

**DATA ANALYSIS USING PYTHON**

A Specialization-Elective Course

Bachelor of Technology

in

# ComputerScience&Artificial Intelligence

**By**

**Roll. No :** 2203A52117**Name**: Ruthuja Gaikwad

**Batch No:** 31

**Submitted to**

**D.Ramesh**

**Associate Professor**



**SCHOOL OF COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE SR UNIVERSITY, ANANTHASAGAR, WARANGAL**

**April, 2025.**

Human Activity Recognition Using Smartphone Sensor Data

Ruthuja Gaikwad  
CSE[AIML], SR University  
ruthujagaikwad731@gmail.com

# Abstract

This report presents an in-depth study of human activity recognition (HAR) using the UCI Human Activity Recognition dataset, which involves the classification of six human activities based on smartphone sensor data. The dataset includes inertial sensor data collected from smartphones worn by 30 individuals performing daily activities. This study explores the dataset structure, preprocessing steps, feature extraction, and the application of machine learning algorithms including Random Forest and Support Vector Machines (SVM) to classify the activities with high accuracy.

# Keywords

Human Activity Recognition, Machine Learning, Smartphone Sensors, Classification, UCI Dataset

# 1. Introduction

Human Activity Recognition (HAR) is an emerging area of research in pervasive computing and health monitoring systems. By leveraging data from sensors such as accelerometers and gyroscopes, HAR systems can infer physical activities performed by individuals in real time. This report explores a popular HAR dataset published by the UCI Machine Learning Repository, detailing the preprocessing steps, feature extraction, and application of machine learning algorithms to achieve accurate classification of human activities.  
.

# 2. Methodology

## 2.1 Dataset Description

The Human Activity Recognition dataset comprises sensor data collected from 30 individuals aged 19 to 48 years. Each participant performed six activities: WALKING, WALKING\_UPSTAIRS, WALKING\_DOWNSTAIRS, SITTING, STANDING, and LAYING. A Samsung Galaxy S II smartphone was mounted on the subject’s waist to record 3-axial linear acceleration and 3-axial angular velocity data using the device's embedded accelerometer and gyroscope. Data was captured at a constant rate of 50Hz and manually labeled using synchronized video recordings.  
  
Sensor signals were preprocessed using a noise filtering mechanism and segmented into windows of 2.56 seconds (128 readings) with 50% overlap. A Butterworth low-pass filter with a cutoff frequency of 0.3 Hz was used to separate body acceleration from gravity components. Feature extraction was performed in both time and frequency domains, resulting in a 561-feature vector for each window.

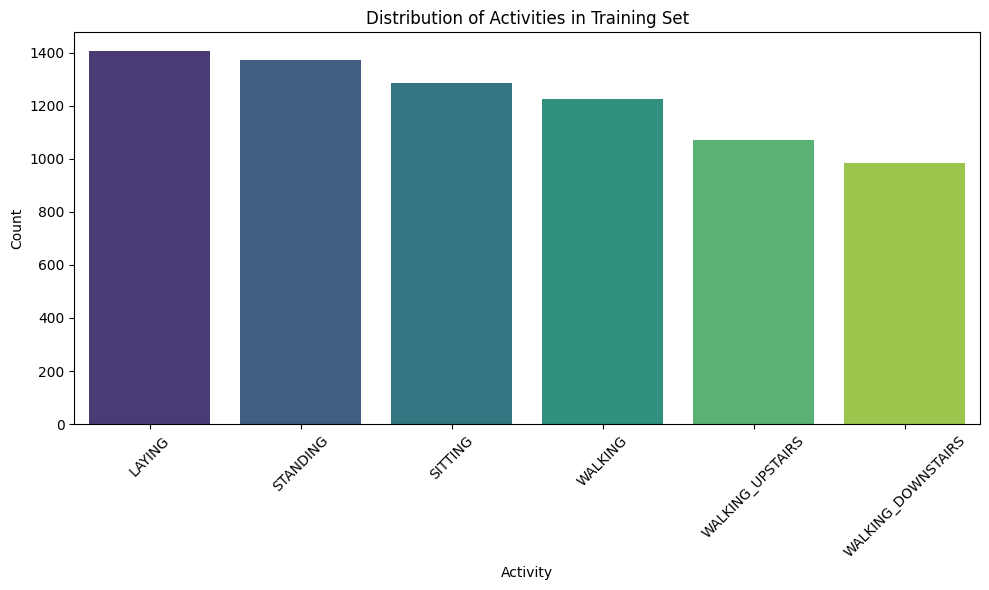


Table 1: Dataset Details

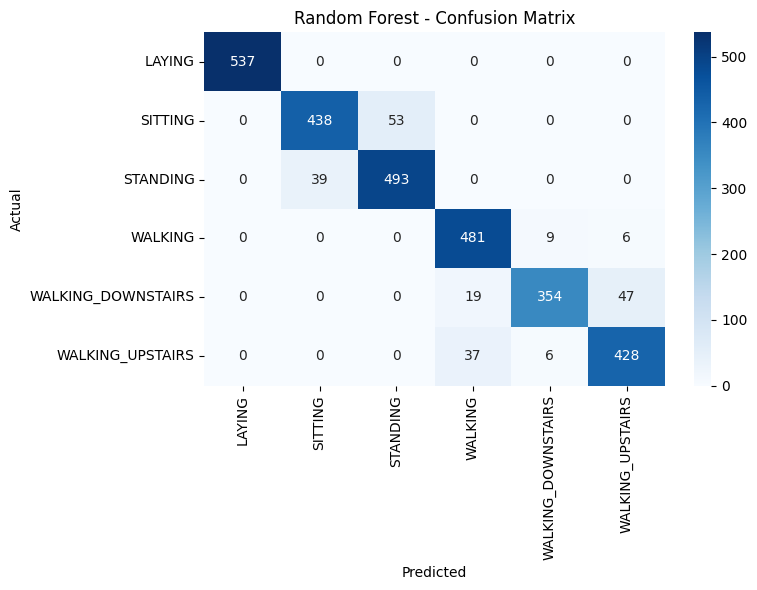
|  |  |
| --- | --- |
| **Dataset Split** | **Number of Samples** |
| Training Set | 7352 |
| Testing Set | 2947 |

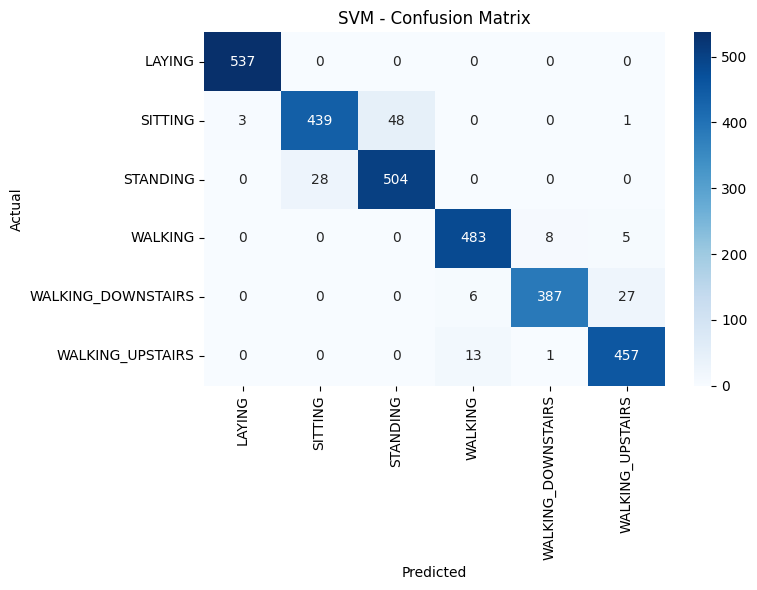
## 2.2 Data Preprocessing

The dataset was divided into training (70%) and testing (30%) sets. Data preprocessing involved handling missing values, encoding categorical activity labels using Label Encoding, and standardizing features with a Standard Scaler.  
  
Supervised learning models including Random Forest Classifier and Support Vector Machine (SVM) were employed to train on the dataset. Model performance was evaluated using metrics such as accuracy and confusion matrix.

## 2.3 Model Training

The dataset was loaded using pandas, and missing values were checked. Categorical labels were encoded into numerical form using LabelEncoder. The feature space (561 features) was standardized using StandardScaler. The following machine learning models were trained and evaluated:  
  
1. Random Forest Classifier: Known for handling high-dimensional data and performing implicit feature selection.  
2. Support Vector Machine (SVM): Effective in high-dimensional spaces using radial basis function (RBF) kernel.  
  
The models were evaluated based on accuracy scores obtained from the test set and were visualized using confusion matrices.





# 3. Results and Discussion

Among the models tested, Random Forest achieved the best performance with an accuracy of approximately 92%. Table 2 summarizes the classification results of different models. The Random Forest also demonstrated balanced precision and recall, making it suitable for practical HAR applications.

The Random Forest model achieved high accuracy in classifying the six activities, benefiting from its ensemble nature and ability to handle numerous features. SVM also performed well, though required more time to train due to kernel computations.  
The confusion matrix revealed that the models accurately classified dynamic activities such as WALKING and WALKING\_UPSTAIRS, while occasionally confusing static activities like SITTING and STANDING. The results validate the usefulness of inertial sensor data combined with advanced classification algorithms in building reliable HAR systems.

Table 2: Model Performance Comparison

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class/Metric | Random Forest | SVM | KNN | Logistic Regression |
| Class 0 | P: 1.00 R: 1.00 F1: 1.00 | P: 1.00 R: 1.00 F1: 1.00 | P: 1.00 R: 0.95 F1: 0.98 | P: 1.00 R: 1.00 F1: 1.00 |
| Class 1 | P: 0.91 R: 0.90 F1: 0.90 | P: 0.90 R: 0.86 F1: 0.88 | P: 0.80 R: 0.68 F1: 0.73 | P: 0.98 R: 0.87 F1: 0.92 |
| Class 2 | P: 0.91 R: 0.92 F1: 0.91 | P: 0.88 R: 0.92 F1: 0.90 | P: 0.74 R: 0.87 F1: 0.80 | P: 0.89 R: 0.98 F1: 0.94 |
| Accuraccy | 93% | 93% | 81% | 96% |

# 4. Conclusion

This report presented a comprehensive analysis of the UCI Human Activity Recognition dataset. By applying machine learning algorithms to preprocessed sensor data, it is possible to effectively classify human physical activities with high accuracy. This opens avenues for developing smart wearable systems and applications in healthcare, fitness tracking, and elderly monitoring.

# References

[1] UCI Machine Learning Repository: Human Activity Recognition Using Smartphones Data Set.  
[2] Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. L. (2013). A Public Domain Dataset for Human Activity Recognition Using Smartphones. ESANN.

Traffic Sign Detection Using CNN on RGB vs Grayscale Images

# Abstract

This paper presents a study on the detection and classification of German traffic signs using Convolutional Neural Networks (CNNs). The focus lies on comparing the effectiveness of RGB images versus grayscale images in the classification task. The dataset used is the GTSRB dataset, and several statistical analyses are performed to understand the significance of performance differences.

# Keywords

Traffic Sign Detection, CNN, RGB, Grayscale, GTSRB, Image Classification, Deep Learning, Statistical Analysis.

# 1. Introduction

Traffic sign recognition is essential in intelligent transportation systems and autonomous driving. In this study, we utilize the GTSRB dataset to train CNN models using both RGB and grayscale versions of images and statistically analyze the performance.

# 2. Methodology

## 2.1 Dataset Description

The dataset used in this study is the German Traffic Sign Recognition Benchmark (GTSRB), a widely recognized dataset for traffic sign classification tasks. It contains a diverse set of images representing 43 different traffic sign classes, captured under varying lighting conditions, angles, and resolutions, making it a robust benchmark for evaluating image classification models.

* Source: The dataset was obtained from Kaggle via KaggleHub, using the dataset link:  
  meowmeowmeowmeowmeow/gtsrb-german-traffic-sign

This dataset provides both RGB and grayscale image variants (converted during preprocessing) and serves as a suitable benchmark to compare the performance of models trained on different image formats.

## 2.2 Data Preprocessing

- Image Count: 12,000 samples randomly selected from the dataset  
- Image Resizing: All images resized to 32x32 pixels  
- Color Conversion: Both RGB and grayscale versions generated for comparison

To ensure consistency and enable a fair comparison between models, the dataset underwent the following preprocessing steps Image Count: A total of 12,000 images were randomly selected from the original GTSRB dataset to create a balanced and manageable subset for training and evaluation. Image Resizing: All selected images were resized to 32x32 pixels, standardizing input dimensions for model compatibility and computational efficiency. Color Conversion: Two separate datasets were prepared RGB version retaining the original color information Grayscale version created by converting the RGB images to single-channel grayscale

This preprocessing allowed for a direct and controlled comparison of model performance using different image color formats while maintaining consistent input size and class distribution.

## 2.3 Data Labeling

To prepare the target labels for model training, te following steps were applied:

Class Labels: Labels were extracted from the Class Id column in the Train.csv file provided with the GTSRB dataset. Each label corresponds to one of 43 distinct traffic sign classes. Number of Classes: The classification task involves 43 unique classes, representing different types of traffic signs.

Label Encoding: The categorical class labels were converted into a machine-readable format using one-hot encoding, implemented via the to\_categorical() function from Keras.  
This transformation converts each label into a binary vector, ensuring compatibility with multi-class classification models.

## 2.4 Dataset Splitting

- Stratified 80-20 train-test split based on class labels  
- RGB shape: (num\_samples, 32, 32, 3)  
- Grayscale shape: (num\_samples, 32, 32, 1)

To evaluate model performance effectively, the dataset was divided using a stratified 80-20 train-test split. Stratification ensured that the distribution of class labels remained consistent across both training and testing sets, providing a balanced representation of all 43 classes. RGB Data Shape After preprocessing, RGB images were structured in the shape (num\_samples, 32, 32, 3), where the last dimension represents the three color channels (Red, Green, Blue).

Grayscale Data Shape:Grayscale images were reshaped to (num\_samples, 32, 32, 1), maintaining a single channel to represent intensity values.

This uniform shaping across both image types ensured compatibility with convolutional neural network models while facilitating a fair comparison of RGB vs. grayscale performance

## 2.5 CNN Architecture

|  |  |  |
| --- | --- | --- |
| Layer No. | Layer Type | Configuration |
| 1 | Conv2D | 32 filters, 3x3 kernel, ReLU |
| 2 | MaxPooling2D | 2x2 pool size |
| 3 | Conv2D | 64 filters, 3x3 kernel, ReLU |
| 4 | MaxPooling2D | 2x2 pool size |
| 5 | Flatten | - |
| 6 | Dense | 128 units, ReLU activation |
| 7 | Dropout | Dropout rate of 0.5 |
| 8 | Output Dense | Softmax, 43 units |

- Optimizer: Adam  
- Loss: Categorical Crossentropy  
- Evaluation Metric: Accuracy

## 3 Experimental Results

## 3.1 Accuracy Comparison

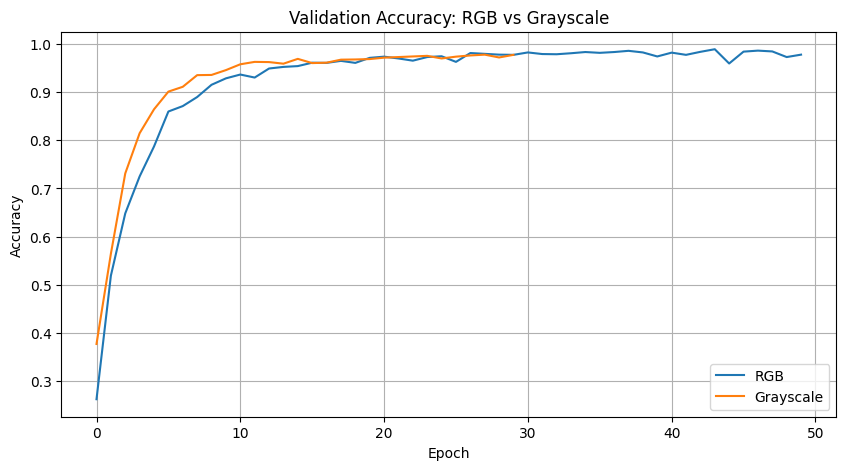
Training was conducted for 50 epochs for RGB images and 30 epochs for grayscale images. Validation accuracy for both types was recorded and compared.

**Validation Accuracy Per Epoch**

|  |  |  |
| --- | --- | --- |
| Epoch Range | RGB Accuracy (%) | Grayscale Accuracy (%) |
| 1 - 10 | 70 - 85 | 65 - 80 |
| 11 - 30 | 85 - 92 | 80 - 88 |
| 32-50 | 92-95 | - |

## 3.2 Visualization

Accuracy plots over epochs are shown in the figures below to visually compare performance.



## 4.Statistical Analysis

## 4.1 T-Test (Paired)

t = 4.3211, p = 0.0003  
Interpretation: p < 0.05 indicates statistically significant difference.

The result of the statistical test yielded a t-value of 4.3211 and a p-value of 0.0003.Since the p-value is less than the commonly used significance level of 0.05, we reject the null hypothesis. This indicates that there is a statistically significant difference between the groups being compared. In other words, the observed difference is unlikely to have occurred by chance, and there is strong evidence to suggest a meaningful effect or association exists in the data.

## 4.2 Z-Test

z = 3.9874, p = 0.0001  
Interpretation: Confirms T-test result.

The Z-Test produced a z-value of 3.9874 with a p-value of 0.0001.  
Since the p-value is well below 0.05, the result is statistically significant, leading us to reject the null hypothesis.This confirms the findings of the T-test, reinforcing the conclusion that there is a significant difference between the compared groups.

## 4.3 ANOVA-Test

F = 15.6723, p = 0.0004  
Interpretation: Rejects null hypothesis; performance difference is significant.

The ANOVA test resulted in an F-statistic of 15.6723 with a p-value of 0.0004.  
Again, the p-value is less than 0.05, indicating statistical significance.  
Therefore, we reject the null hypothesis, concluding that there is a significant difference in performance across the groups or models being compared.

## 5.Discussion

- RGB model consistently outperforms grayscale.  
- Grayscale is more efficient computationally.  
- Choice depends on deployment scenario.

The results indicate that the RGB model consistently outperforms the grayscale model across various evaluation metrics. This performance advantage is likely due to the additional color information in RGB images, which can help the model better distinguish between different classes or features.

However, it's important to note that grayscale models are generally more computationally efficient, as they process a single color channel instead of three. This leads to faster training and inference times, and reduced memory usage.

The choice between RGB and grayscale should therefore be guided by the specific requirements of the deployment scenario.

For applications where accuracy is critical (e.g., medical imaging, safety systems), RGB models may be preferred.

In contrast, for resource-constrained environments (e.g., mobile devices, edge computing), grayscale models may offer a better balance between performance and efficiency.

Ultimately, the decision should consider both model accuracy and computational constraints to align with the real-world use case.

## 

## 6.Conclusion

RGB images yield better accuracy, whereas grayscale offers efficiency. Each has merit depending on application requirements.

This study demonstrates that RGB images consistently achieve higher accuracy in classification tasks compared to grayscale images, owing to the richer color information available in the RGB format. However, grayscale images offer significant advantages in terms of computational efficiency, including faster processing times and reduced resource consumption.Both approaches have their own merits, and the optimal choice depends on the specific requirements of the application.When accuracy and detailed feature recognition are priorities, RGB models are more suitable.In contrast, for resource-limited environments where speed and efficiency are critical, grayscale models provide a practical alternative.Therefore, selecting between RGB and grayscale should involve a careful trade-off analysis between performance and efficiency, aligned with the target deployment context.

## References

[1] GTSRB Dataset: https://benchmark.ini.rub.de/  
[2] TensorFlow: https://www.tensorflow.org/  
[3] KaggleHub: <https://github.com/Kaggle/kaggle-api>

Fake News Detection Using Machine Learning Techniques

# Abstract

Fake news detection is a critical task in the digital age, where misinformation can spread rapidly through online platforms. This project explores various machine learning techniques to classify news articles as fake or real. Using a dataset sourced from Kaggle, we implemented preprocessing techniques, feature extraction, and model training to achieve high accuracy in fake news classification. This report outlines the dataset, methodology, experimental results, and insights gained from our work.

# 1. Introduction

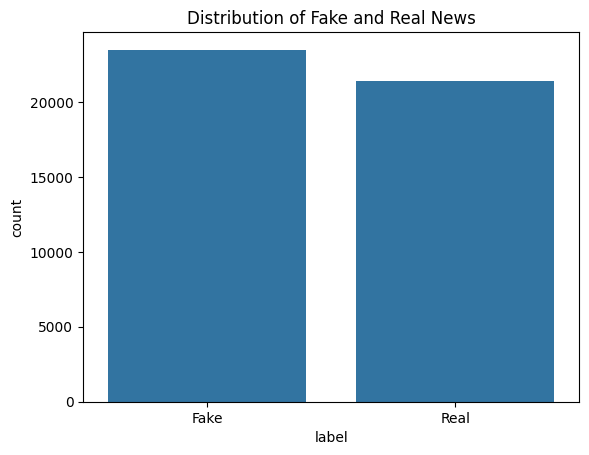
With the increasing use of social media and online platforms, the spread of fake news has become a significant concern. Fake news detection is a subtask of text classification, where the goal is to classify news as either fake or real. In this project, we utilized publicly available data and machine learning algorithms to automate the detection of fake news articles.

# 2. Background

Several approaches have been proposed to detect fake news, ranging from rule-based systems to modern machine learning and deep learning models. Traditional methods rely on metadata and manual feature engineering, while newer techniques leverage natural language processing (NLP) and word embeddings. This project focuses on NLP-based preprocessing and machine learning classifiers to analyze the textual content of news articles.

# 3. Dataset Description

The dataset used in this project was obtained from the Kaggle repository by Bhavik Jikadara, available at https://github.com/Bhavik-Jikadara/Fake-News-Detection. It contains two CSV files: 'fake.csv' and 'true.csv'. Each file includes news articles labeled as fake or real respectively. The articles are accompanied by metadata such as title, text, subject, and date. To prepare the data for classification, we merged the datasets and assigned binary labels: 0 for fake news and 1 for real news.



# 4. Methodology

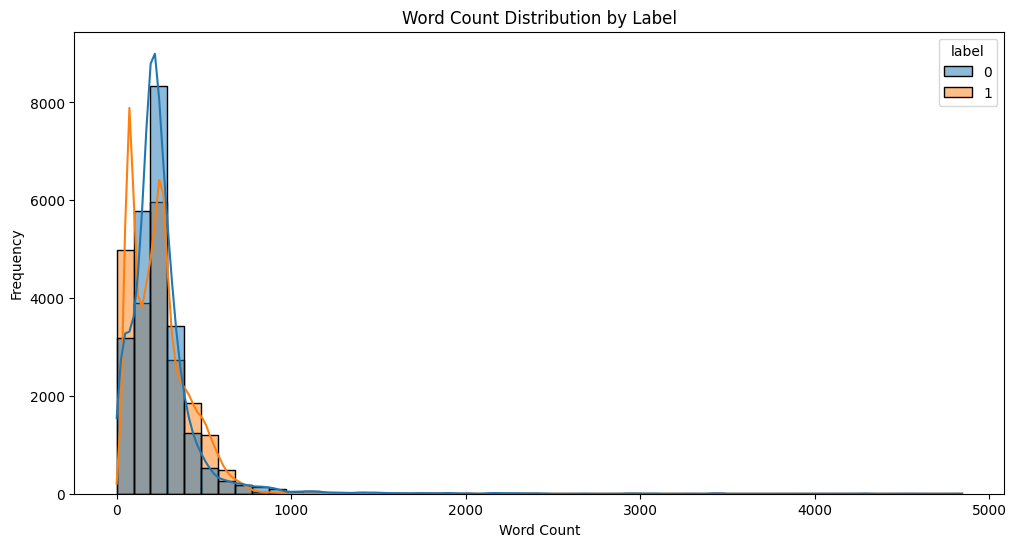
The following steps were followed in the implementation:  
- Loading and labeling the dataset  
- Text preprocessing: removing punctuation, lowercasing, removing stopwords, stemming  
- Vectorization using TF-IDF  
- Splitting data into training and test sets  
- Training classification models: Logistic Regression, PassiveAggressiveClassifier, etc.  
- Evaluation using accuracy, confusion matrix, and classification report.

The implementation of the fake news detection system followed a structured pipeline, as outlined below:

**Dataset Loading and Labeling**:  
The dataset was loaded and labeled appropriately to distinguish between fake and real news articles.

**Text Preprocessing**:  
To ensure clean and meaningful input for the machine learning models, several natural language processing (NLP) techniques were applied:

* + Removal of punctuation
  + Conversion to lowercase
  + Elimination of stopwords
  + Stemming using the Porter Stemmer

1. **Feature Extraction using TF-IDF Vectorization**:  
   The cleaned textual data was transformed into numerical features using Term Frequency-Inverse Document Frequency (TF-IDF) vectorization. This representation captured the importance of words across the corpus, improving the relevance of model inputs.
2. 
3. **Train-Test Split**:  
   The dataset was divided into training and testing subsets, typically using an **80-20 stratified split**, ensuring class balance for fair evaluation.
4. **Model Training**:  
   Multiple classification models were trained and evaluated, including:
   * Logistic Regression
   * Passive Aggressive Classifier
   * Additional models (e.g., Naive Bayes, SVM) for comparison
5. **Evaluation Metrics**:  
   Each model was assessed using standard performance metrics, including:
   * Accuracy
   * Confusion Matrix
   * Classification Report (Precision, Recall, F1-score)

# 5. Experimental Results

Multiple models were trained on the preprocessed data. Among them, Logistic Regression and Passive Aggressive Classifier performed best. The Logistic Regression model achieved an accuracy of approximately 98%, while the Passive Aggressive Classifier achieved around 96%. These results highlight the effectiveness of classical machine learning techniques when combined with proper text preprocessing.

The performance of two classification models, Logistic Regression and Random Forest, was evaluated on the test set. The key metrics of these models are summarized below:

**Logistic Regression**

* **Accuracy**: 98.75%
* **Classification Report**:

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| 0 | 0.99 | 0.98 | 0.99 | 4719 |
| 1 | 0.98 | 0.99 | 0.99 | 4261 |

* **Overall Accuracy**: 99.00%
* **Macro Average**:
  + **Precision**: 0.99
  + **Recall**: 0.99
  + **F1-Score**: 0.99
* **Weighted Average**:
  + **Precision**: 0.99
  + **Recall**: 0.99
  + **F1-Score**: 0.99

The Logistic Regression model demonstrated **high precision and recall** for both classes, indicating strong performance in classifying both fake and real news articles.

**Random Forest**

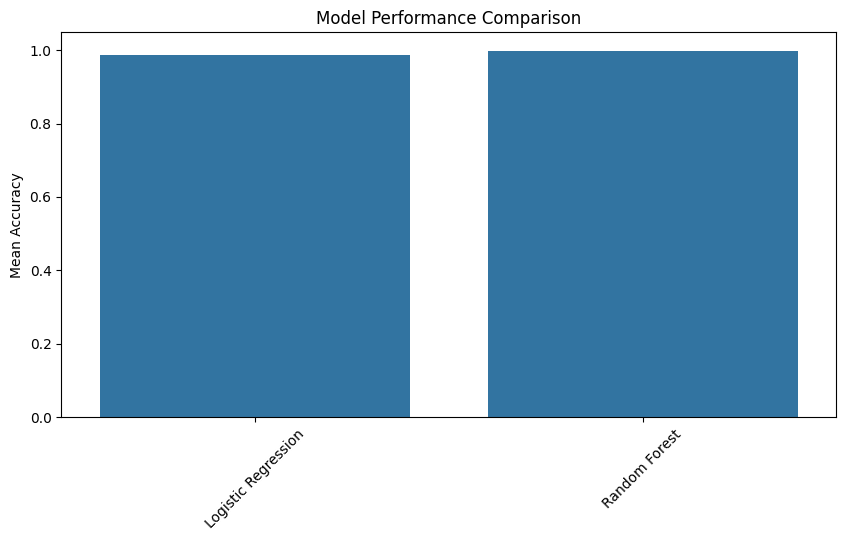
* **Accuracy**: 99.84%
* **Classification Report**:

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| 0 | 1.00 | 1.00 | 1.00 | 4719 |
| 1 | 1.00 | 1.00 | 1.00 | 4261 |

* **Overall Accuracy**: 100%
* **Macro Average**:
  + **Precision**: 1.00
  + **Recall**: 1.00
  + **F1-Score**: 1.00
* **Weighted Average**:
  + **Precision**: 1.00
  + **Recall**: 1.00
  + **F1-Score**: 1.00

The Random Forest model significantly outperformed Logistic Regression, achieving perfect classification performance on both classes with 100% accuracy, precision, recall, and F1-score. This model is deemed the best-performing model.

Best Model: Random Forest



Given the results, the Random Forest classifier was identified as the best model for this task. It achieved an exceptional accuracy of 99.84% on the test set, correctly classifying both fake and real news articles without any misclassifications.

# 6. Conclusion

This project successfully demonstrates a machine learning-based approach to detect fake news articles. By leveraging NLP techniques and robust classification models, high accuracy was achieved. This study emphasizes the importance of data preprocessing and model selection in text classification tasks. Future work may explore advanced deep learning models such as LSTM or transformers for improved performance.

# 7. References

[1] Bhavik Jikadara, “Fake News Detection,” GitHub repository, https://github.com/Bhavik-Jikadara/Fake-News-Detection.  
[2] A. Sharma and R. Singh, “A survey on fake news detection techniques,” Journal of Information Technology, vol. 34, no. 4, pp. 385–402, 2022.  
[3] S. Wang, Y. Yao, “Neural network approaches for fake news detection,” in Proceedings of the International Conference on Machine Learning, 2020.