Fake News Detection: A Machine Learning Approach

Master of Science in Statistics

By

Arnav Banerjee

Registration No: 22360401008

Session: 2022-2024

Under the guidance of

Dr. Bhagwati Devi

Department of Statistics
Central University of Jharkhand
Ranchi - 835222, Jharkhand, India

झारखण्ड केन्द्रीय विश्वविद्यालय
Central University of Jharkhand
(Established by an Act of Parliament of India,2009)

CERTIFICATE OF APPROVAL

It is hereby informed that the project entitled **Fake News Detection: A Machine Learning Approach** has been submitted by **Arnav Banerjee**, Reg.no. 22360401008 for the partial fulfillment of the degree of M.Sc. in Statistics from Department of Statistics, Central University of Jharkhand, Manatu, Ranchi.

[illegible] 31/5/24

SUPERVISOR

Dr. Bhagwati Devi

(Assistant Professor)

Department of Statistics

Central University of Jharkhand

[illegible] 05/06/2024

Head of Department

Dr.Kunja Bihari Panda

(Professor)

Department of Statistics

Central University of Jharkhand

Head
Department of Statistics
Central University of Jharkhand
Cheri-Manatu, Kanke, Ranchi

झारखण्ड केन्द्रीय विश्वविद्यालय
Central University of Jharkhand
(Established by an Act of Parliament of India,2009)

DECLARATION

I declare that:

- This project contains my original work, carried out under the supervision of my advisor.
- This work has not been submitted to any other university or institute for a degree or diploma.
- Whenever materials (data, theoretical analysis, statistical analysis, and text) from other sources have been used, proper credit has been given by citing them in the text and listing them in the references.

31/05/2024
Date:

Banerjee
Signature of the Student

झारखण्ड केन्द्रीय विश्वविद्यालय
Central University of Jharkhand
(Established by an Act of Parliament of India,2009)

ACKNOWLEDGMENT

The completion of this project was made possible through the support and guidance of numerous individuals. I am deeply grateful for the assistance I have received throughout this journey. I would like to express my profound appreciation to my supervisor, **Dr. Bhagwati Devi**, Assistant Professor, Department of Statistics, Central University of Jharkhand, for her crucial contributions to my project titled **Fake News Detection: A Machine Learning Approach**.

I am also greatly indebted to my mentor, **Dr. Kunja Bihari Panda**, Professor, Department of Statistics, Central University of Jharkhand, for his invaluable advice and insights throughout the year. His dedication and effort have been instrumental in the success of this project.

Finally, I extend my sincere thanks to all my classmates for their continuous support and assistance.

Banerjee

Arnav Banerjee

Abstract

In recent years, the spread of digital media and social networks has led to an easy access of news and knowledge for everyone. While this has led to gaining access to news and knowledge, it also gave rise to the rapid spread of misinformation and fake news. Fake news is misleading information presented as news with the intention of deceiving the public. This phenomenon poses significant threats to social stability and public trust in actually true news sources.

This project focuses on classifying the news articles into fake and real categories using Machine Learning algorithms. The primary objective is to build a classification model that can accurately differentiate fake news from real news articles. The Machine Learning algorithms used are Logistic Regression, Decision tree, Random Forest, Support Vector Machines(SVM), K-Nearest Neighbour(KNN) and Naive Bayes. These models were trained and evaluated using standard performance metrics such as accuracy, precision and recall.

The performance of each model was compared and the model with highest effectiveness is used. The results indicate that the chosen machine learning model can classify the news articles with high accuracy. The analysis provides insights onto the strengths and weaknesses of each model. The project concludes with recommendations for future work to improve classification accuracy and extend the application of the developed models to real-world scenarios.

Contents

Chapter 1

Introduction

1.1 Background

The rapid spread of digital media has significantly changed the norms of society. With the introduction social media, news and information can be shared immediately and also to global audience. While this normalized the access to information, it also led to spread of false information and fake news. This further led to emergence of a major problem in todays information network. The impact of fake news is far reaching, influencing public openion, promoting social unrest and creating distrust in credible news sources. The spread of fake news can have serious consequences such as it can affect election outcome, damage reputation, spreading false health information, etc. Seeing the potential harm caused by fake news, there is a need to develop an effective method to classify the fake news from the credible ones and suppress its spread

1.2 Problem Statement

We face several problems while developing a reliable fake news detection system.

Data quality:

There is a need for high quality, labelled datasets with accurate representation of both real and fake news articles. It should ensure diversity and reliability while training the dataset.

Feature extraction:

Identifying and extracting relevant features from data text is essential to distinguish fake news from credible news. It should involves techniques to distinguish the complexities of languages used in news articles.

Model accuracy:

We should build a Machine Learning model to achieve high accuracy inclassifying the articles. Both flase positive and false negative must be minimized.

Adaptability:

The model should be adaptable and evolving in nature. As fake news creaters divise new strategies, the detection model should keep up with these changes.

1.3 Objectives

The primary objective of my project is to develop a efdective machine learning based classification model for detecting fake news. This objective is divided into following parts.

- Collecting a labelled dataset from a reliable source. I got my dataset from kaggle.com.
- Developing and training multiple Machine learning models. I trained Logistic Regression, Decision tree, Random forest, KNN, SVM and Naive Bayes model to classify news aricles.
- Evaluate the performance of the developed models using standard metrics such as accuracy, precision and recall. Then compare the different models to identify the most effective approach for fake news detection.
- Analyze the result to understand the strength and limitations of each model. Interpret the findings to provide insight into the most effective techniques to dect fake news.

By achieving these objectives, the project aims to advance the field of automated fake news detection providing an effective solution to reduce the impact of fake news in this digital era.

Chapter 2

Review of Literature

Introduction

The rapid growth of digital media has intensified the spread of fake news, making it a significant area of concern and research. With the increasing popularity of social media, more and more people consume news from social media instead of traditional news media. However, social media has also been used to spread fake news, which has strong negative impacts on individual users and broader society.[3] Various approaches have been proposed to address this issue, leveraging both traditional techniques and advanced machine learning algorithms. This literature review explores the existing research on fake news detection, focusing on data preprocessing methods, feature extraction techniques, and machine learning models.

Data Preprocessing

Data preprocessing is a fundamental stage of ML method and has large impact on the accuracy of ML methods. [1] The performance of machine learning algorithms and deep learning algorithms depends on the pre-processing and size of the dataset. Nonetheless, in some cases, machine learning models fail to extract some implicit features or aspects of the text. In situations where the dataset is vast, the deep learning approach performs better than machine learning. [5]

Feature Extraction

Feature selection improves knowledge of the process under consideration, as it points out the features that mostly affect the considered phenomenon .[6] . The extraction method can have a significant impact on the results of multi-label classification. The best choice of extraction method depends on the what the multi-label classifications are to be used for. [4]

Machine Learning Models

The key question when dealing with ML classification is not whether a learning algorithm is superior to others, but under which conditions a particular method can significantly outperform others on a given application problem. [7]. After a better understanding of the strengths and limitations of each method, the possibility of integrating two or more algorithms together to solve a problem should be investigated. The objective is to utilize the strengthes of one method to complement the weaknesses of another. If we are only interested in the best possible classification accuracy, it might be difficult or impossible to find a single classifier that performs as well as a good ensemble of classifiers [8]

Challenges

Despite significant advancements, fake news detection remains a challenging task. One major issue is the evolving nature of fake news, which requires models to adapt continuously. On the bright side, automatic fake news detection could be used by fact checkers as an auxiliary tool for identifying content that is more likely to be fake. [2] With ever increasing fake content, the research on the fake news detection is also making progress and various approaches from vast domains have been implemented ranging from artificial intelligence to linguistic and knowledge engineering but no ideal methodology has been devised yet that can accurately classify real news and fake news [9]

Conclusion

The literature on fake news detection underscores the importance of data preprocessing, feature extraction, and the selection of appropriate machine learning models. While traditional models like Naive Bayes and SVM have demonstrated effectiveness, deep learning approaches offer promising advancements. However, continuous adaptation to new types of fake news and the development of high-quality datasets remain critical areas for future research. By addressing these challenges, researchers can develop more robust and accurate fake news detection systems, contributing to a more informed and trustworthy digital information network.

Chapter 3

Research Methodology

3.1 Dataset

The Fake News Detection Data of was collected from www.kaggle.com . The data set consists of total 4500 rows and 6 columns which include 1483 real news and 3018 fake news. Real news are labelled as “1” and Fake news are labelled as “0”. There are 5 Independent Variables and 1 Dependent Variable representing “1” and “0” for real and fake news respectively.

3.2 Data Processing

This phase inspects various preprocessing methods to provide us with the most appropriate and cleaned data for further use in detection techniques using machine learning algorithms.

3.2.1 Detection Of Missing Values

In this method, we check if the data contains any missing values. If missing values are present, we estimate them using one of the following strategies:

- Mean: Used when the data is significantly affected by outliers.
- Median: Used when the data is less affected by outliers compared to the mean.
- Mode: Used for qualitative data.

Several techniques can be employed to detect and address missing values, such as examining individual rows or columns, among others.

Analysis revealed that there is no missing values in this dataset

3.2.2 Detection Of Outliers

Outliers are those characteristics that holds certain different anomalies behaviour from other attributes or features. We can detect the Outliers by using the BOXPLOT visualization. In BOXPLOT Representation , the data points lying above 100%-percentile or below 0%-percentile represents the Outliers. There are other methods for detecting Outliers like Scatter Plot ,Z-Score, etc.

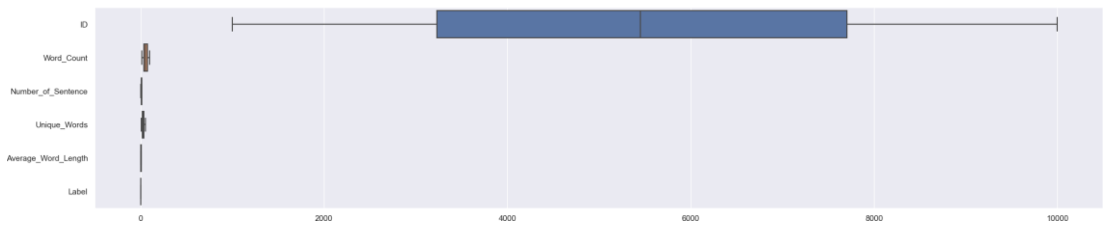


Figure 3.1: Boxplot representation of the columns

3.2.3 Detection of Correlation among the features using Correlation Heatmap

Using Correlation Heatmap , We try to find the relationship among the features. The relationship implies to be either Strong , Weak or No Relation between the features. Before using the Correlation Heatmap, we are provided with the Correlation Matrix which consists of all the correlated values among the features that ranges according to the dataset. Correlation ranges between -1 to 1. The more closer is the Correlated value to -1 or 1, more the strong and positive or negtive correlation to be. No Correlation holds when the Correlated value between any two feature becomes zero implies the two variables to be independent.

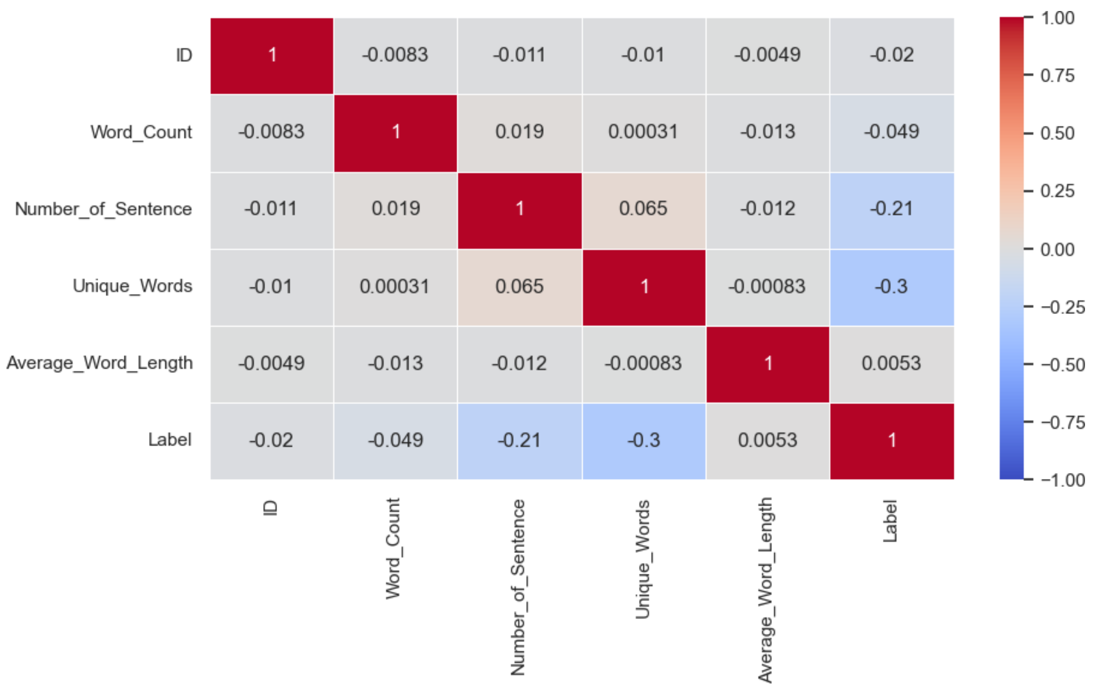
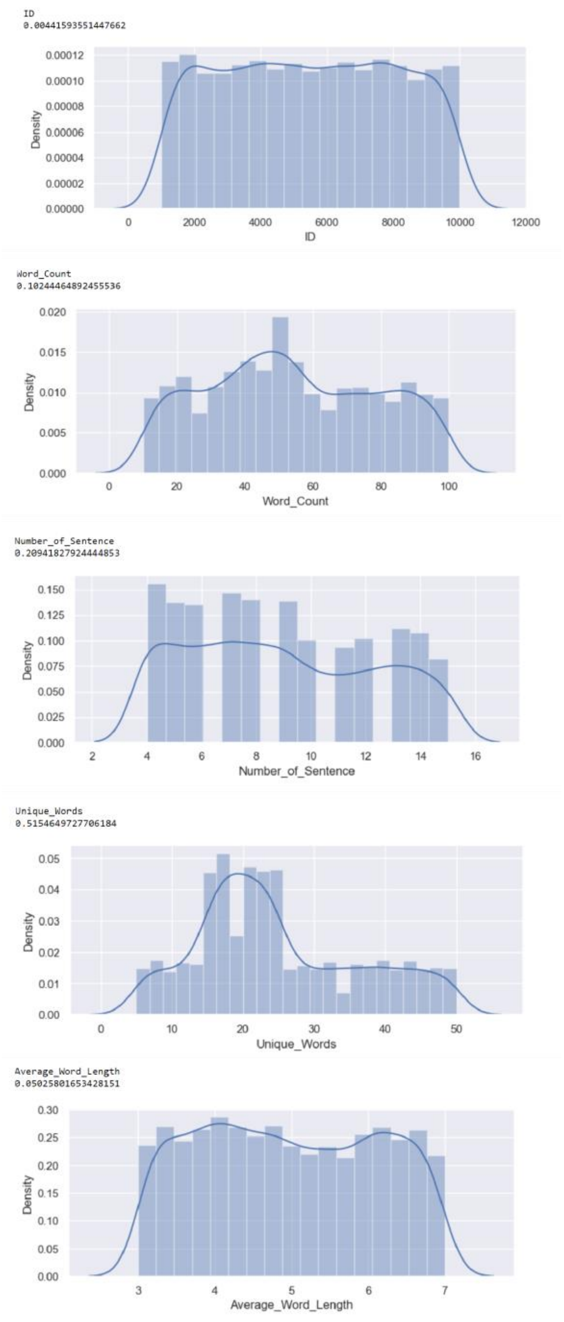


Figure 3.2: Correlation Heatmap

3.2.4 Detection of Skewness in the Dataset

Skewness is a measure of the asymmetry of a distribution. A distribution can have right (or positive), left (or negative), or zero skewness. By the help of Skewness , one can check the dataset to be positively or negatively skewed as The larger the skew, the greater are the proportion of correlations that are negative . In this dataset , most of the features are positively skewed. If the dataset are badly skewed, bimodal, or otherwise violate the assumptions of the general linear model, then one can better use Spearman’s rho (rank-order correlation) or Kendall’s tau.



```
ID
0.00441593551447662
```

```
Word_Count
0.10244464892455536
```

```
Number_of_Sentence
0.20941827924444853
```

```
Unique_Words
0.5154649727706184
```

```
Average_Word_Length
0.05025801653428151
```

Figure 3.3: Skewness of all the columns

3.3 Feature Extraction

Feature selection is a way of selecting the subset of the most relevant features from the original features set by removing the redundant, irrelevant, or noisy features. While developing the machine learning model, only a few variables in the dataset are useful for building the model, and the rest features are either redundant or irrelevant. If we input the dataset with all these redundant and irrelevant features, it may negatively impact and reduce the overall performance and accuracy of the model. Hence it is very important to identify and select the most appropriate features from the data and remove the irrelevant or less important features, which is done with the help of feature selection in machine learning. Below are some benefits of using feature selection in machine learning. It helps in avoiding the curse of dimensionality. It helps in the simplification of the model so that it can be easily interpreted by the researchers. It reduces the training time. It reduces over-fitting hence enhance the generalization. Two methods of Feature Selection are used :-

3.3.1 Mutual Information

It checks the dependency between the two variables and if the two variables are independent then there correlation is cosiderered as zero. Higher the values of the features , more is the Dependency between the variables. Mutual Information always gives either Positive Value for Dependency or Zero Value for Independency.

3.3.2 Feature Importance

This Technique gives the Scores of all features individually, The Higher the Score More Relevant it is. Feature Importance refers to techniques that calculate a score for all the input features for a given model — the scores simply represent the “importance” of each feature. A higher score means that the specific feature will have a larger effect on the model that is being used to predict a certain variable. Feature importance is calculated as the decrease in node impurity weighted by the probability of reaching that node. The node probability can be calculated by the number of samples that reach the node, divided by the total number of samples. The higher the value the more important the feature. in this project, unique words are found to be most important feature with respect to label. Whereas ID is found to be least important feature.

3.4 Machine Learning Models

Six machine learning classification model Logistic, Decision Tree, Random Forest and Support Vector Machine, K-Nearest Neighbour, Naïve Bayes Classifier has been selected to detect the fake news.

3.4.1 Logistic Regression Classifier

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes. Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

3.4.2 Decision Tree Classifier

One of the most widely used algorithm in machine learning technology. Decision tree algorithm is easy to understand and also easy to implement. Decision tree begins its work by choosing best splitter from the available attributes for classification which is considered as a root of the tree. Algorithm continues to build tree until it finds the leaf node. Decision tree creates training model which is used to predict target value or class in tree representation each internal node of the tree belongs to attribute and each leaf node of the tree belongs to class label. In decision tree algorithm, Gini index and Information gain methods are used to calculate these nodes.

3.4.3 Random Forest Classifier

Random Forest algorithm is one of the most powerful algorithms in machine learning technology and it is based on concept of decision tree algorithm. Random Forest algorithm creates the forest with number of decision trees. High number of tree gives high detection accuracy. Creation of trees are based on bootstrap method. In bootstrap method features and samples of dataset are randomly selected with replacement to construct single tree. Among randomly selected features, Random Forest algorithm will choose best splitter for the classification and like decision tree algorithm; Random Forest algorithm also uses Gini index and Information gain methods to find the best splitter. This process will get continue until Random Forest creates n number of trees. Each tree in forest predicts the target value and then algorithm will calculate the votes for each predicted target. Finally Random Forest algorithm considers high voted predicted target as a final prediction.

3.4.4 Support Vector Machine

Support Vector Machine is another powerful algorithm in machine learning technology. In Support Vector Machine algorithm each data item is plotted as a point in n-dimensional space and Support Vector Machine algorithm constructs separating line for classification of two classes, this separating line is well known as hyperplane. Support Vector Machine seeks for the closest points called as support vectors and once it finds the closest point it draws a line connecting to them. Support Vector Machine then construct separating line which bisects and perpendicular to the connecting line. In order to classify data perfectly the margin should be maximum. Here the margin is a distance between hyperplane and support vectors. In real scenario it is not possible to separate complex and non linear data, to solve this problem Support Vector Machine uses kernel trick which transforms lower dimensional space to higher dimensional space.

3.4.5 K-Nearest Neighbour

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. It assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. It stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. It can be used for Regression as well as for Classification but mostly it is used for the Classification problems. K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data. It is simple to implement. It is robust to the noisy training data KNN Algorithm can be more effective if the training data is large.

3.4.6 Naïve Bayes Classifier

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles. The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:
Naïve: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of colour, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem.

3.5 Evaluation Metrics

Evaluation metrics are important for assessing the performance of machine learning models. In the context of fake news detection, these metrics help determine how well the models classify news articles as either fake or real. We use the following componentsin in the evaluation metrices:

- TP = True Positives
- TN = True Negatives
- FP = False Positives
- FN = False Negatives

The following are common evaluation metrics used in classification problems:

3.5.1 Accuracy

Accuracy is the ratio of correctly predicted instances to the total instances. It is a common metric but can be misleading if the classes are imbalanced.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

3.5.2 Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positives. It is a measure of the accuracy of the positive predictions.

$$Precision = \frac{TP}{TP + FP}$$

3.5.3 Recall

Recall, also known as Sensitivity or True Positive Rate, is the ratio of correctly predicted positive observations to all observations in the actual class.

$$Recall = \frac{TP}{TP + FN}$$

3.5.4 F1-Score

The F1-Score is the harmonic mean of precision and recall. It provides a single metric that balances both concerns, especially useful when you need to find an optimal balance between precision and recall.

$$F1Score = 2 \times \frac{Precision \times recall}{Precision + Recall}$$

3.5.5 Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. It allows visualization of the performance of an algorithm by showing the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions..

TP	FP
FN	TN

3.6 Tools and Libraries

For this project I employed various tools and libraries to complete the different tasks, including data preprocessing, model development, and evaluation. Below are the key tools and libraries used in the implementation:

3.6.1 Programming Language

Python: Python serves as the primary language for implementing the fake news detection system due to its extensive libraries and frameworks for machine learning and natural language processing tasks.

3.6.2 Libraries

• **Numpy:** NumPy is used for numerical computations and array operations, providing efficient data structures for handling large datasets.

• **pandas:** pandas is utilized for data manipulation and analysis, facilitating tasks such as data loading, manipulation, and transformation.

• **scikit-learn:** scikit-learn is a machine learning library that offers a wide array of tools for classification, regression, clustering, and model evaluation. It is utilized for model development, training, and evaluation in the fake news detection system.

• **matplotlib and seaborn:** These libraries are used for data visualization, enabling the creation of plots, charts, and visualizations to analyze and interpret the results effectively.

3.6.3 Development Environment:

Jupyter Notebook serves as an interactive development environment for experimentation, and documentation of the fake news detection system. It offers features such as code execution, visualization, and text integration in a single document.

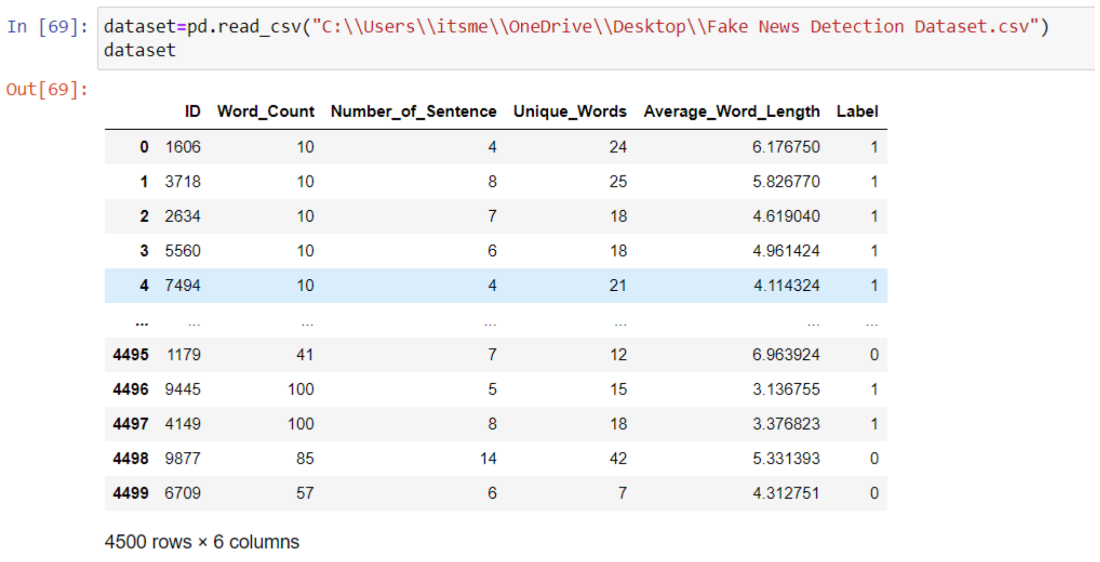
Chapter 4

Implementation

In this chapter, the practical implementation of the fake news detection system is detailed. It encompasses the application of the proposed methodology, including model training, evaluation, and application.

4.1 Data Loading

Below is a Python code snippet demonstrating the data loading process in our fake news detection system. In this example, we use the pandas library to load a preprocessed dataset from a CSV file.



```
In [69]: dataset=pd.read_csv("C:\\Users\\itsme\\OneDrive\\Desktop\\Fake News Detection Dataset.csv")
         dataset
Out[69]:
```

	ID	Word_Count	Number_of_Sentence	Unique_Words	Average_Word_Length	Label
0	1606	10	4	24	6.176750	1
1	3718	10	8	25	5.826770	1
2	2634	10	7	18	4.619040	1
3	5560	10	6	18	4.961424	1
4	7494	10	4	21	4.114324	1
...	...	...	...	...	...	...
4495	1179	41	7	12	6.963924	0
4496	9445	100	5	15	3.136755	1
4497	4149	100	8	18	3.376823	1
4498	9877	85	14	42	5.331393	0
4499	6709	57	6	7	4.312751	0

4500 rows × 6 columns

Figure 4.1: Loading the dataset

4.2 Feature Engineering

Although the data is preprocessed, additional feature engineering may be conducted to enhance model performance. This includes removing the columns whose mutual information and feature importance is least with respect to "Label".

Below are the representations of mutual information and feature importance with respect to the label

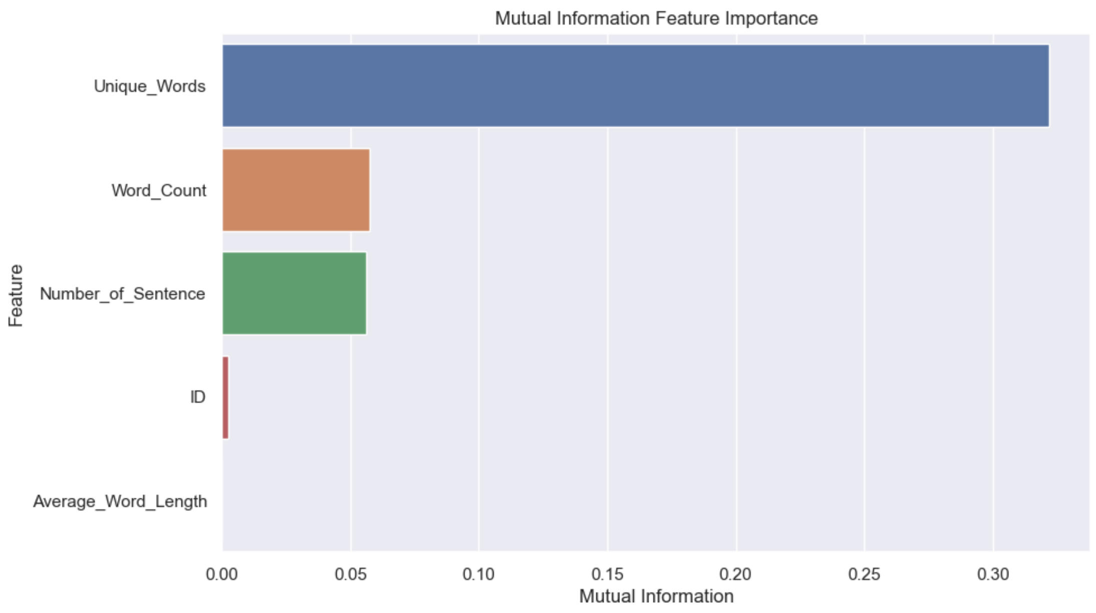


Figure 4.2: Mutual information about each column w.r.t. Label

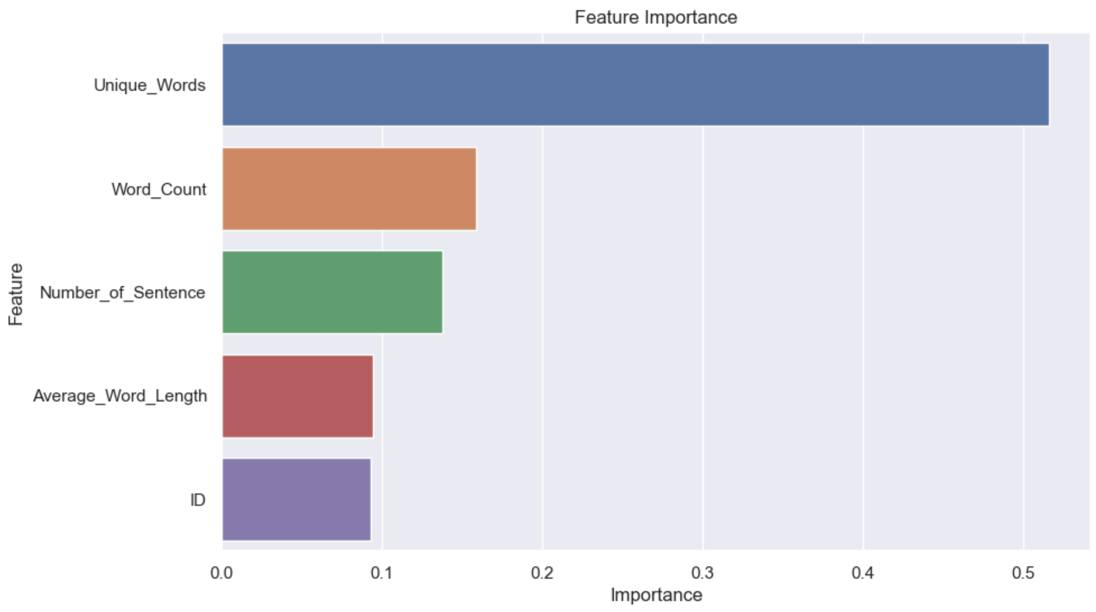
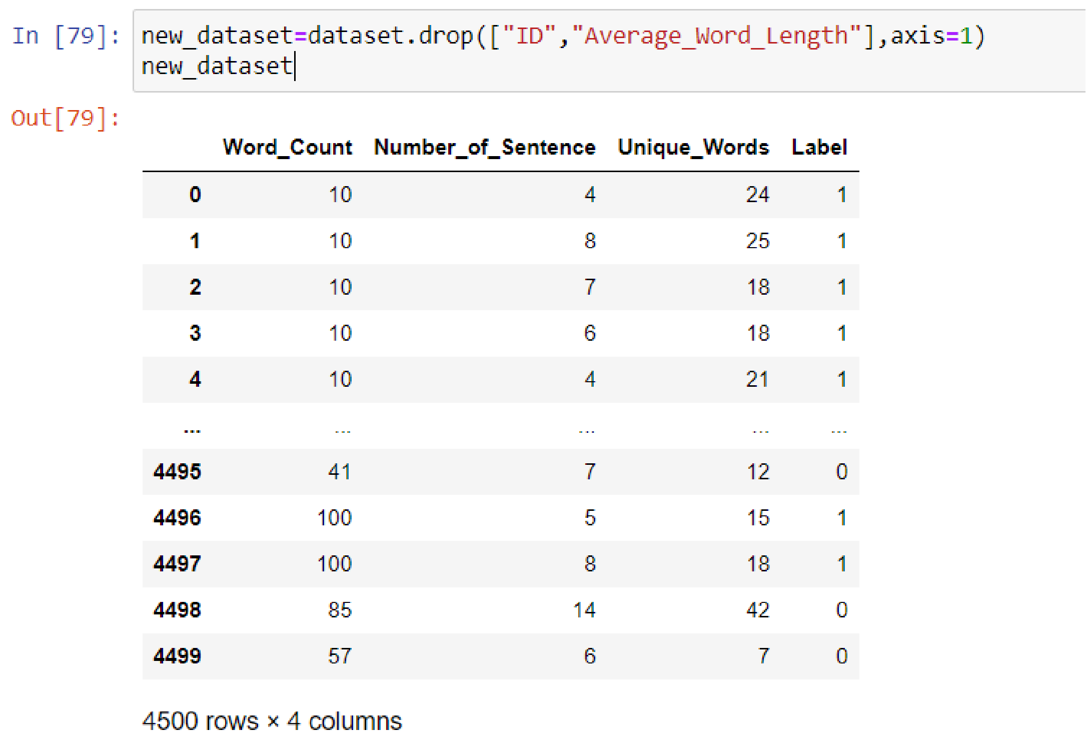


Figure 4.3: Features importance with respect to Label

Analyzing the figures, Figure 4.2 and Figure 4.3 we can see that the features "ID" and "Average_Word_Length" provide the least amount of information and are also the least important features. So proceeding with our analysis we will discard these two features and work with other feaures.



```
In [79]: new_dataset=dataset.drop(["ID","Average_Word_Length"],axis=1)
         new_dataset
```

Out[79]:

	Word_Count	Number_of_Sentence	Unique_Words	Label
0	10	4	24	1
1	10	8	25	1
2	10	7	18	1
3	10	6	18	1
4	10	4	21	1
...	...	...	...	...
4495	41	7	12	0
4496	100	5	15	1
4497	100	8	18	1
4498	85	14	42	0
4499	57	6	7	0

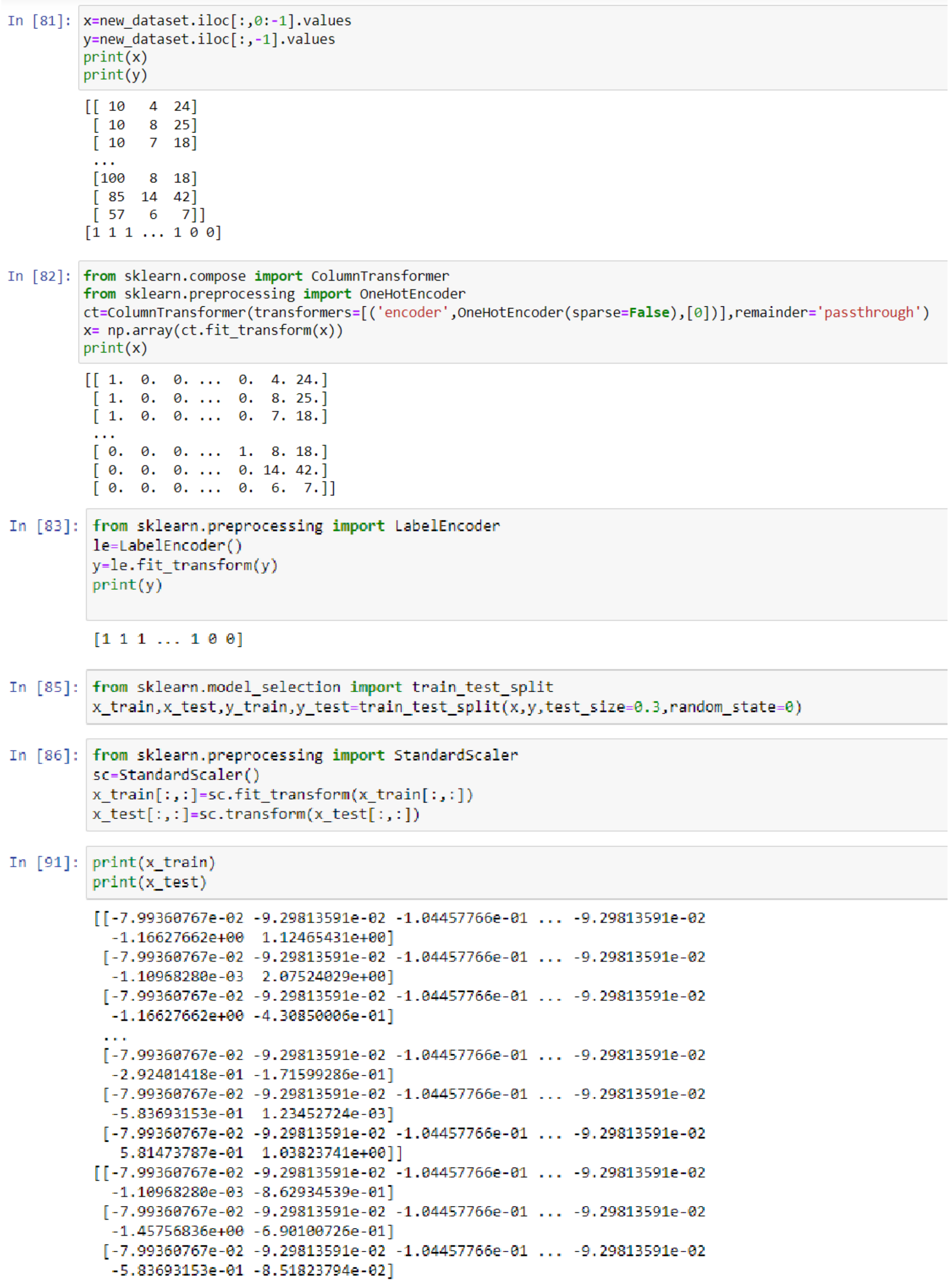
4500 rows × 4 columns

Figure 4.4: New dataset after removing ID and Average_Word_Length

4.3 Model Selection and Training

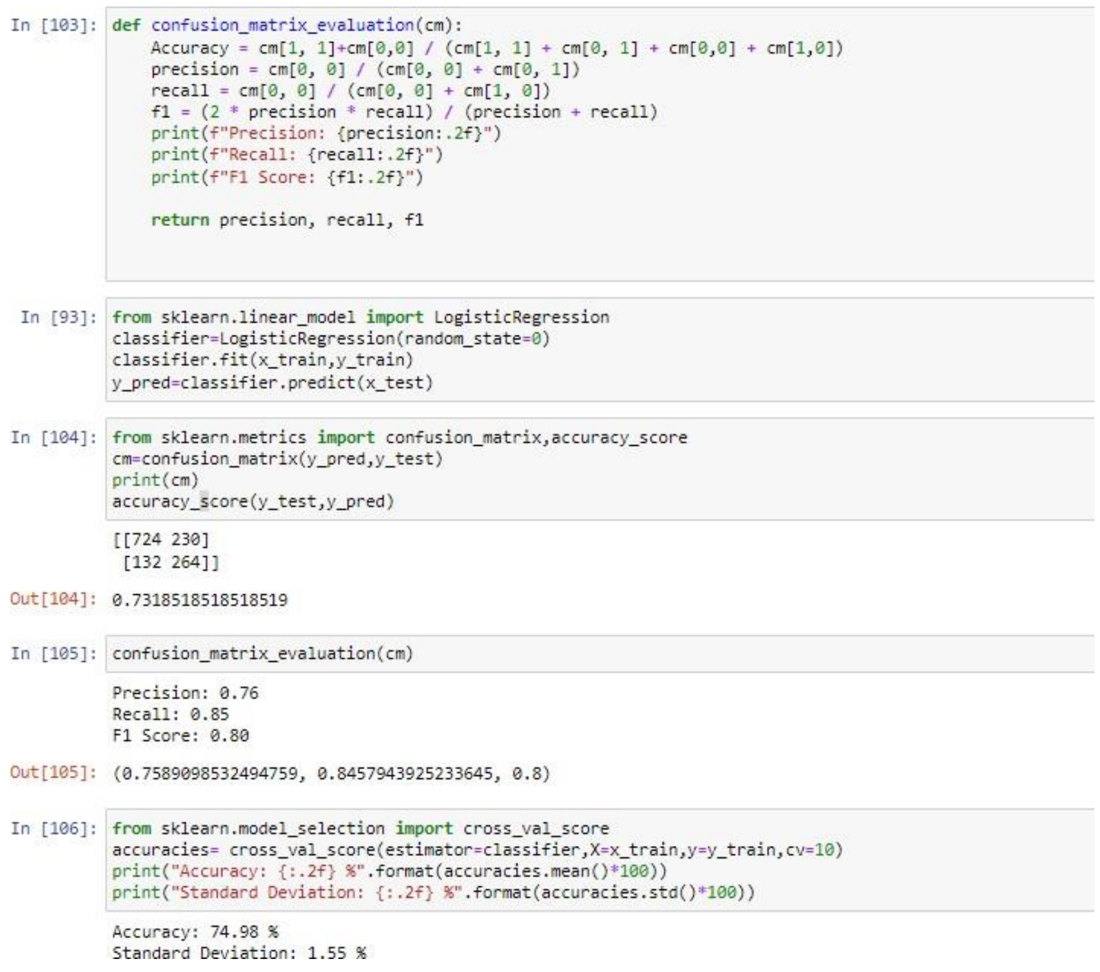
We choose appropriate models based on the nature of the problem, the available data, and the desired performance metrics. In this case we are working with Logistic regression, Decision Tree, Random Forest, Support Vector Machine, K-Nearest Neighbour and Naïve Bayes Classifier. Scikit-learn tool has been used to import Machine learning algorithms. Dataset is divided into training set and testing set in 70:30 ratio respectively. I provided the codes for training these models below.

Figure 4.5: Splitting the Dataset into Training and Testing Sets for Features and Labels



```
In [81]: x=new_dataset.iloc[:,0:-1].values
         y=new_dataset.iloc[:,-1].values
         print(x)
         print(y)

         [[ 10   4  24]
          [ 10   8  25]
          [ 10   7  18]
          ...
          [100   8  18]
          [ 85  14  42]
          [ 57   6   7]]
         [1 1 1 ... 1 0 0]

In [82]: from sklearn.compose import ColumnTransformer
         from sklearn.preprocessing import OneHotEncoder
         ct=ColumnTransformer(transformers=[('encoder',OneHotEncoder(sparse=False),[0])],remainder='passthrough')
         x= np.array(ct.fit_transform(x))
         print(x)

         [[ 1.  0.  0. ...  0.  4. 24.]
          [ 1.  0.  0. ...  0.  8. 25.]
          [ 1.  0.  0. ...  0.  7. 18.]
          ...
          [ 0.  0.  0. ...  1.  8. 18.]
          [ 0.  0.  0. ...  0. 14. 42.]
          [ 0.  0.  0. ...  0.  6.  7.]]

In [83]: from sklearn.preprocessing import LabelEncoder
         le=LabelEncoder()
         y=le.fit_transform(y)
         print(y)

         [1 1 1 ... 1 0 0]

In [85]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)

In [86]: from sklearn.preprocessing import StandardScaler
         sc=StandardScaler()
         x_train[:,:]=sc.fit_transform(x_train[:,:])
         x_test[:,:]=sc.transform(x_test[:,:])

In [91]: print(x_train)
         print(x_test)

         [[-7.99360767e-02 -9.29813591e-02 -1.04457766e-01 ... -9.29813591e-02
           -1.16627662e+00  1.12465431e+00]
          [-7.99360767e-02 -9.29813591e-02 -1.04457766e-01 ... -9.29813591e-02
           -1.10968280e-03  2.07524029e+00]
          [-7.99360767e-02 -9.29813591e-02 -1.04457766e-01 ... -9.29813591e-02
           -1.16627662e+00 -4.30850006e-01]
          ...
          [-7.99360767e-02 -9.29813591e-02 -1.04457766e-01 ... -9.29813591e-02
           -2.92401418e-01 -1.71599286e-01]
          [-7.99360767e-02 -9.29813591e-02 -1.04457766e-01 ... -9.29813591e-02
           -5.83693153e-01  1.23452724e-03]
          [-7.99360767e-02 -9.29813591e-02 -1.04457766e-01 ... -9.29813591e-02
            5.81473787e-01  1.03823741e+00]]
         [[-7.99360767e-02 -9.29813591e-02 -1.04457766e-01 ... -9.29813591e-02
           -1.10968280e-03 -8.62934539e-01]
          [-7.99360767e-02 -9.29813591e-02 -1.04457766e-01 ... -9.29813591e-02
           -1.45756836e+00 -6.90100726e-01]
          [-7.99360767e-02 -9.29813591e-02 -1.04457766e-01 ... -9.29813591e-02
           -5.83693153e-01 -8.51823794e-02]
```

Figure 4.6: Training a Logistic Regression Model on the Training Dataset



```
In [103]: def confusion_matrix_evaluation(cm):
              Accuracy = cm[1, 1]+cm[0,0] / (cm[1, 1] + cm[0, 1] + cm[0,0] + cm[1,0])
              precision = cm[0, 0] / (cm[0, 0] + cm[0, 1])
              recall = cm[0, 0] / (cm[0, 0] + cm[1, 0])
              f1 = (2 * precision * recall) / (precision + recall)
              print(f"Precision: {precision:.2f}")
              print(f"Recall: {recall:.2f}")
              print(f"F1 Score: {f1:.2f}")

              return precision, recall, f1

 In [93]: from sklearn.linear_model import LogisticRegression
          classifier=LogisticRegression(random_state=0)
          classifier.fit(x_train,y_train)
          y_pred=classifier.predict(x_test)

In [104]: from sklearn.metrics import confusion_matrix,accuracy_score
          cm=confusion_matrix(y_pred,y_test)
          print(cm)
          accuracy_score(y_test,y_pred)

          [[724 230]
           [132 264]]

Out[104]: 0.7318518518518519

In [105]: confusion_matrix_evaluation(cm)

          Precision: 0.76
          Recall: 0.85
          F1 Score: 0.80

Out[105]: (0.7589098532494759, 0.8457943925233645, 0.8)

In [106]: from sklearn.model_selection import cross_val_score
          accuracies= cross_val_score(estimator=classifier,X=x_train,y=y_train,cv=10)
          print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
          print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))

          Accuracy: 74.98 %
          Standard Deviation: 1.55 %
```

Precision: 0.76

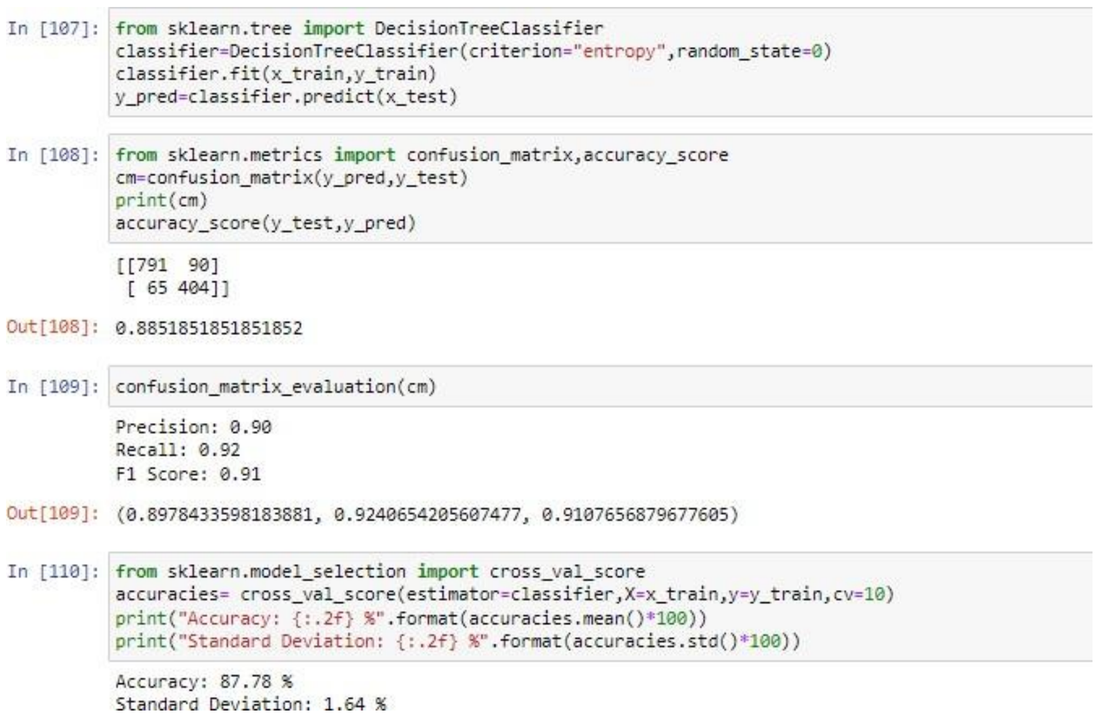
Recall: 0.85

F1 Score: 0.80

Accuracy: 74.98 %

Standard Deviation: 1.55

Figure 4.7: Training a Decision Tree Model on the Training Dataset



```
In [107]: from sklearn.tree import DecisionTreeClassifier
          classifier=DecisionTreeClassifier(criterion="entropy",random_state=0)
          classifier.fit(x_train,y_train)
          y_pred=classifier.predict(x_test)

In [108]: from sklearn.metrics import confusion_matrix,accuracy_score
          cm=confusion_matrix(y_pred,y_test)
          print(cm)
          accuracy_score(y_test,y_pred)

          [[791  90]
           [ 65 404]]

Out[108]: 0.8851851851851852

In [109]: confusion_matrix_evaluation(cm)

          Precision: 0.90
          Recall: 0.92
          F1 Score: 0.91

Out[109]: (0.8978433598183881, 0.9240654205607477, 0.9107656879677605)

In [110]: from sklearn.model_selection import cross_val_score
          accuracies= cross_val_score(estimator=classifier,X=x_train,y=y_train,cv=10)
          print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
          print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))

          Accuracy: 87.78 %
          Standard Deviation: 1.64 %
```

Precision: 0.90

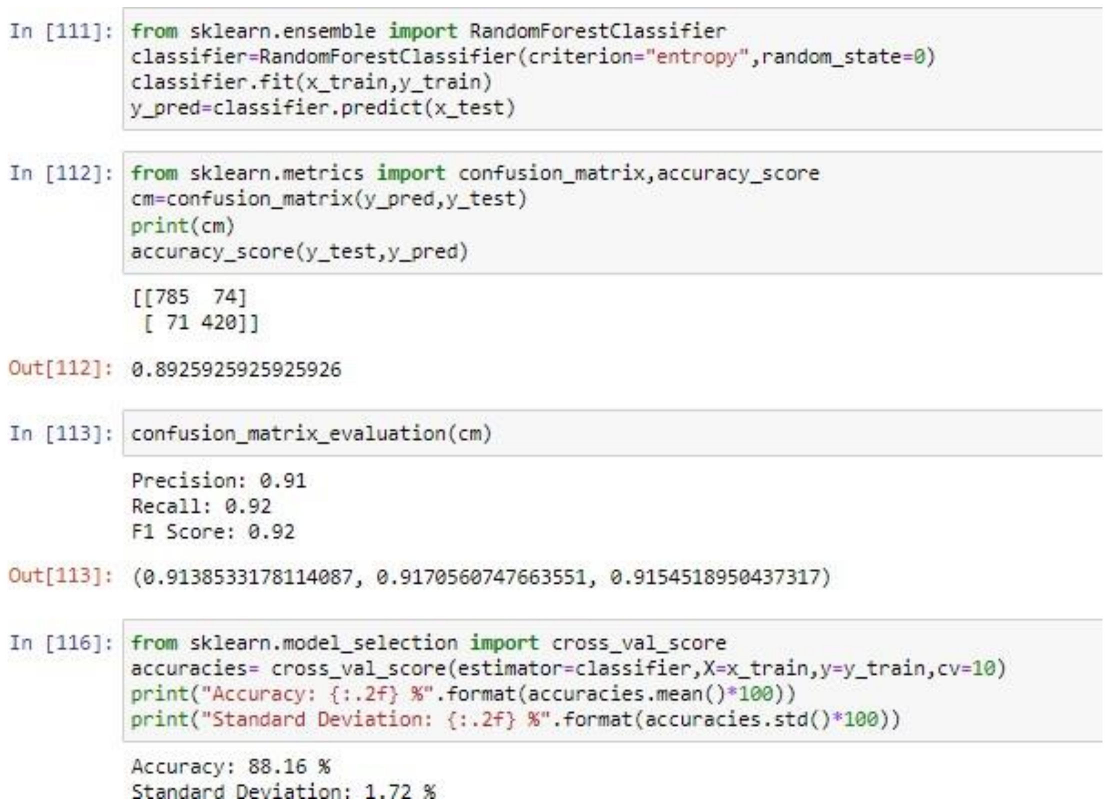
Recall: 0.92

F1 Score: 0.91

Accuracy: 87.78 %

Standard Deviation: 1.64

Figure 4.8: Training a Random Forest Model on the Training Dataset



```
In [111]: from sklearn.ensemble import RandomForestClassifier
          classifier=RandomForestClassifier(criterion="entropy",random_state=0)
          classifier.fit(x_train,y_train)
          y_pred=classifier.predict(x_test)

In [112]: from sklearn.metrics import confusion_matrix,accuracy_score
          cm=confusion_matrix(y_pred,y_test)
          print(cm)
          accuracy_score(y_test,y_pred)

          [[785  74]
           [ 71 420]]

Out[112]: 0.8925925925925926

In [113]: confusion_matrix_evaluation(cm)

          Precision: 0.91
          Recall: 0.92
          F1 Score: 0.92

Out[113]: (0.9138533178114087, 0.9170560747663551, 0.9154518950437317)

In [116]: from sklearn.model_selection import cross_val_score
          accuracies= cross_val_score(estimator=classifier,X=x_train,y=y_train,cv=10)
          print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
          print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))

          Accuracy: 88.16 %
          Standard Deviation: 1.72 %
```

Precision: 0.91

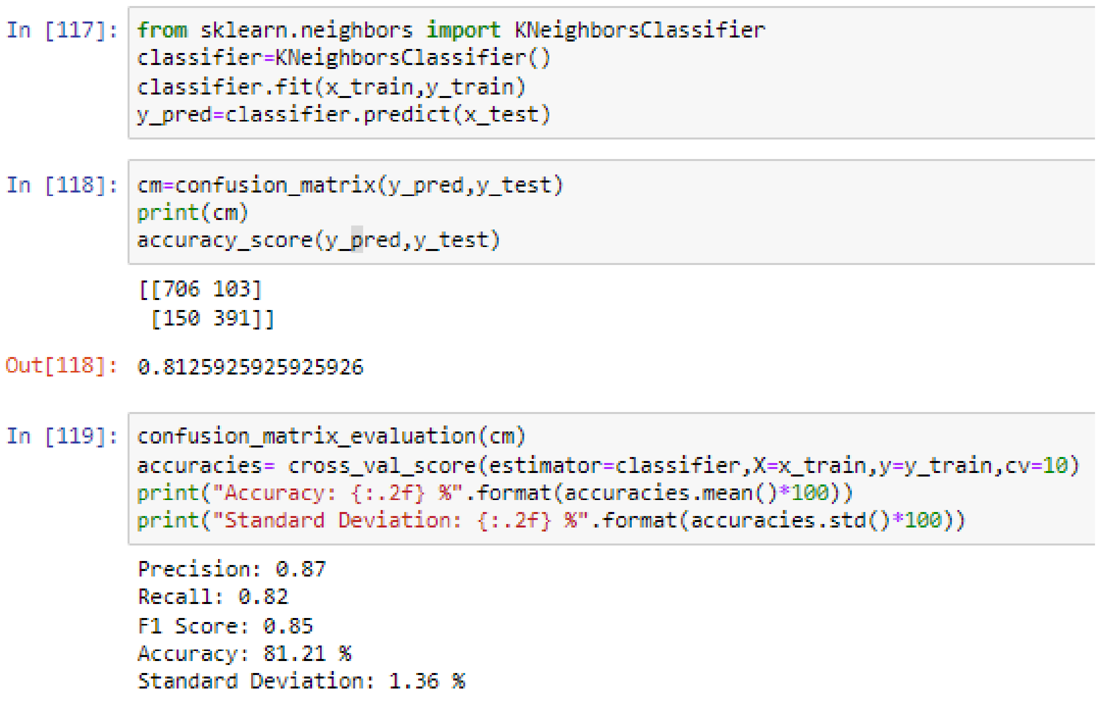
Recall: 0.92

F1 Score: 0.92

Accuracy: 88.16 %

Standard Deviation: 1.72

Figure 4.9: Training a KNN Model on the Training Dataset



```
In [117]: from sklearn.neighbors import KNeighborsClassifier
          classifier=KNeighborsClassifier()
          classifier.fit(x_train,y_train)
          y_pred=classifier.predict(x_test)

In [118]: cm=confusion_matrix(y_pred,y_test)
          print(cm)
          accuracy_score(y_pred,y_test)

          [[706 103]
           [150 391]]

Out[118]: 0.8125925925925926

In [119]: confusion_matrix_evaluation(cm)
          accuracies= cross_val_score(estimator=classifier,X=x_train,y=y_train,cv=10)
          print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
          print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))

          Precision: 0.87
          Recall: 0.82
          F1 Score: 0.85
          Accuracy: 81.21 %
          Standard Deviation: 1.36 %
```

Precision: 0.87

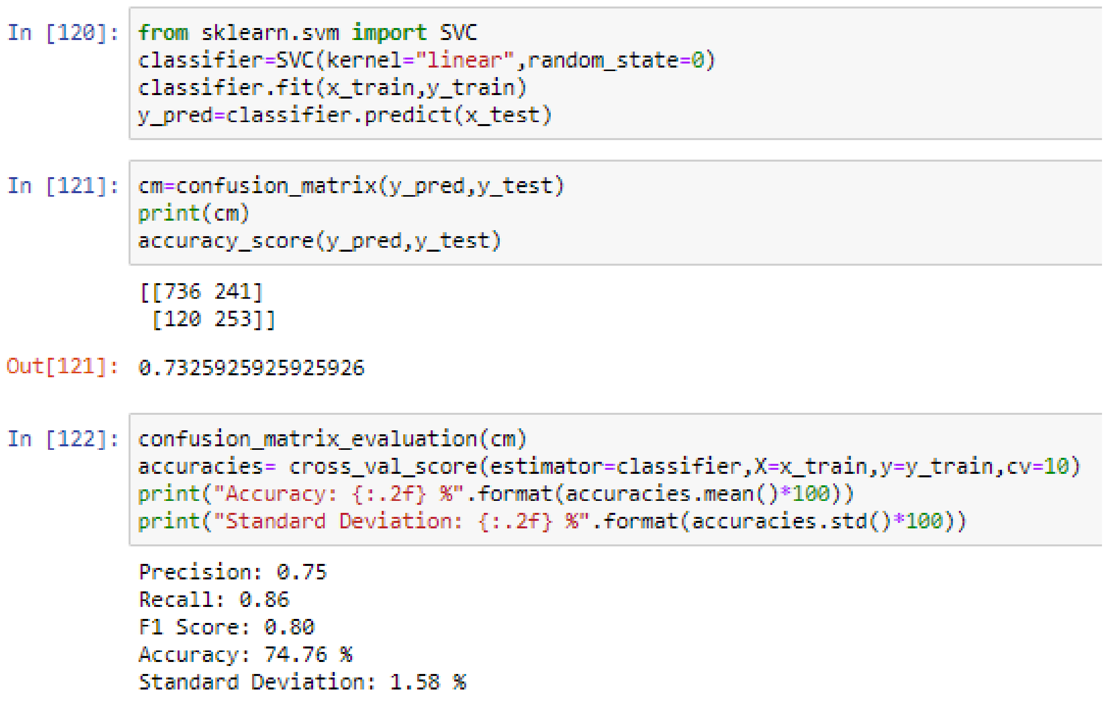
Recall: 0.82

F1 Score: 0.85

Accuracy: 81.21 %

Standard Deviation: 1.36

Figure 4.10: Training a SVM Model on the Training Dataset



```
In [120]: from sklearn.svm import SVC
          classifier=SVC(kernel="linear",random_state=0)
          classifier.fit(x_train,y_train)
          y_pred=classifier.predict(x_test)

In [121]: cm=confusion_matrix(y_pred,y_test)
          print(cm)
          accuracy_score(y_pred,y_test)

          [[736 241]
           [120 253]]

Out[121]: 0.7325925925925926

In [122]: confusion_matrix_evaluation(cm)
          accuracies= cross_val_score(estimator=classifier,X=x_train,y=y_train,cv=10)
          print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
          print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))

          Precision: 0.75
          Recall: 0.86
          F1 Score: 0.80
          Accuracy: 74.76 %
          Standard Deviation: 1.58 %
```

Precision: 0.75

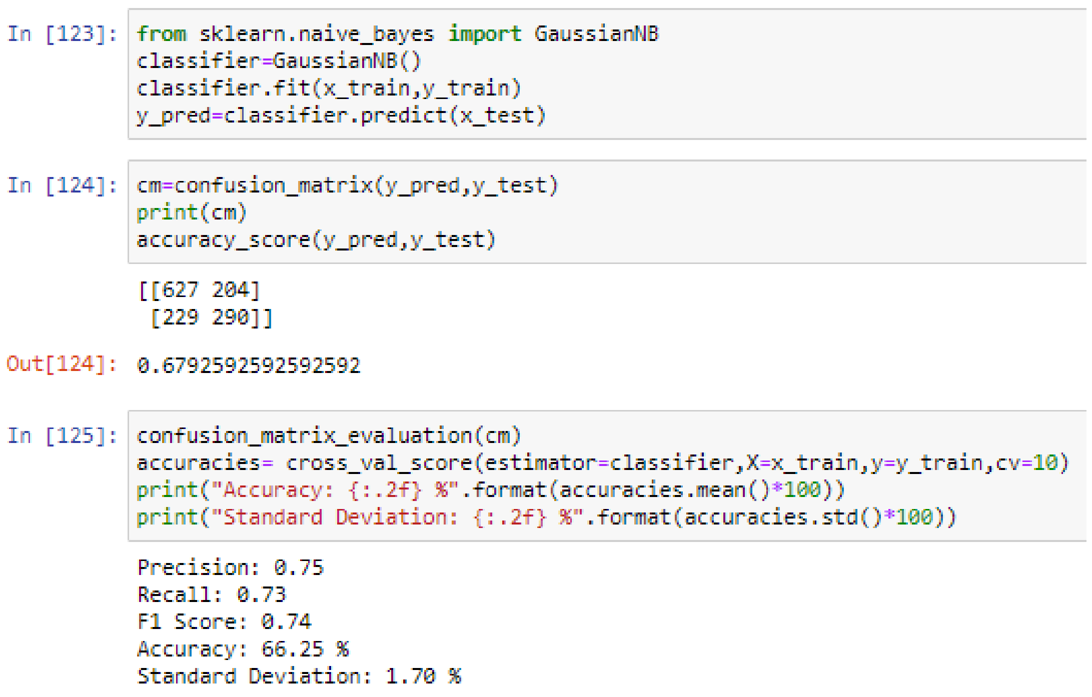
Recall: 0.86

F1 Score: 0.80

Accuracy: 74.76 %

Standard Deviation: 1.58

Figure 4.11: Training a Naive-Bayes Model on the Training Dataset



```
In [123]: from sklearn.naive_bayes import GaussianNB
          classifier=GaussianNB()
          classifier.fit(x_train,y_train)
          y_pred=classifier.predict(x_test)

In [124]: cm=confusion_matrix(y_pred,y_test)
          print(cm)
          accuracy_score(y_pred,y_test)

          [[627 204]
           [229 290]]

Out[124]: 0.6792592592592592

In [125]: confusion_matrix_evaluation(cm)
          accuracies= cross_val_score(estimator=classifier,X=x_train,y=y_train,cv=10)
          print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
          print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))

          Precision: 0.75
          Recall: 0.73
          F1 Score: 0.74
          Accuracy: 66.25 %
          Standard Deviation: 1.70 %
```

Precision: 0.75

Recall: 0.73

F1 Score: 0.74

Accuracy: 66.25 %

Standard Deviation: 1.70

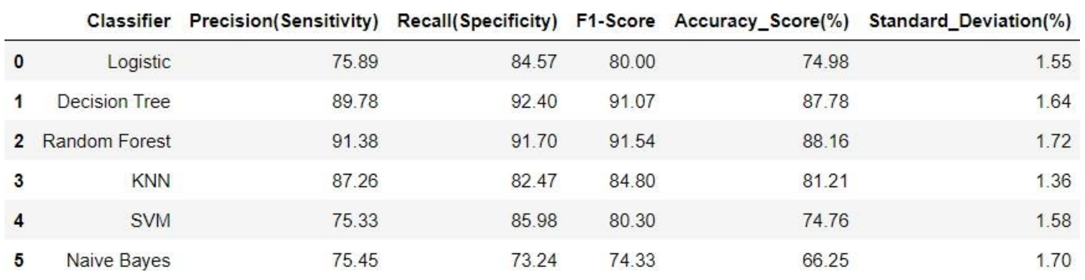
Chapter 5

Results and Discussion

In this section, we present the results of various classification models used for fake news detection. We evaluate each model using precision, recall, and F1 score, Accuracy, Standard deviation and provide a comparative analysis.

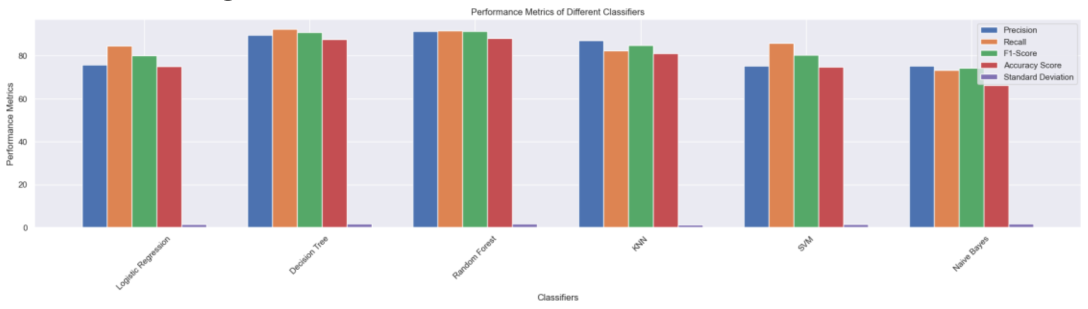
5.1 Result

Figure 5.1: Comparison of the models



	Classifier	Precision(Sensitivity)	Recall(Specificity)	F1-Score	Accuracy_Score(%)	Standard_Deviation(%)
0	Logistic	75.89	84.57	80.00	74.98	1.55
1	Decision Tree	89.78	92.40	91.07	87.78	1.64
2	Random Forest	91.38	91.70	91.54	88.16	1.72
3	KNN	87.26	82.47	84.80	81.21	1.36
4	SVM	75.33	85.98	80.30	74.76	1.58
5	Naive Bayes	75.45	73.24	74.33	66.25	1.70

Figure 5.2: Performance Metrics of Different Classifiers



5.2 Analysis

1. **Precision and Recall:**

   • Decision Tree and Random Forest classifiers have the highest precision, recall, and F1-score, indicating they are strong in correctly identifying both positive and negative instances.

   • Naive Bayes has the lowest precision and recall, suggesting it may struggle to accurately classify both classes.

2. **Accuracy Score:**

   • Decision Tree and Random Forest classifiers also have the highest accuracy scores, indicating their overall effectiveness in classification.

   • Naive Bayes has the lowest accuracy score, indicating it may not perform as well overall compared to other classifiers.

3. **Standard Deviation:**

   • All classifiers have relatively low standard deviations, suggesting consistency in their performance across different runs or samples of the data.

5.3 Conclusion

Our analysis revealed that Decision Tree and Random Forest classifiers consistently outperformed other models, exhibiting high precision, recall, F1-score, and accuracy score. We achieved 88.16% detection accuracy using random forest algorithm with some amount of True Positive Rate.Using the knowledge we've gained from this research, we can develop more effective strategies for identifying and addressing the challenges posed by fake news in the digital age.

5.4 Future Directions

For future research, we can explore ways to improve fake news detection. One approach is to experiment with new computer techniques, such as combining different methods or using more advanced learning methods. Additionally, we can consider analyzing how words are used and the context in which they appear. Another possibility is to integrate emotion detection into the analysis, which could help computers better distinguish between real and fake news.

References:

[1] Jianglin Huang, Yan-Fu Li, and Min Xie. An empirical analysis of data preprocessing for machine learning-based software cost estimation. *Information and software Technology*, 67:108–127, 2015.

[2] Julio CS Reis, André Correia, Fabrício Murai, Adriano Veloso, and Fabrício Benevenuto. Supervised learning for fake news detection. *IEEE Intelligent Systems*, 34(2):76–81, 2019.

[3] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter*, 19(1):22–36, 2017.

[4] Mustazzihim Suhaidi, R Abdul Kadir, and Sabrina Tiun. A review of feature extraction methods on machine learning. *J. Inf. Syst. Technol. Manag.*, 6(22):51–59, 2021.

[5] Pansy Nandwani and Rupali Verma. A review on sentiment analysis and emotion detection from text. *Social network analysis and mining*, 11(1):81, 2021.

[6] Samina Khalid, Tehmina Khalil, and Shamila Nasreen. A survey of feature selection and feature extraction techniques in machine learning. In *2014 science and information conference*, pages 372–378. IEEE, 2014.

[7] Sotiris B Kotsiantis, Ioannis D Zaharakis, and Panayiotis E Pintelas. Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review*, 26:159–190, 2006.

[8] Sotiris B Kotsiantis, Ioannis Zaharakis, P Pintelas, et al. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1):3–24, 2007.

[9] Wajiha Shahid, Bahman Jamshidi, Saqib Hakak, Haruna Isah, Wazir Zada Khan, Muhammad Khurram Khan, and Kim-Kwang Raymond Choo. Detecting and mitigating the dissemination of fake news: Challenges and future research opportunities. *IEEE Transactions on Computational Social Systems*, 2022.