 Project Organization Plan

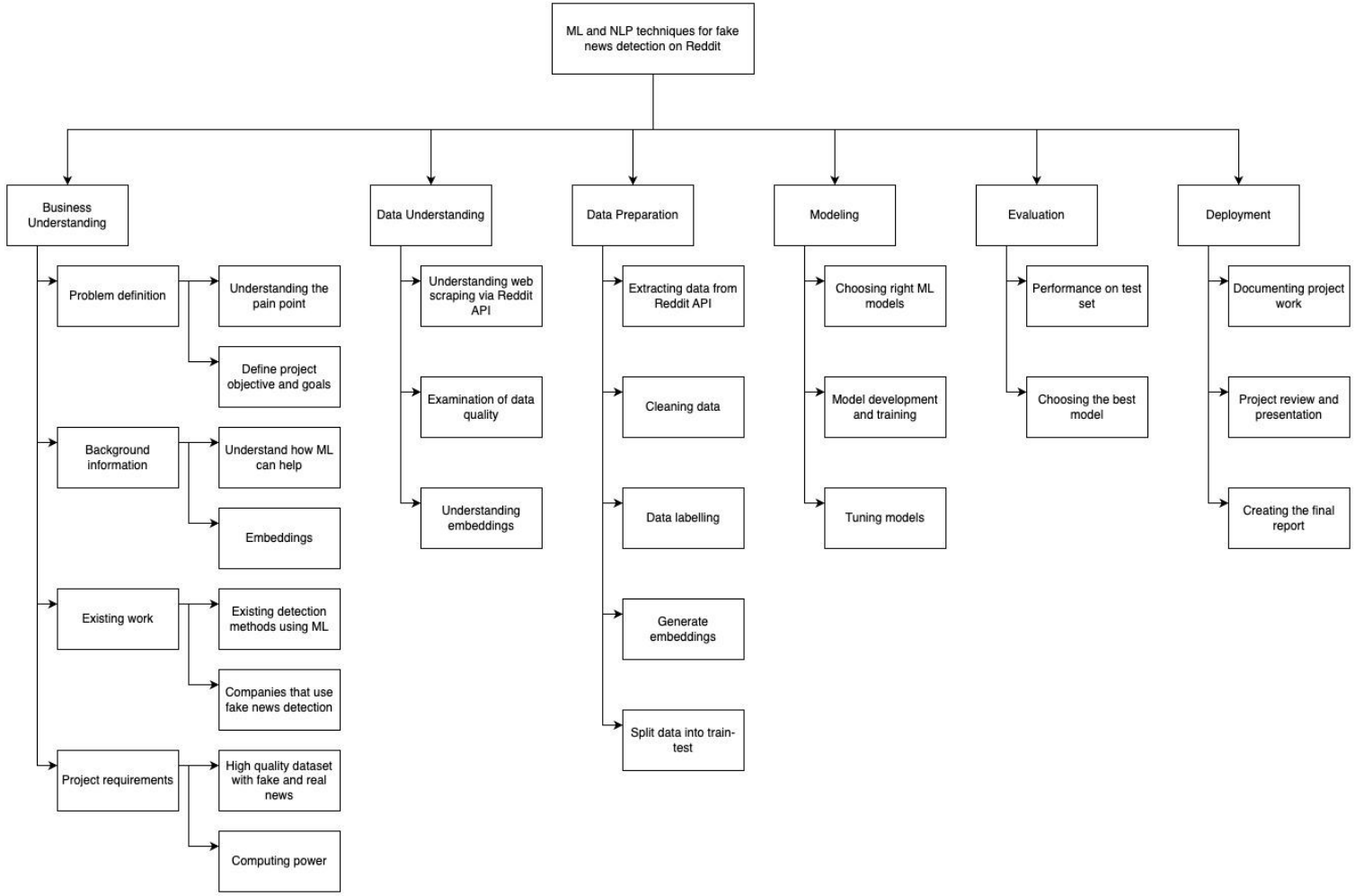



Figure: Work breakdown structure

# 2.4 Project Resource Requirements Plan

The project is implemented using tools such as

- **Hardware Req**: Local Machine 8GB RAM 64-Bit Version, minimum of 2 CPU cores
- **Data Cleaning and Preprocessing**: Excel, Python Jupyter Notebook, Google Collab
- **Machine Learning Framework and Algorithms**: Scikit Learn and Google Colab
- **Data Visualization**: Python Jupyter Notebook, Tableau

**Requirements of Reddit API Praw**

certifi==2023.7.22, charset-normalizer==3.2.0, idna==3.4, numpy==1.26.0, pandas==2.1.1, praw==7.7.1, prawcore==2.3.0, python-dateutil==2.8.2, pytz==2023.3.post1, requests==2.31.0, six==1.16.0, tzdata==2023.3, update-checker==0.18.0, urllib3==2.0.5, websocket-client==1.6.3

# 2.4 Project Resource Requirements Plan (continued)

| Utility | Rss Type | Tool | Duration | Estimated Cost |
|---|---|---|---|---|
| Cloud Service | Software | AWS EC2 | 3 Months | Free |
| Cloud Service | Software | AWS Eventbridge | 3 Months | Free |
| Cloud Service | Software | AWS Lambda | 3 Months | Free |
| Data Extraction | Software | Reddit API PRAW | 3 Months | Free |
| Cloud Service Storage) | Software | AWS S3 (S3 Glacier) | 2 Months | Free / < 5$ |
| Local Machine | Hardware | 64 - Bit Version | 3 Months | $1000 |
| Visualisation Tool | Software | Tableau | 1 Month | Free Student Lic |
| Machine Learning algos and frameworks | Software | Google Colab, Tensor Flow, Scikit-Learn | 2 Months | Free |
| Data Pre-processing and cleaning | Software | Google Colab, Python Jupyter Notebook, Excel | 1 Month | Free |

Table: Resources and Cost Estimation

# 2.5 Project Schedule

| Project | Status | Timeline |
|---|---|---|
| BUSINESS UNDERSTANDING | Done | August 28, 2023 → September 10, 2023 |
| Problem Def | Done | August 28, 2023 → August 31, 2023 |
| Understanding the pain point | Done | August 28, 2023 → August 31, 2023 |
| Define project objective and goals | Done | August 30, 2023 → September 2, 2023 |
| Background Info | Done | August 31, 2023 → September 3, 2023 |
| Understand how ML can help | Done | August 31, 2023 → September 3, 2023 |
| Embeddings | Done | September 3, 2023 → September 6, 2023 |
| Existing Work | Done | September 4, 2023 → September 7, 2023 |
| Existing detection methods using ML | Done | September 4, 2023 → September 7, 2023 |
| Companies that use fake news detection | Done | September 5, 2023 → September 8, 2023 |
| Project Requirement | Done | September 6, 2023 → September 9, 2023 |
| High quality dataset with fake and real new | Done | September 6, 2023 → September 9, 2023 |
| Computing power | Done | September 7, 2023 → September 10, 2023 |
| + New sub-item | | |
| DATA UNDERSTANDING | Done | September 11, 2023 → September 24, 2023 |
| Understanding web scraping via Reddit API | Done | September 11, 2023 → September 16, 2023 |
| Explore Reddit API Documentation | Done | September 11, 2023 → September 14, 2023 |
| Setup API Access | Done | September 15, 2023 → September 18, 2023 |
| Examination of data quality | Done | September 17, 2023 → September 20, 2023 |
| Understanding embeddings | Done | September 21, 2023 → September 24, 2023 |
| + New sub-item | | |
| DATA PREPARATION | Done | September 25, 2023 → October 22, 2023 |
| Extracting data from Reddit API | Done | September 25, 2023 → October 6, 2023 |
| Data Retrival & Pagination | Done | September 25, 2023 → September 28, 2023 |
| Data Parsing & Transformation | Done | September 29, 2023 → October 2, 2023 |
| Data quality checks | Done | October 3, 2023 → October 6, 2023 |
| Cleaning data | Done | October 7, 2023 → October 10, 2023 |
| Data labelling | Done | October 11, 2023 → October 14, 2023 |
| Generate embeddings | Done | October 15, 2023 → October 18, 2023 |
| Split data into train-val-test | Done | October 19, 2023 → October 22, 2023 |
| + New sub-item | | |
| MODELING | Done | October 23, 2023 → November 5, 2023 |
| choosing right ML and DL models | Done | October 23, 2023 → October 24, 2023 |
| Model development and training | Done | October 25, 2023 → October 28, 2023 |
| Tuning models | Done | October 29, 2023 → November 1, 2023 |
| Ensembles | Done | November 2, 2023 → November 5, 2023 |
| + New sub-item | | |
| EVALUATION | Done | November 6, 2023 → November 12, 2023 |
| Performance on test set | Done | November 6, 2023 → November 7, 2023 |
| Choosing the best model | Done | November 8, 2023 → November 12, 2023 |
| Model selection and exploration | Done | November 8, 2023 → November 9, 2023 |
| Hyperparameter Tuning | Done | November 10, 2023 |
| Model Comparison & Selection | Done | November 11, 2023 → November 12, 2023 |
| + New sub-item | | |
| DEPLOYMENT | Done | November 13, 2023 → November 27, 2023 |
| Documenting project work | Done | November 13, 2023 → November 15, 2023 |
| Project review and presentation | Done | November 16, 2023 → November 21, 2023 |
| Prepare Presentation Materials | Done | November 16, 2023 → November 19, 2023 |
| Practice and Rehearse | Done | November 20, 2023 → November 21, 2023 |
| Creating the final report | In progress | November 22, 2023 → November 27, 2023 |
| Compile Documentation OPEN | In progress | November 22, 2023 → November 24, 2023 |
| Format and Design | In progress | November 25, 2023 |
| Review and Proofreading | In progress | November 26, 2023 → November 27, 2023 |

Figure: GANTT Chart

# 2.5 Project Schedule (continued)

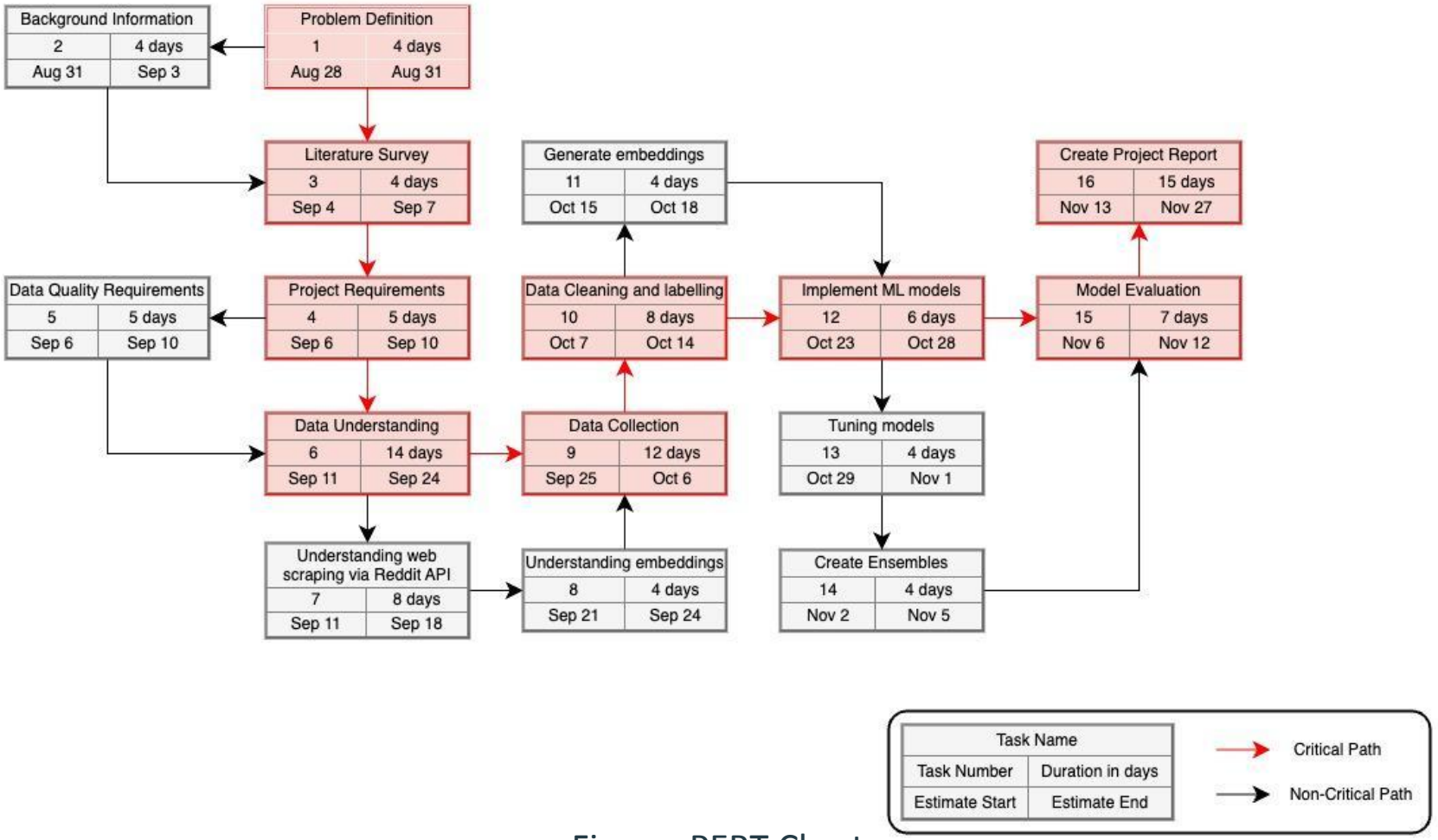



Figure: PERT Chart

# 3. Data Engineering

# 3.1 Data Process

Data Collection with PRAW: The project utilizes the PRAW library to gather data from Reddit, focusing on popular posts from six specific subreddits. Information collected includes post details such as title, body text, likes, comments, URL, and post IDs.
Data Storage on AWS: The collected data is initially stored on EBS volumes before being transferred to S3 buckets. AWS EC2 instances and data collection automation via a CRON process provide a scalable and efficient solution for continuous data gathering.
Hot Post Focus: The data collection method specifically targets 'hot' posts, defined as newly created posts that have gained significant upvotes relative to their age. This ensures that the dataset reflects the most relevant and engaging content actively interacted with by Reddit users.
Train-Test Split: The collected data is divided into training, and test sets, following machine learning best practices. The training set is used for model training, and the test set for evaluating models on previously unseen data.
Scheduled Operation for Timely Updates: A CRON process enables a scheduled operation to run every six hours, ensuring the continuous collection of current and upvoted data points. This approach allows for a thorough examination of ongoing trends and patterns in post interactions on Reddit.

# 3.2 Data Collection

1. Wrote a python program which collects 100 ”hot” posts from a subreddit.
2. Iterated it through six different subreddits to collect 600 datapoints per run.
3. Uploaded the script to AWS EC2 instance.
4. Scheduled CRON job to run the python script once every 6 hours.
5. Tools used: Python, Reddit API & PRAW, AWS EC2 & CRON

| Variable Name | Type | Description |
|---|---|---|
| title | Text | Title of the post |
| selfText | Text | Body of the post |
| author | Text | Author of the post |
| subreddit | Text | Page it was posted on |
| created_utc | Timestamp | Timestamp when the post was created |
| upvote_ratio | Float | No. of upvotes / Total votes |
| num_comments | Integer | Number of Comments on the post |
| permalink | Text | Internal Reddit path to the post |
| url | Text | Link to the post |

To use the reddit API smoothly we need client_id and client_secret, this can be done by registering as Developer on the Reddit Dev website.
Add the client_id and secret as environment variable in your system.
In the code below we are accessing the client_id and client_secret using the os library and the method getenv.
This was we are successfully authenticated to use the reddit API and our client_id and client_secret remains secured.

# 3.3 Data Pre-processing

Image Text Extraction: To utilize image data in the dataset, a script downloads images using the URL column. Pytesseract extracts text from images, adding it to the dataset under the column 'image_text'. This, along with post titles and selftext, forms the basis for further processing.

Text Preprocessing Steps: The raw text from posts and images undergoes extensive preprocessing. New line characters and special characters are removed, and the text is converted to lowercase for consistency and reduced vocabulary size. URLs are removed to focus on semantic content, and tokenization using word_tokenize aids in understanding sentence structure.

Stemming and Lemmatization: Stemming (using PorterStemmer) reduces words to their root form, simplifying vocabulary. Lemmatization (using WordNetLemmatizer) ensures valid words and captures nuanced meanings for accurate classification.

Concatenation and Final Preprocessing: The 'image_text' is loaded into a new column, combined with 'selftext' and 'title', and subjected to preprocessing methods. The entire dataset is then converted to lowercase, characters are removed.

# 3.4 DATA PRE-PROCESSING (RAW DATA)

```
# Raw Data
df.head()
```

[37] ✓ 0.0s Python

| title | image_text | selftext | author | subreddit | created_utc | upvote_ratio | num_comments | permalink | |
|---|---|---|---|---|---|---|---|---|---|
| Family of Armed Robbery Suspect Outraged Pizza... | Following the shooting death of 28-year-old ar... | NaN | NYPD-32 | WhitePeopleTwitter | 1480870334 | 0.68 | 21 | /r/whitepeopletwitter/comments/5ggakpv/Family_... | https |
| Hurricane Irma likely to drop to Category 4 up... | (Reuters) - Hurricane Irma is likely to be dow... | NaN | NYPD-32 | fakenews | 1480870334 | 0.67 | 21 | /r/fakenews/comments/5ggakps/Hurricane_Irma_likel | https://w |
| Factbox: Trump on Twitter (July 26) - U.S. Mil... | The following statementsÊwere posted to the ve... | NaN | NYPD-32 | fakenews | 1480870334 | 0.67 | 21 | /r/fakenews/comments/5ggakpq/Factbox:_Trump_on_Tw | https://w |
| Senior Saudi prince freed in settlement agreem... | DUBAI (Reuters) - Senior Saudi Prince Miteb bi... | NaN | NYPD-32 | fakenews | 1480870334 | 0.68 | 21 | /r/fakenews/comments/5ggakpn/Senior_Saudi_prince_ | https://w |
| WATCH: REMEMBER WHEN HILLARY SAID She WasnÕt D... | That Hillary, she sure is a forked-tongued sil... | NaN | NYPD-32 | WhitePeopleTwitter | 1480870334 | 0.68 | 21 | /r/whitepeopletwitter/comments/5ggakpl/WATCH:_... | https |

# 3.4 DATA PRE-PROCESSING (AFTER)

```python
import re

def preProcess(input_string):
    # Define a regular expression pattern for matching text inside square brackets
    pattern1 = re.compile(r'\[.*?\]')
    # Define a regular expression pattern for removing special characters
    pattern2 = re.compile(r'[^A-Za-z0-9\s]')

    # Use sub() function to replace the matched pattern with an empty string
    brackettext_remove = re.sub(pattern1, '', input_string)

    # strip spaces from start and end
    stripped = brackettext_remove.strip()

    # convert the string to lower
    to_lower = stripped.lower()

    # remove special characters from the string
    final_text = re.sub(pattern2, '', to_lower)

    return final_text

imp_variables.loc[:, 'concatenated'] = imp_variables.concatenated.apply(preProcess)
```

(variable) imp_variables: DataFrame

[16] ✓ 7.7s Python

```python
# After PreProcessing
imp_variables.head()
```

[17] ✓ 0.0s Python

| | concatenated | label | upvote_ratio | no_of_words |
|---|---|---|---|---|
| 0 | family of armed robbery suspect outraged pizza... | 1 | 0.68 | 477 |
| 1 | hurricane irma likely to drop to category 4 up... | 0 | 0.67 | 249 |
| 2 | factbox trump on twitter july 26 us military ... | 0 | 0.67 | 242 |
| 3 | senior saudi prince freed in settlement agreem... | 0 | 0.68 | 85 |
| 4 | watch remember when hillary said she wasnt dro... | 1 | 0.68 | 174 |

# 3.5 Data Transformation

Text Data Transformation Techniques: Three distinct techniques - Count Vectorization, TF-IDF Vectorization, and GloVe Embedding - are employed to prepare text data for a fake news identification machine learning model.
Count Vectorization: Represents documents as word frequency vectors, creating high-dimensional sparse matrices. Each unique term serves as a feature, forming the basis for subsequent model training.
TF-IDF Vectorization: Extends Count Vectorization by considering term importance in the entire corpus. Higher weights are assigned to terms that are frequent in a document but rare in the corpus, enhancing the model's ability to discern relevance.
GloVe Embedding: Utilizes pre-trained word embeddings to transform words into dense vectors based on global statistical information. Captures semantic relationships, allowing the model to understand context and meaning, enhancing nuanced semantics in fake news detection.
Multifaceted Approach for Model Enhancement: Incorporating these vectorization methods transforms raw text data into numerical formats, providing structured input for machine learning. This multifaceted approach captures diverse aspects of the data, empowering the model to discern subtle linguistic nuances crucial for accurate classification.

# DATA TRANSFORMATION (COUNT VECTORIZATION)

```
print(X_train_tfidf_matrix)
```

[62] ✓ 0.2s Python

```
  (0, 166)      0.19389766563144004
  (0, 169)      0.07285269605461099
  (0, 178)      0.03366007770634383
  (0, 281)      0.04150702008433275
  (0, 324)      0.029628595649727432
  (0, 396)      0.05726388988673884
  (0, 486)      0.5403692122437811
  (0, 541)      0.061538547186009464
  (0, 543)      0.05561219387997997
  (0, 565)      0.06701473346203214
  (0, 743)      0.027710285839168625
  (0, 813)      0.04360067750201602
  (0, 927)      0.16720305810640748
  (0, 1010)     0.1792497685390776
  (0, 1028)     0.054448853558770285
  (0, 1100)     0.051650411440789475
  (0, 1250)     0.034991485013416444
  (0, 1252)     0.04290622807533482
  (0, 1264)     0.048228818637214683
  (0, 1490)     0.05529060256571881
  (0, 1495)     0.03733794417761238
  (0, 1509)     0.04329017684213351
  (0, 1531)     0.053143070686464434
  (0, 1696)     0.06035305066492797
  (0, 1752)     0.07194086622418358
...
  (42471, 4952) 0.03233917980264867
  (42471, 4960) 0.036013242525011334
  (42471, 4964) 0.025513175508844244
  (42471, 4972) 0.042474279806041675
```

*Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...*

# DATA TRANSFORMATION (TF-IDF)

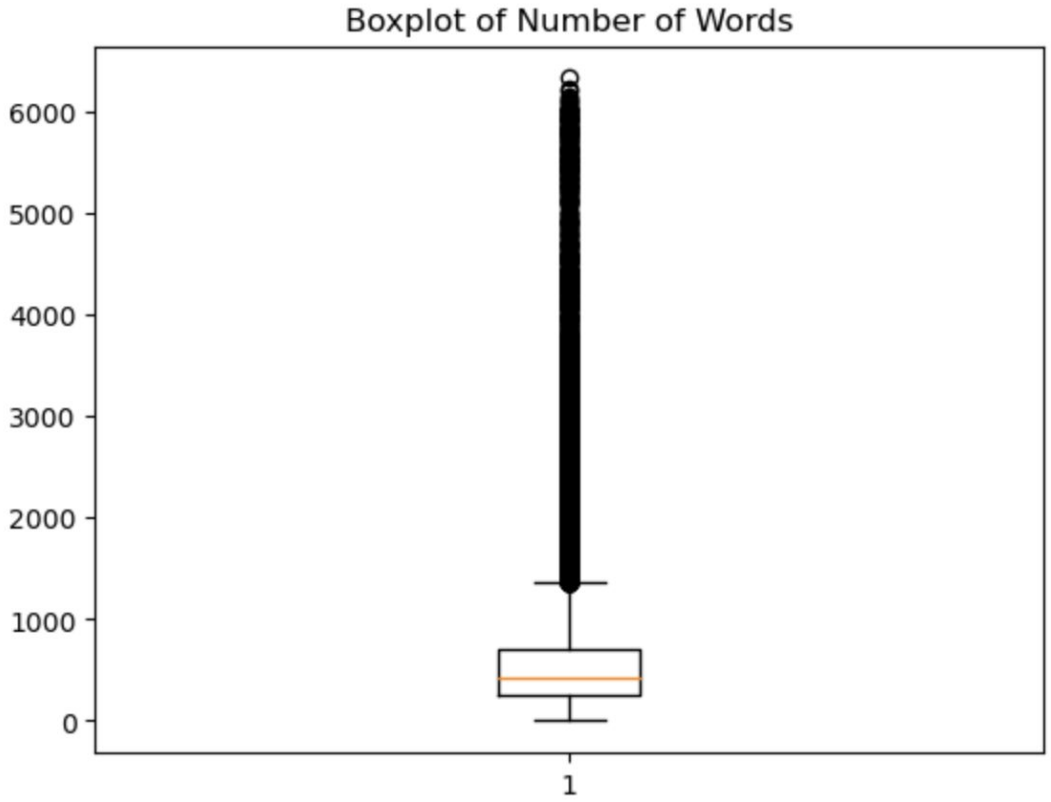
```
print(X_train_count_matrix)
```

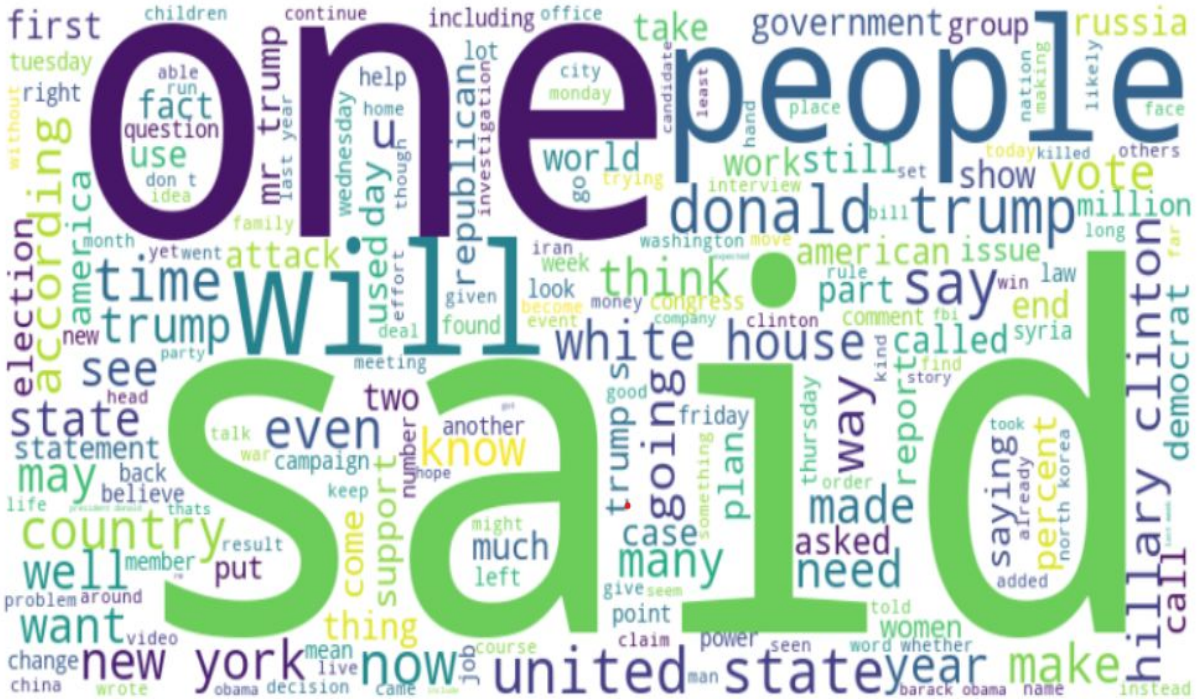
[61] ✓ 0.1s Python

```
  (0, 166)      5
  (0, 169)      2
  (0, 178)      1
  (0, 281)      2
  (0, 324)      1
  (0, 396)      2
  (0, 486)      12
  (0, 541)      2
  (0, 543)      1
  (0, 565)      1
  (0, 743)      1
  (0, 813)      1
  (0, 927)      5
  (0, 1010)     3
  (0, 1028)     1
  (0, 1100)     1
  (0, 1250)     1
  (0, 1252)     1
  (0, 1264)     1
  (0, 1490)     1
  (0, 1495)     1
  (0, 1509)     1
  (0, 1531)     2
  (0, 1696)     1
  (0, 1752)     1
...
  (42471, 4952) 1
  (42471, 4960) 1
  (42471, 4964) 1
  (42471, 4972) 3
```

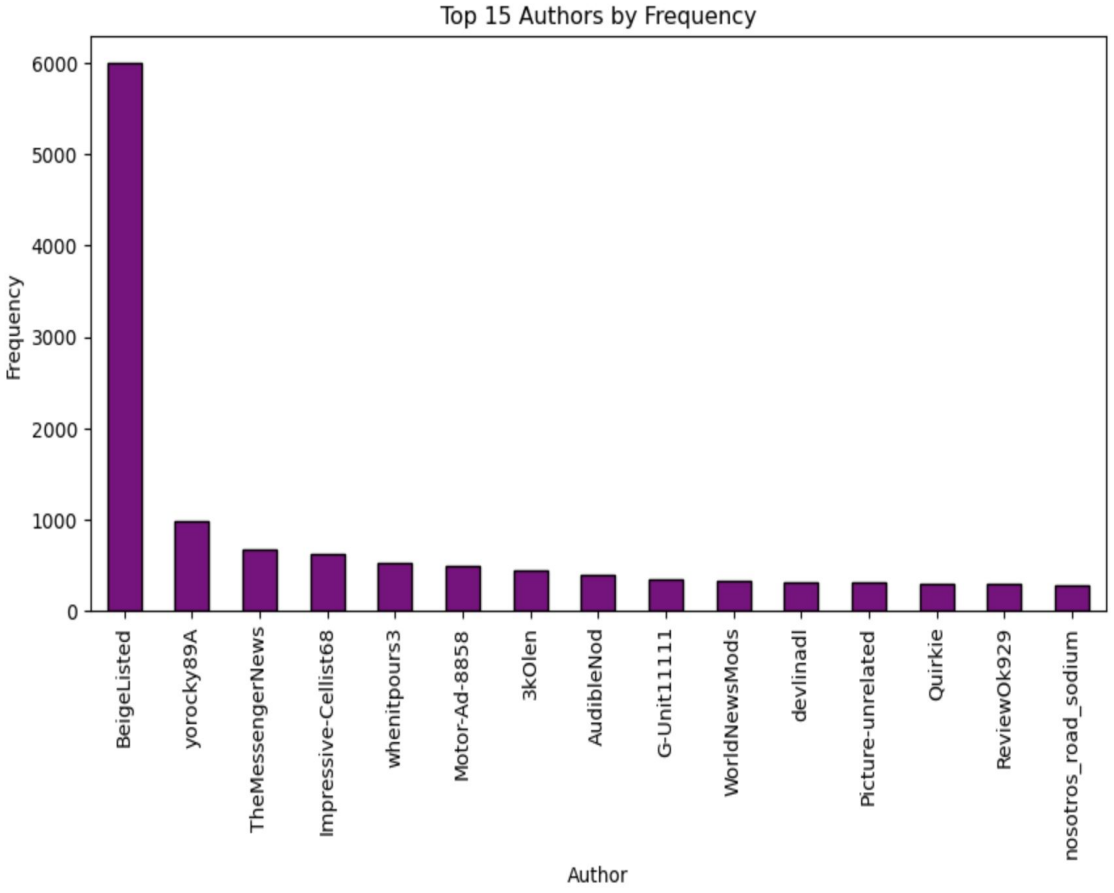
*Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...*

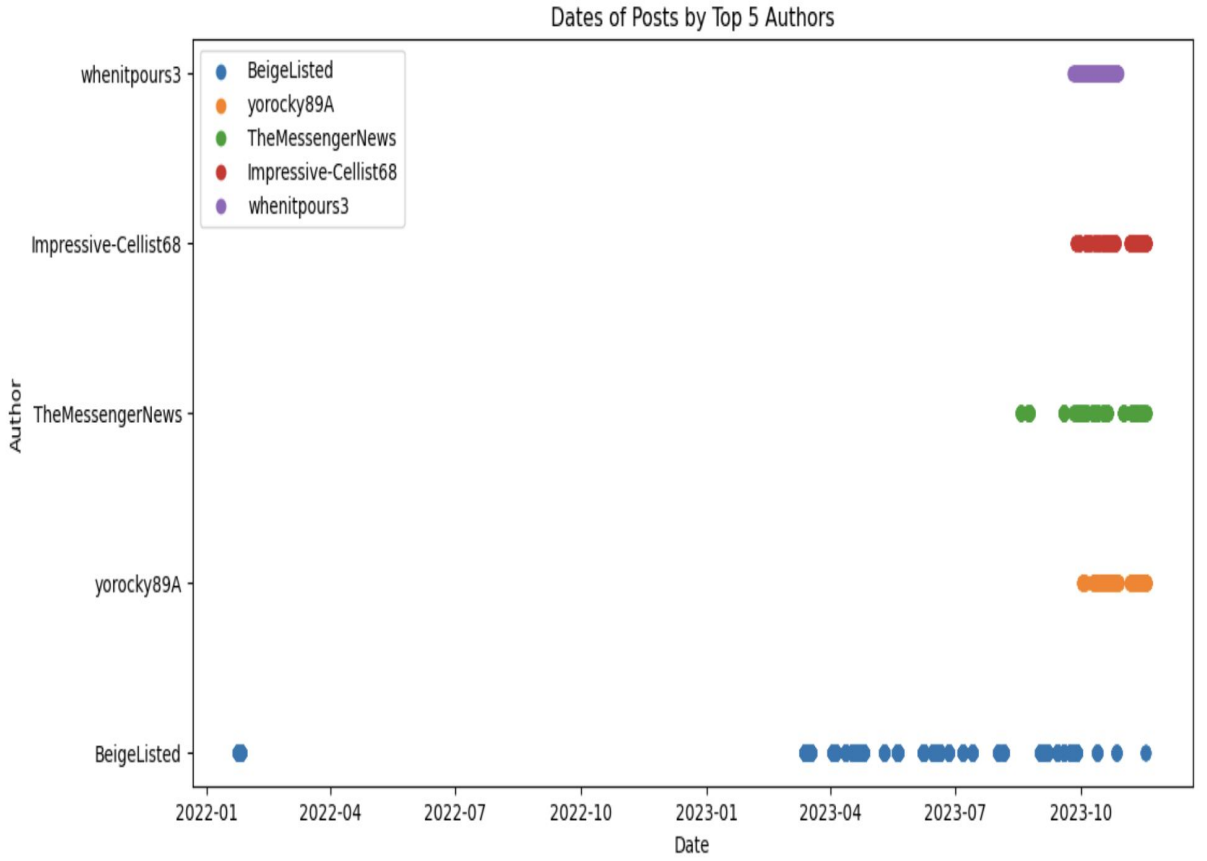
# 3.6 Exploratory Data Analysis



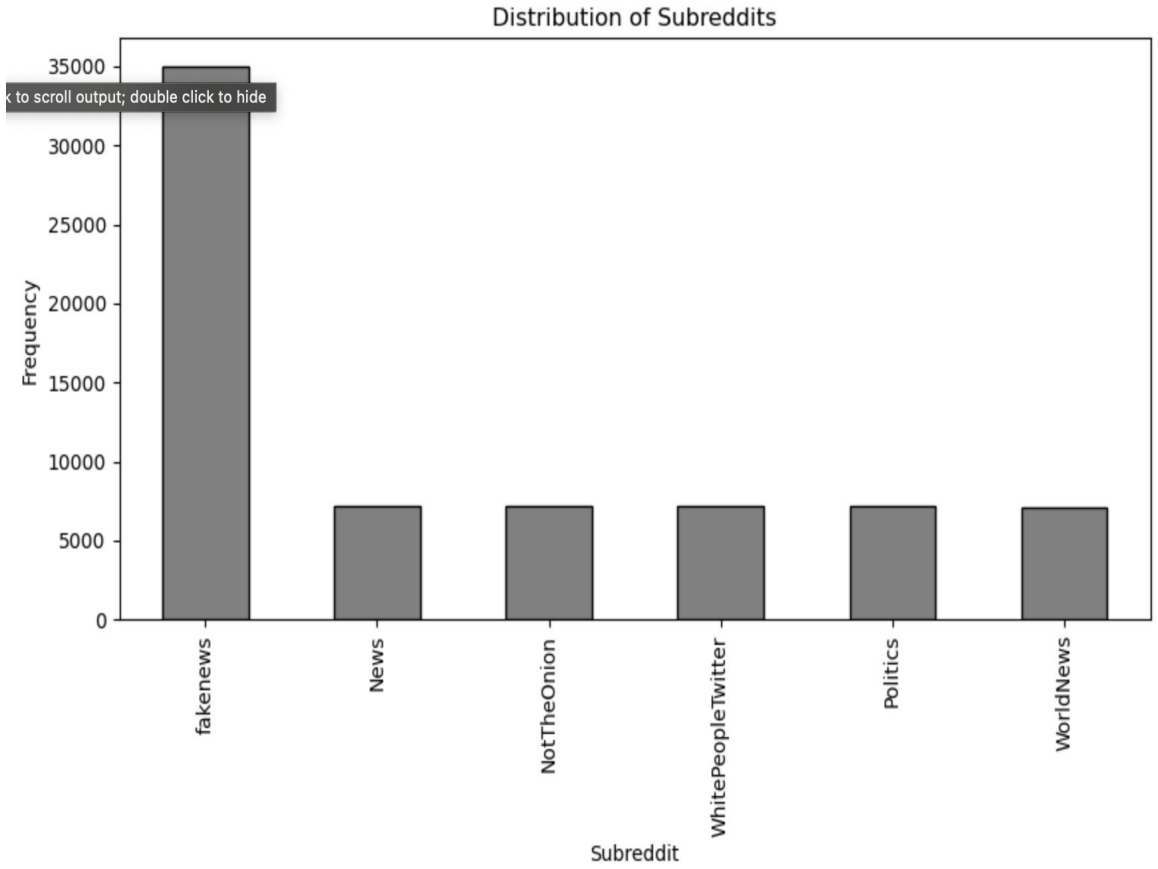



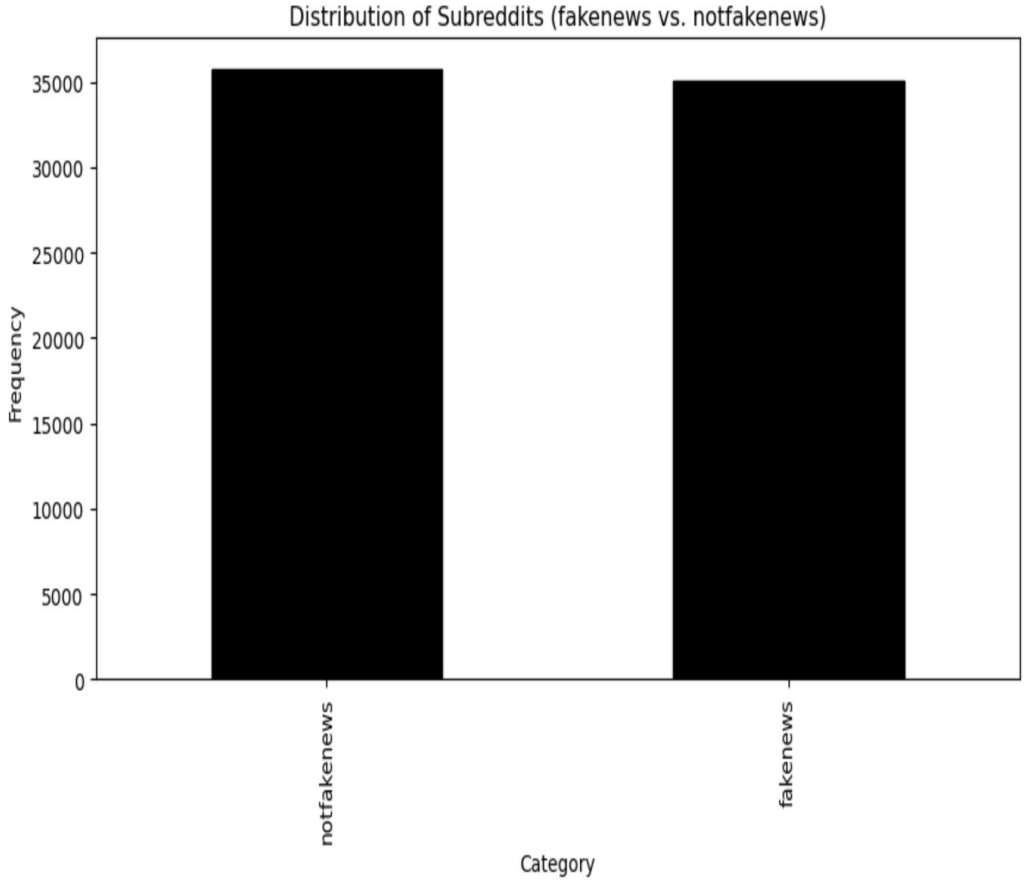
# 3.6 Exploratory Data Analysis (continued)



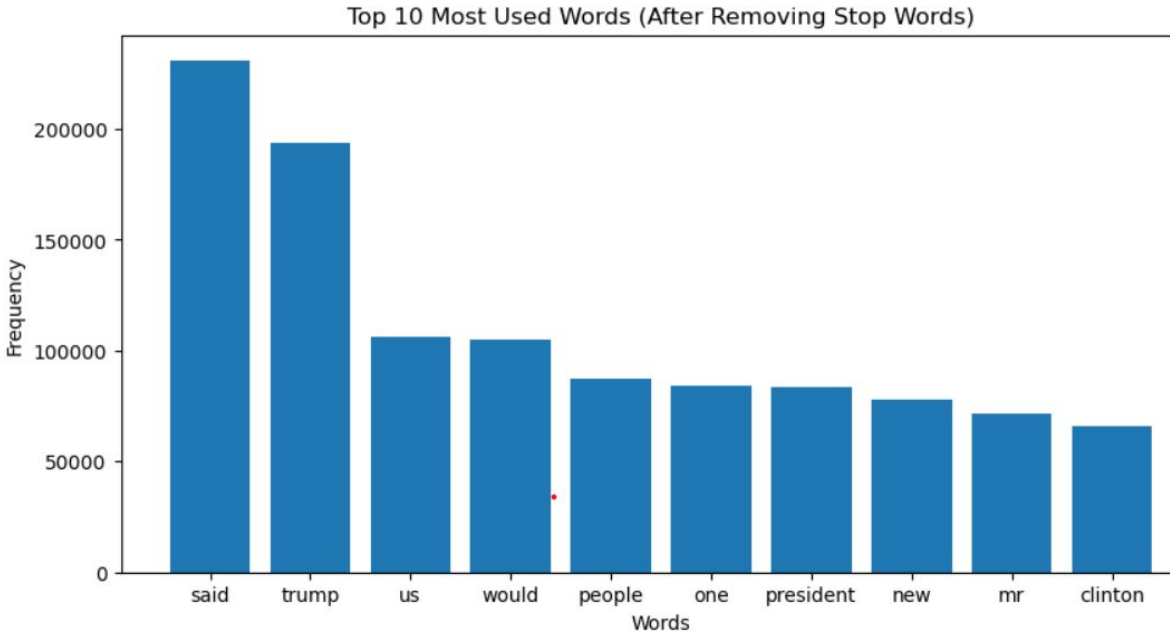



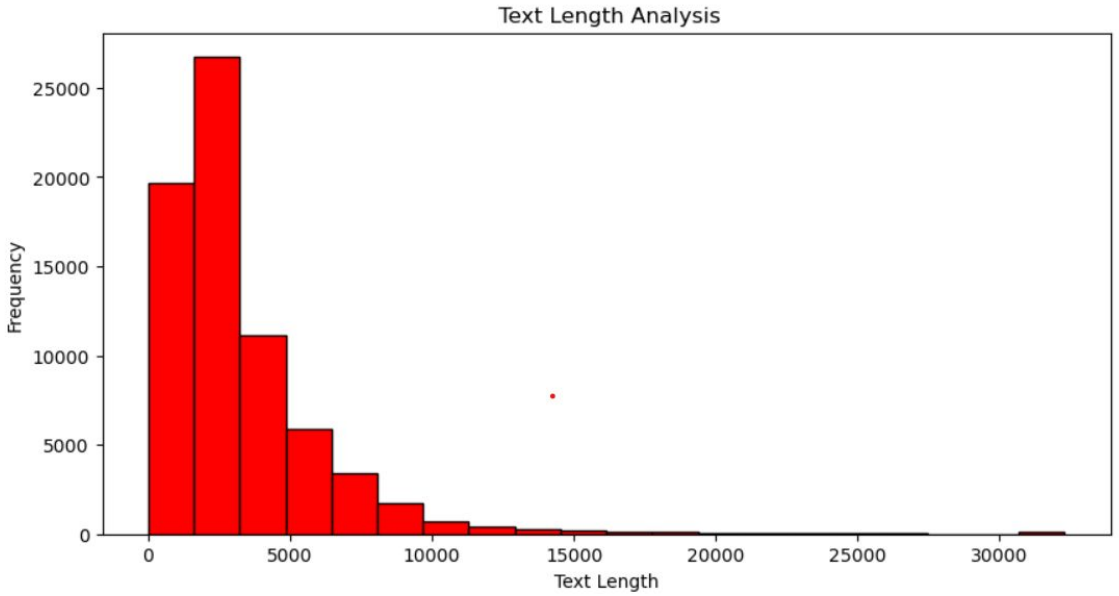
# 3.6 Exploratory Data Analysis (continued)



# 3.6 Exploratory Data Analysis (continued)



# 4. Model Development

# 4.1 Model Proposals

- Logistic Regression: A linear regression-based classification model that uses a logistic function to estimate probabilities.
- Support Vector Machine (SVM): A supervised learning model that maps input data to a high-dimensional space using kernel functions to separate classes.
- Decision Tree: A tree-structured model that splits data based on feature values to create hierarchical decision rules.
- Random Forest: An ensemble learning method that combines multiple decision trees trained on random subsets of features and data instances.
- Naive Bayes: A probabilistic classifier based on Bayes' theorem, assuming feature independence.

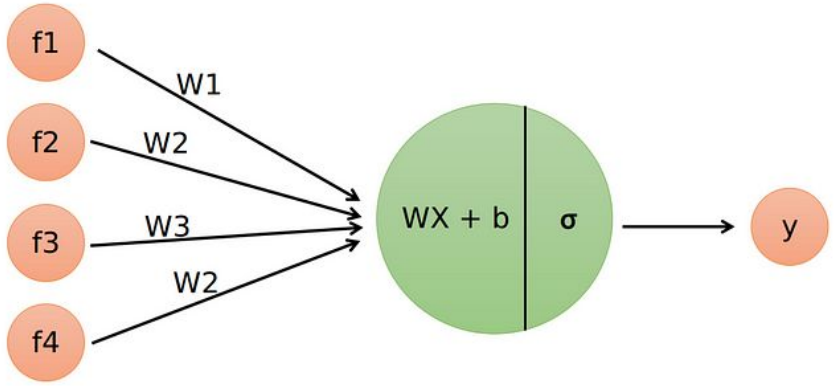
# 4.2 Model Supports

Tools:

- **Python**: Python is a widely used programming language for data analytics and machine learning. It offers extensive libraries such as scikit-learn, pandas, and numpy, which are essential for model development.
- **Jupyter Notebook**: Jupyter Notebook provides an interactive environment for writing and executing code. It allows you to document your code, visualize data, and share your work with others.
- **scikit-learn**: scikit-learn is a powerful machine learning library in Python. It provides implementations of various models, including logistic regression, SVM, decision trees, random forests, and Naive Bayes. It also offers tools for data preprocessing, model evaluation, and cross-validation.

# Logistic Regression

- Binary Classification: Logistic regression is a statistical method used for binary classification problems, making it suitable for distinguishing between real and fake news.
- Interpretability: It provides a clear interpretation of the relationship between input features and the likelihood of a news article being fake.
- Efficiency: Logistic regression is computationally efficient and can handle large datasets, making it practical for processing a significant amount of news articles.



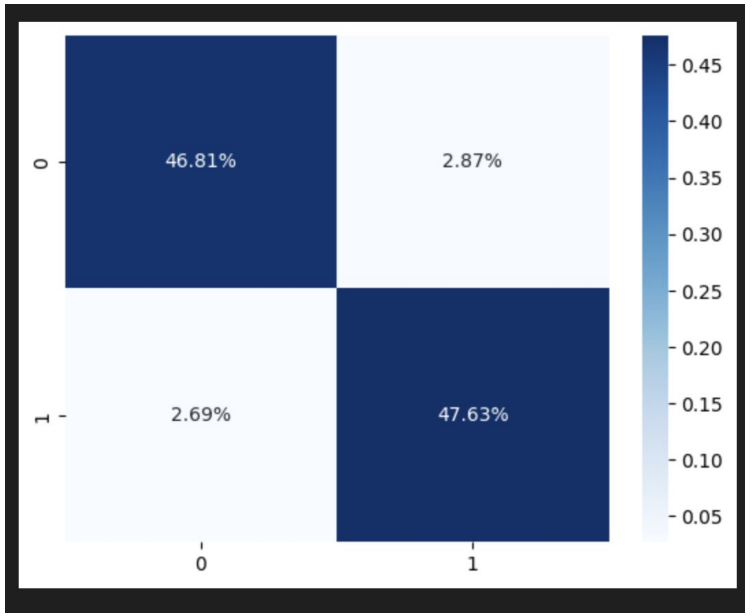
# 4.3 Model Comparison

## 4.3.2) Logistic Regression with Tf-IDF Vectorized Values

```
model_name = 'TfIDFVectorizerLR.pkl'

# Create a Logistic Regression model
tm_logistic_model = LogisticRegression(random_state=42)

# Train the model
tm_logistic_model.fit(X_train_tfidf_matrix, y_train)

# Make predictions on the test set
tm_log_predictions = tm_logistic_model.predict(X_test_tfidf_matrix)
evaluateModel(tm_log_predictions)

pickle.dump(tm_logistic_model, open(save_models_path+"_"+version+"_"+model_name,'wb'))
```

[31] ✓ 1.4s Python

```
Accuracy: 0.944413052691058

Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.94      0.94      7034
           1       0.94      0.95      0.94      7124

    accuracy                           0.94     14158
   macro avg       0.94      0.94      0.94     14158
weighted avg       0.94      0.94      0.94     14158


Confusion Matrix:
 [[6628  406]
 [ 381 6743]]
```



# 4.3 Model Comparison

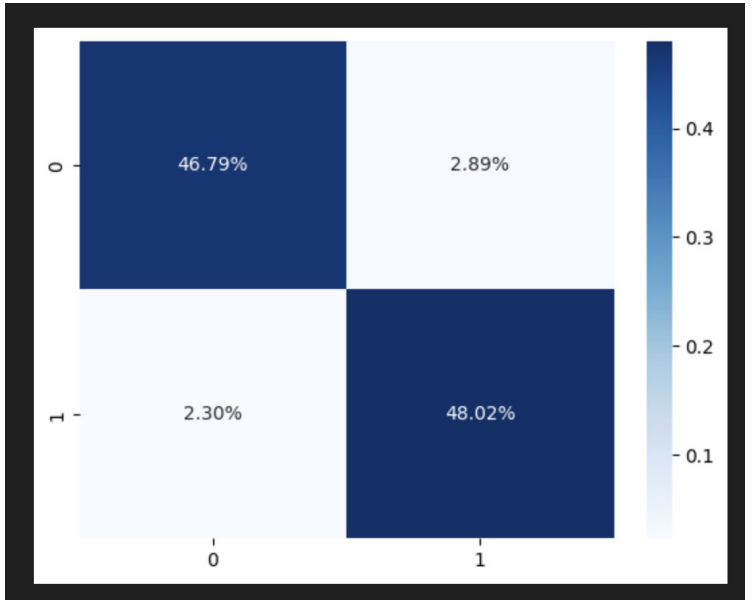
## 4.3.1) Logistic Regression with Count Vectorized Values

LogisticRegression

```
model_name = 'CountVectorizerLR.pkl'

# Create a Logistic Regression model
cm_logistic_model = LogisticRegression(random_state=42)

# Train the model
cm_logistic_model.fit(X_train_count_matrix, y_train)
# Make predictions on the test set
cm_log_predictions = cm_logistic_model.predict(X_test_count_matrix)

evaluateModel(cm_log_predictions)

pickle.dump(cm_logistic_model, open(save_models_path+"_"+version+"_"+model_name,'wb'))
```

[30] ✓ 1.8s Python

```
Accuracy: 0.9481565192823845

Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.94      0.95      7034
           1       0.94      0.95      0.95      7124

    accuracy                           0.95     14158
   macro avg       0.95      0.95      0.95     14158
weighted avg       0.95      0.95      0.95     14158


Confusion Matrix:
 [[6625  409]
 [ 325 6799]]
```

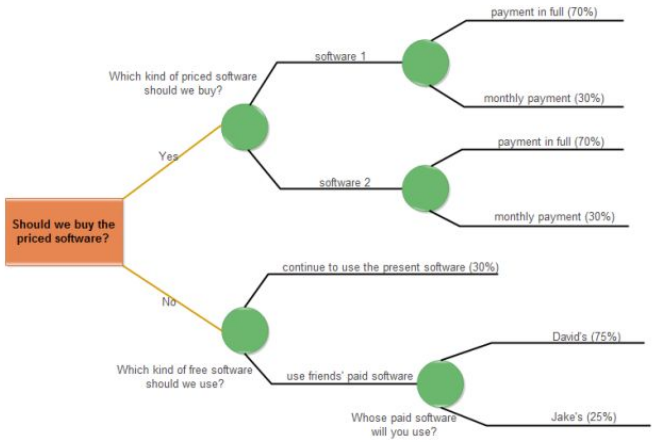


# Decision Tree

**Intuitive Decision Making**: Decision trees are easy to understand and interpret, making it simpler to identify the key features influencing the classification of news as real or fake.

**Feature Importance**: Decision trees can highlight important features for classification, aiding in understanding the criteria used to make decisions.

**Nonlinear Relationships**: Decision trees can capture nonlinear relationships in data, allowing them to discern intricate patterns in news articles that may indicate misinformation.



# 4.3 Model Comparison

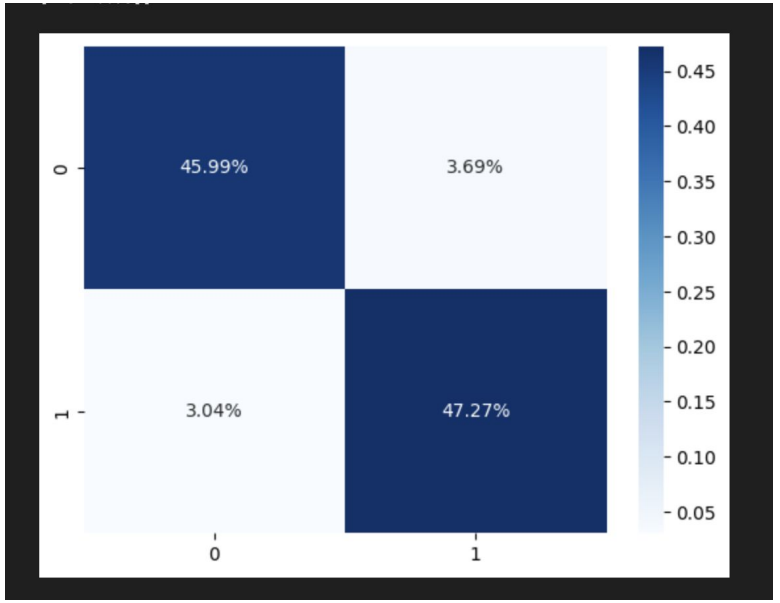
## 4.3.3) Decision Tree with Count Vectorized Values

Decision Tree

```
model_name = 'CountVectorizeDT'

cm_dt_model = DecisionTreeClassifier(random_state=42)
cm_dt_model.fit(X_train_count_matrix, y_train)
cm_dt_predictions = cm_dt_model.predict(X_test_count_matrix)

evaluateModel(cm_dt_predictions)

pickle.dump(cm_dt_model, open(save_models_path+"_"+version+"_"+model_name,'wb'))
```

[32] ✓ 28.6s Python

```
Accuracy: 0.9326176013561237

Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.93      0.93      7034
           1       0.93      0.94      0.93      7124

    accuracy                           0.93     14158
   macro avg       0.93      0.93      0.93     14158
weighted avg       0.93      0.93      0.93     14158


Confusion Matrix:
 [[6511  523]
 [ 431 6693]]
```



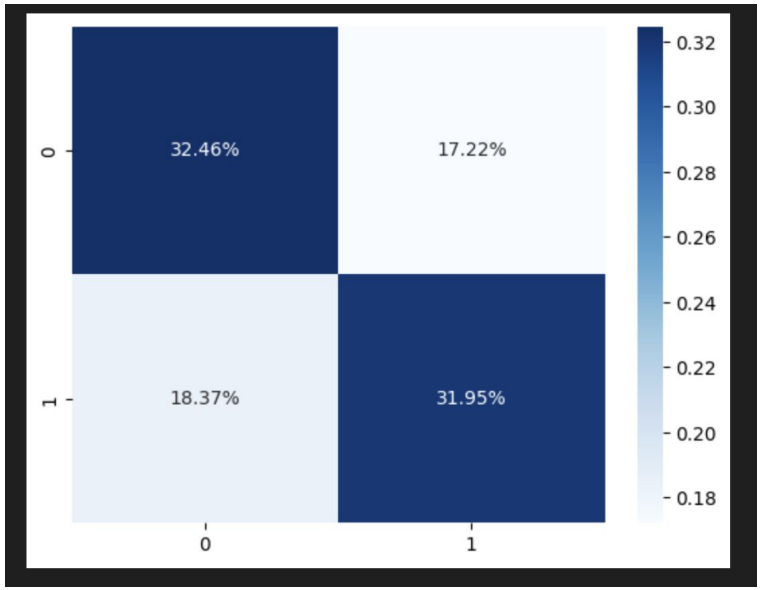
# 4.3 Model Comparison

## 4.3.4) Decision Tree with Tf-IDF Vectorized Values

```
model_name = 'TfidfVectorizeDT'

tm_dt_model = DecisionTreeClassifier(random_state=42)
tm_dt_model.fit(X_train_tfidf_matrix, y_train)
tm_dt_predictions = tm_dt_model.predict(X_test_count_matrix)

evaluateModel(tm_dt_predictions)

pickle.dump(tm_dt_model, open(save_models_path+"_"+version+"_"+model_name,'wb'))
```

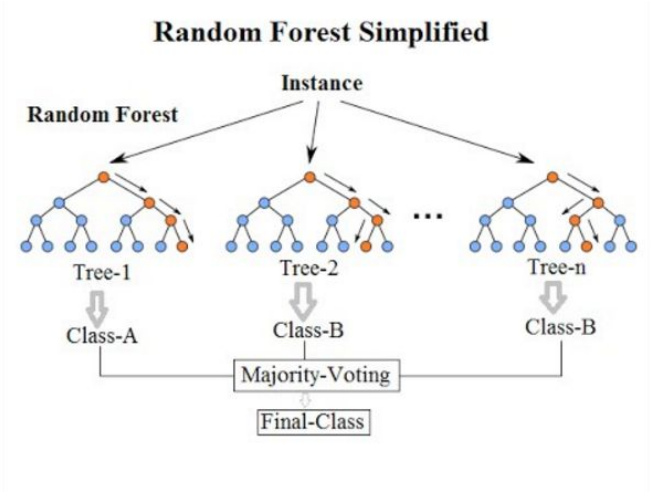
[33] ✓ 50.7s Python

```
Accuracy: 0.644088148043509

Classification Report:
              precision    recall  f1-score   support

           0       0.64      0.65      0.65      7034
           1       0.65      0.63      0.64      7124

    accuracy                           0.64     14158
   macro avg       0.64      0.64      0.64     14158
weighted avg       0.64      0.64      0.64     14158


Confusion Matrix:
 [[4596 2438]
 [2601 4523]]
```



# Random Forest

**Ensemble Learning**: Random Forest is an ensemble of decision trees, which enhances the model's accuracy and robustness by aggregating predictions from multiple trees.
**Reduced Overfitting**: The ensemble nature of Random Forest helps mitigate overfitting, providing better generalization to unseen data.
**Feature Importance**: Random Forest can provide insights into feature importance, aiding in identifying the most relevant factors for fake news classification.



# 4.3 Model Comparison

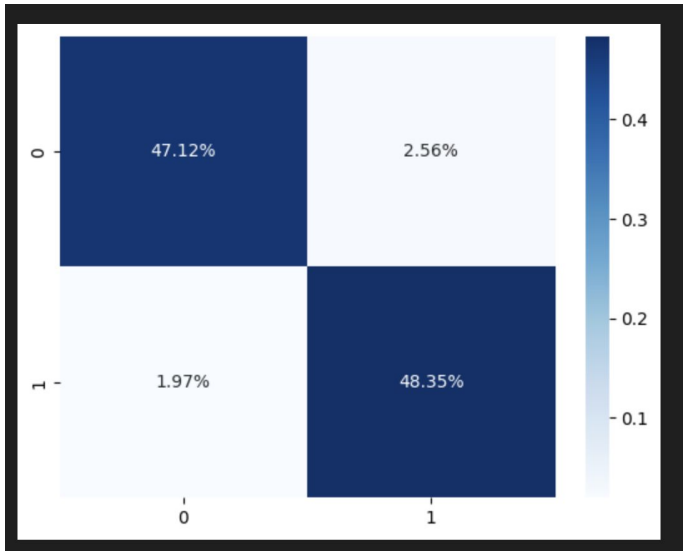
## 4.3.5) Random Forest with Count Vectorized Values

Random Forest

```
model_name = 'CountVectorizerRF'

cm_rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
cm_rf_model.fit(X_train_count_matrix, y_train)
cm_rf_predictions = cm_rf_model.predict(X_test_count_matrix)

evaluateModel(cm_rf_predictions)

pickle.dump(cm_rf_model, open(save_models_path+"_"+version+"_"+model_name,'wb'))
```

[34] ✓ 1m 39.0s Python

```
Accuracy: 0.9546546122333663

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.95      0.95      7034
           1       0.95      0.96      0.96      7124

    accuracy                           0.95     14158
   macro avg       0.95      0.95      0.95     14158
weighted avg       0.95      0.95      0.95     14158


Confusion Matrix:
 [[6671  363]
 [ 279 6845]]
```



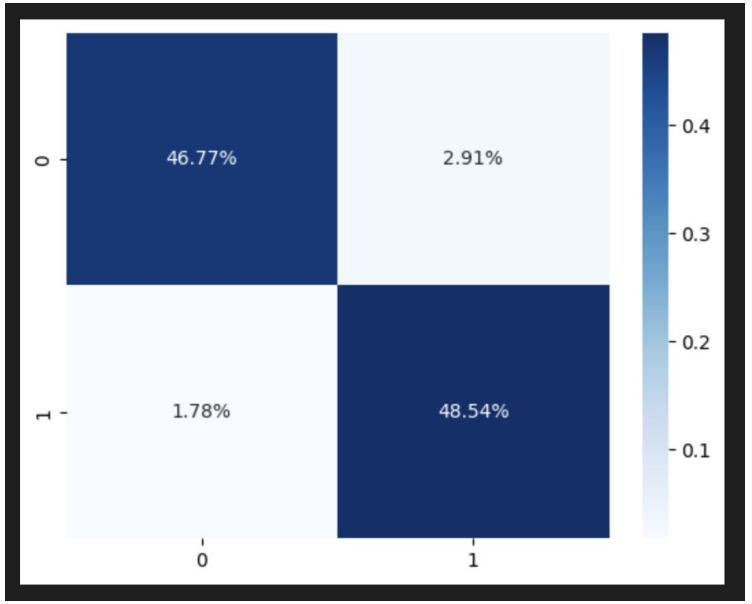
# 4.3 Model Comparison

## 4.3.6) Random Forest with Tf-IDF Vectorized Values

```
model_name = 'CountVectorizerRF'

tm_rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
tm_rf_model.fit(X_train_tfidf_matrix, y_train)
tm_rf_predictions = tm_rf_model.predict(X_test_tfidf_matrix)

evaluateModel(tm_rf_predictions)

pickle.dump(tm_rf_model, open(save_models_path+"_"+version+"_"+model_name,'wb'))
```

[35] ✓ 1m 36.9s Python

```
Accuracy: 0.9531007204407402

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      7034
           1       0.94      0.96      0.95      7124

    accuracy                           0.95     14158
   macro avg       0.95      0.95      0.95     14158
weighted avg       0.95      0.95      0.95     14158


Confusion Matrix:
 [[6622  412]
 [ 252 6872]]
```



# Support Vector Machine

**Effective in High-Dimensional Spaces**: SVM is powerful in high-dimensional spaces, making it suitable for tasks like text classification where the feature space can be vast.
**Margin Maximization**: SVM aims to find the hyperplane that maximizes the margin between different classes, providing a robust decision boundary.
**Kernel Tricks**: SVM can use kernel functions to handle non-linear relationships in data, allowing it to capture complex patterns in news articles that might indicate fake news.

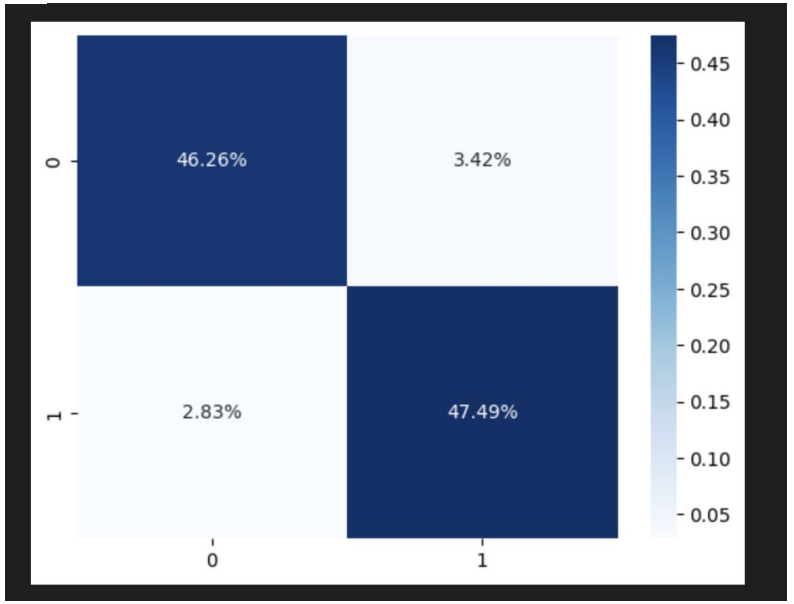
# 4.3 Model Comparison

## 4.3.7) SVM with Count-IDF Vectorized Values

```
model_name = 'CountVectorizerSVM.pkl'

# Create an SVM model
cm_svm_model = SVC(kernel='linear', random_state=42)

# Train the model
cm_svm_model.fit(X_train_count_matrix, y_train)

# Make predictions on the test set
cm_svm_predictions = cm_svm_model.predict(X_test_count_matrix)

evaluateModel(cm_svm_predictions)

pickle.dump(cm_svm_model, open(save_models_path+"_"+version+"_"+model_name,'wb'))
```

[28] ✓ 79m 7.5s Python

```
Accuracy: 0.9375618025144794

Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.93      0.94      7034
           1       0.93      0.94      0.94      7124

    accuracy                           0.94     14158
   macro avg       0.94      0.94      0.94     14158
weighted avg       0.94      0.94      0.94     14158


Confusion Matrix:
 [[6550  484]
 [ 400 6724]]
```



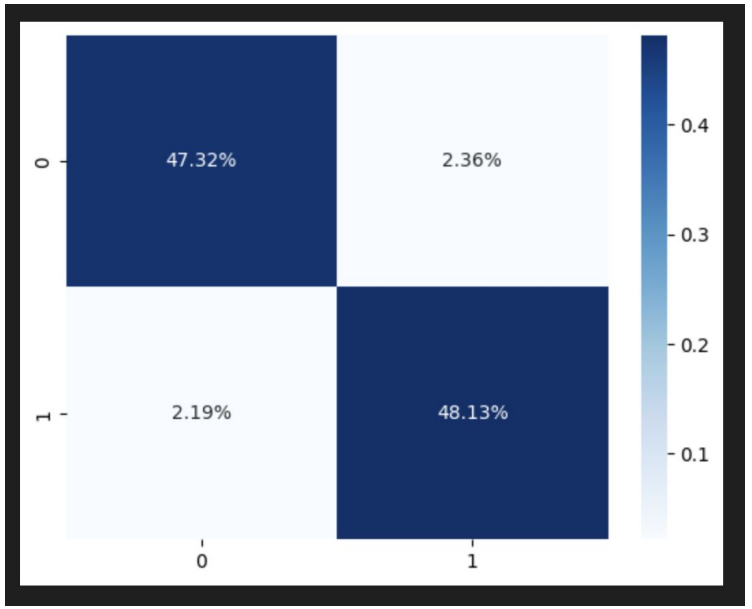
# 4.3 Model Comparison

## 4.3.8) SVM with Tf-IDF Vectorized Values

```python
model_name = 'TfIDFVectorizerSVM.pkl'

# Create an SVM model
tm_svm_model = SVC(kernel='linear', random_state=42)

# Train the model
tm_svm_model.fit(X_train_tfidf_matrix, y_train)

# Make predictions on the test set
tm_svm_predictions = tm_svm_model.predict(X_test_tfidf_matrix)

evaluateModel(tm_svm_predictions)

pickle.dump(tm_svm_model, open(save_models_path+"_"+version+"_"+model_name,'wb'))
```

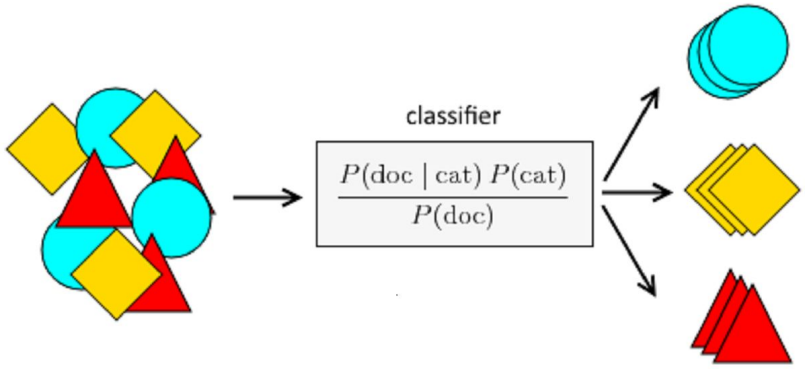
[29] ✓ 15m 49.2s Python

```
Accuracy: 0.9545133493431276

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.95      0.95      7034
           1       0.95      0.96      0.95      7124

    accuracy                           0.95     14158
   macro avg       0.95      0.95      0.95     14158
weighted avg       0.95      0.95      0.95     14158


Confusion Matrix:
 [[6700  334]
 [ 310 6814]]
```



# Naive Bayes

**Probabilistic Approach**: Naive Bayes is based on probabilistic principles, making it effective for modeling uncertainty and probability of an article being fake.
**Efficiency**: It is computationally efficient and can be trained on relatively small datasets, making it suitable for quick classification tasks.
**Text Classification**: Naive Bayes is particularly useful for text classification, as it can handle the high-dimensional and sparse nature of text data, which is common in news articles.



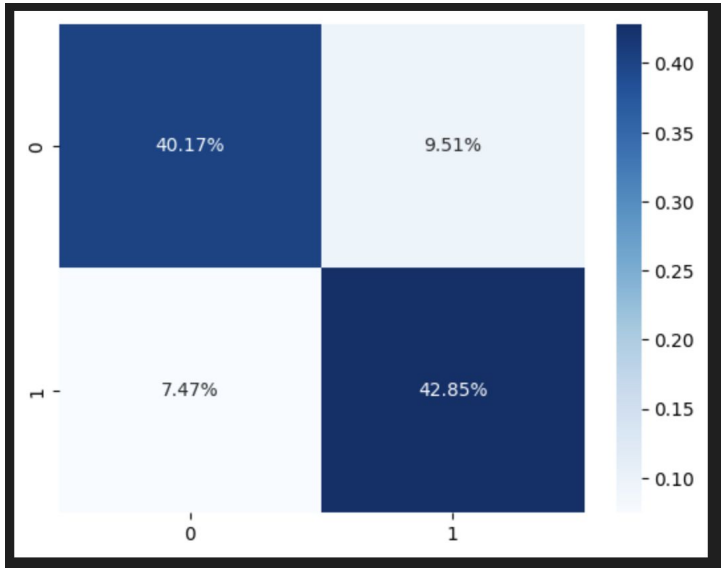
# 4.3 Model Comparison

## 4.3.9) Naive Bayes with Count Vectorized Values

```
from sklearn.naive_bayes import GaussianNB

cm_nb_model = GaussianNB()

cm_nb_model.fit(X_train_count_matrix.toarray(), y_train)
cm_nb_predictions = cm_nb_model.predict(X_test_count_matrix.toarray())

evaluateModel(cm_nb_predictions)

pickle.dump(cm_nb_model, open(save_models_path+"_"+version+"_"+model_name,'wb'))
```

[64] ✓ 8.5s Python

```
Accuracy: 0.8302020059330414

Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.81      0.83      7034
           1       0.82      0.85      0.83      7124

    accuracy                           0.83     14158
   macro avg       0.83      0.83      0.83     14158
weighted avg       0.83      0.83      0.83     14158


Confusion Matrix:
 [[5687 1347]
 [1057 6067]]
```



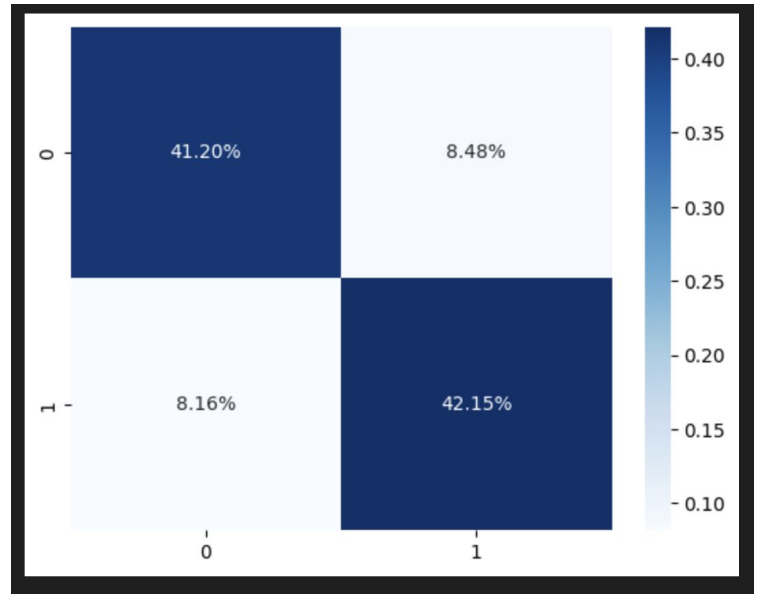
# 4.3 Model Comparison

## 4.3.10) Naive Bayes with Tf-IDF Vectorized Values

```
tm_nb_model = GaussianNB()

tm_nb_model.fit(X_train_tfidf_matrix.toarray(), y_train)
tm_nb_predictions = tm_nb_model.predict(X_test_tfidf_matrix.toarray())

evaluateModel(tm_nb_predictions)

pickle.dump(tm_nb_model, open(save_models_path+"_"+version+"_"+model_name,'wb'))
```

[65] ✓ 7.6s Python

```
Accuracy: 0.8335216838536517

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.83      0.83      7034
           1       0.83      0.84      0.84      7124

    accuracy                           0.83     14158
   macro avg       0.83      0.83      0.83     14158
weighted avg       0.83      0.83      0.83     14158


Confusion Matrix:
 [[5833 1201]
 [1156 5968]]
```



# 4.4 Model Evaluations Metrics

## Count Vectorized

| Models | Vectorizer Type | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|---|
| Logistic Regression | Count Vectorizer | 94.81% | 95% | 94% | 95% |
| Naive Bayes | Count Vectorizer | 83.02% | 83% | 83% | 83% |
| Decision Tree | Count Vectorizer | 93.26% | 93% | 93% | 93% |
| Random Forest | Count Vectorizer | 95.46% | 95% | 95% | 95% |
| Support Vector Machines | Count Vectorizer | 93.75% | 94% | 94% | 94% |

# 4.4 Model Evaluation Metrics

## Tf-IDF Vectorized

| Models | Vectorizer Type | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|---|
| Logistic Regression | TF-IDF Vectorizer | 94.44% | 94% | 94% | 94% |
| Naive Bayes | TF-IDF Vectorizer | 83.35% | 83% | 83% | 83% |
| Decision Tree | TF-IDF Vectorizer | 64.40% | 64% | 64% | 64% |
| Random Forest | TF-IDF Vectorizer | 95.31% | 95% | 95% | 95% |
| Support Vector Machines | TF-IDF Vectorizer | 95.45% | 95% | 95% | 95% |

# Thank you