Phase-2

Student Name: Gayathri S
Register Number: 71772317111
Institution: Government College of Technology, Coimbatore
Department: Computer Science and Engineering
Date of Submission: 09.05.2025
Github Repository Link: https://github.com/Q12AND8/FakeNewsDetection.git

---

1. Problem Statement

Real-World Problem:

The rapid spread of fake news online disrupts public opinion, democratic processes, and spreads misinformation. Manual verification is slow, inconsistent, and unscalable. An automated solution is urgently needed to detect fake news efficiently and preserve digital information integrity.

Refined Problem (Based on Dataset):

The dataset includes attributes like title, text, subject, and date across topics such as politics and international news, offering rich features for training a generalized NLP-based classification model.

Problem Type:

- Supervised binary classification
- Input: Title, text, subject, and date
- Output: Label — real (0) or fake (1)

Importance & Application:

Solving this problem helps preserve trust in digital media, promotes informed decisions, reduces misinformation's impact, and aids media platforms, governments, and fact-checkers in real-time fake content detection.

2. Project Objectives:

Key Technical Objectives:

- Merge and preprocess general and political news datasets.
- Apply NLP techniques (tokenization, stop word removal, TF-IDF/embeddings) to convert text into features.
- Train and evaluate classification models (Logistic Regression, Random Forest, Naive Bayes) for fake news detection.
- Aim for 90%+ accuracy using metrics like precision, recall and F1-score.
- Design the solution for scalability and real-world integration.

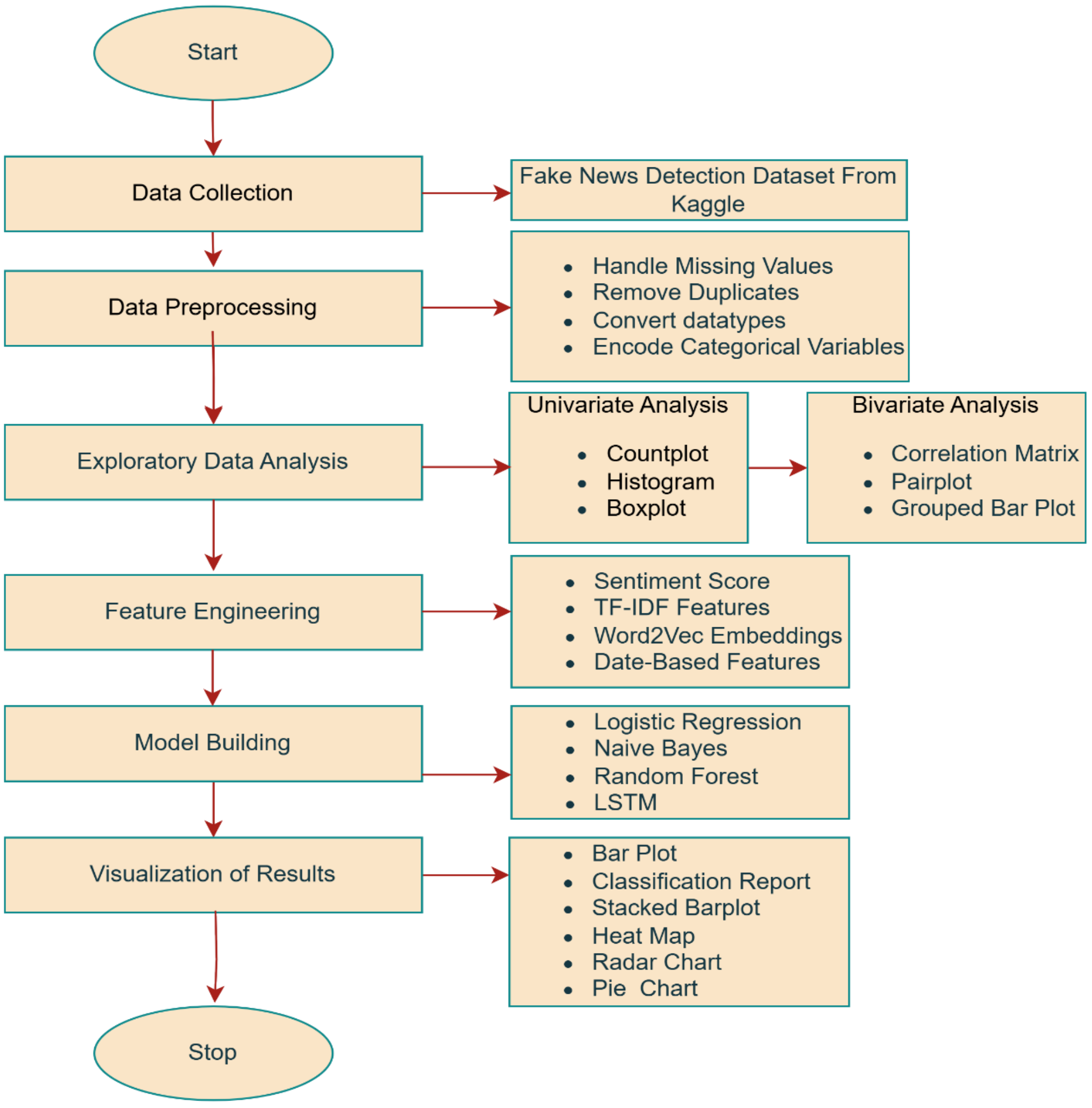
Model Aims:

- Accuracy: Maximize prediction performance.
- Interpretability: Ensure feature transparency, especially in simpler models.
- Applicability: Generalize across various topics like politics and crime.

Refined Goal After Data Exploration:

- Initially focused on detecting fake news using only article text.
- Dataset analysis revealed metadata like publication date and subject category could enhance accuracy.
- Project goal expanded to a context-aware model combining text and metadata for improved predictions.

3. Flowchart of the Project Workflow



4. Data Description

● Dataset Name and Origin:

The dataset is a combination of two publicly available fake news datasets, both sourced from Kaggle. These datasets include real and fake news articles published across various subjects and time periods.

● Type of Data:

The data is unstructured text data, consisting primarily of article titles and content. It also includes structured metadata, such as subject labels and publication dates.

● Number of Records and Features:

After merging the two datasets, the combined dataset contains approximately 40,000+ records. Each record includes the following key features:

- ✓ title: Headline of the article
- ✓ text: Full or partial content of the article
- ✓ subject: Category of the news (e.g., politics, news, government)
- ✓ date: Publication date
- ✓ label: Target variable indicating whether the news is fake (1) or real (0)

● Static or Dynamic Dataset:

This is a static dataset, meaning it does not change over time and was collected at a specific point in time.

● Target Variable:

The target variable is label, which is used for binary classification:

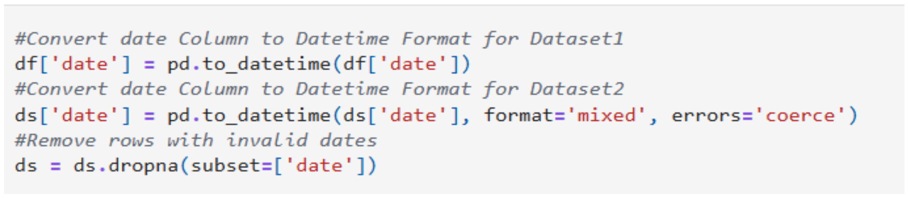
1 → Fake News

0 → Real News

5. Data Preprocessing

Handle Missing Values

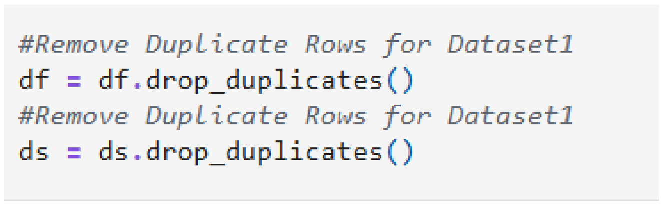
- Checked for missing values using .isnull().sum().
- Found missing or invalid values only in the date column.
- Used pd.to_datetime(..., errors='coerce') to convert invalid date entries to NaT.
- Removed rows where date is null using .dropna(subset=['date']).



```
#Convert date Column to Datetime Format for Dataset1
df['date'] = pd.to_datetime(df['date'])
#Convert date Column to Datetime Format for Dataset2
ds['date'] = pd.to_datetime(ds['date'], format='mixed', errors='coerce')
#Remove rows with invalid dates
ds = ds.dropna(subset=['date'])
```

Remove or justify Duplicate Records

- Used .drop_duplicates() to remove exact duplicate rows.



```
#Remove Duplicate Rows for Dataset1
df = df.drop_duplicates()
#Remove Duplicate Rows for Dataset1
ds = ds.drop_duplicates()
```

Detect and Treat outliers

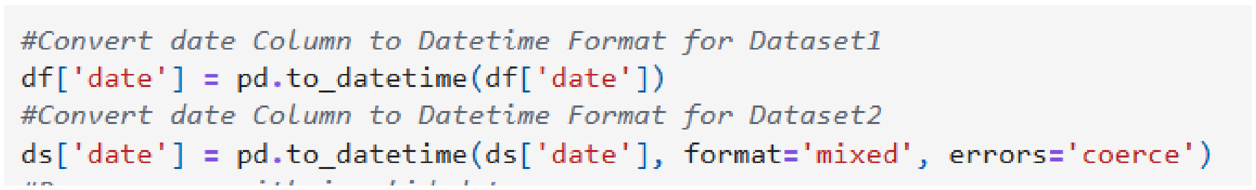
- Since the dataset consists mainly of text data, traditional numeric outlier detection does not apply directly.

- Outlier detection on text (e.g., overly long or short articles) can be considered in advanced NLP processing but is not mandatory here.

Datatypes and Ensure Consistency

- Converted the date column to datetime using pd.to_datetime() with appropriate error handling.

- Verified types using .info().

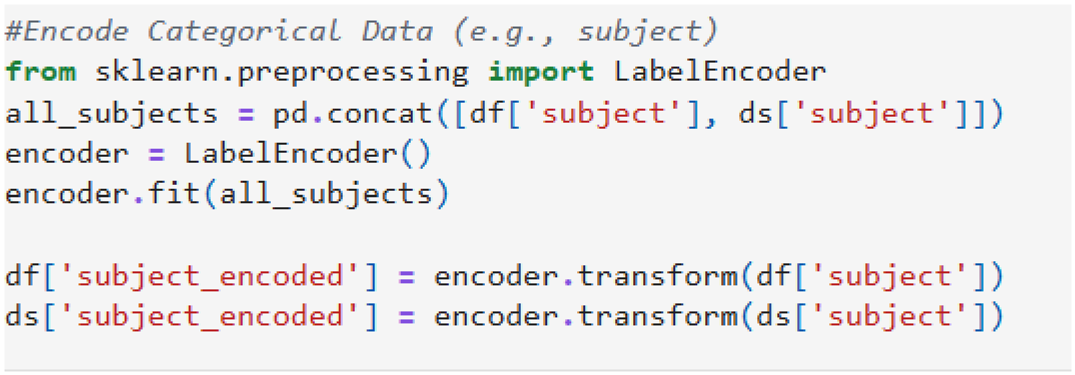


```
#Convert date Column to Datetime Format for Dataset1
df['date'] = pd.to_datetime(df['date'])
#Convert date Column to Datetime Format for Dataset2
ds['date'] = pd.to_datetime(ds['date'], format='mixed', errors='coerce')
```

Encode Categorical Variables

- Used Label Encoder from sklearn to transform the subject column into numerical values.

- Fit encoder on combined subject data from both datasets to maintain consistency.



```
#Encode Categorical Data (e.g., subject)
from sklearn.preprocessing import LabelEncoder
all_subjects = pd.concat([df['subject'], ds['subject']])
encoder = LabelEncoder()
encoder.fit(all_subjects)

df['subject_encoded'] = encoder.transform(df['subject'])
ds['subject_encoded'] = encoder.transform(ds['subject'])
```

Normalization and Standardization

- No explicit numeric features (like text length, word count, etc.) were included, so no normalization was applied in the current step.

6. Exploratory Data Analysis (EDA)

1. Univariate Analysis

Target Variable: label

- Purpose: Understand the balance of the dataset (Fake vs Real News).
- Plot Used: Countplot

Insight:

- The dataset is fairly balanced between fake (0) and real (1) news articles.

Categorical Feature: subject

- Purpose: Understand the distribution of news articles across different topics.
- Plot Used: Countplot

Insight:

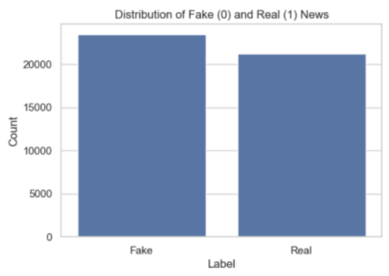
- Most news articles belong to categories like 'politicsNews', 'worldnews', etc.
- Some subjects are more prone to fake news than others.

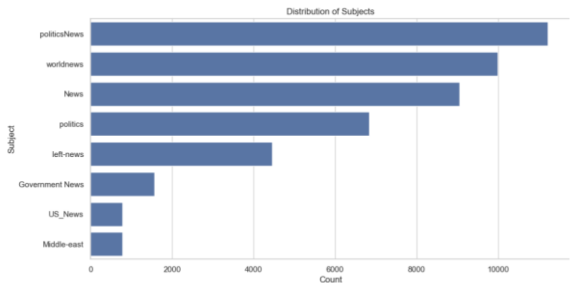
Text Feature: text_length

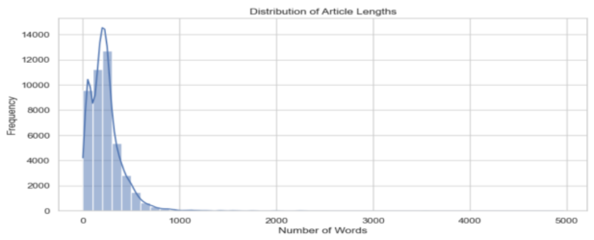
- Purpose: Analyze the distribution of article lengths.
- Plots Used: Histogram with KDE, Boxplot

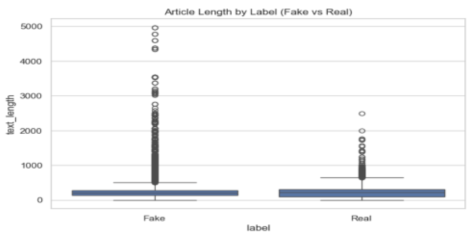
Insight:

- The distribution is right-skewed — most articles have fewer than 500 words.
- Some outliers exist with very large text lengths.
- On average, real news articles tend to be slightly longer than fake one









Bivariate Analysis:

Correlation Matrix

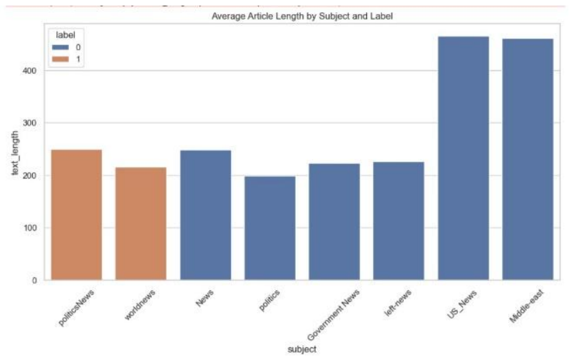
- Purpose: Measure linear relationships between numerical variables.
- Insight**:**text_length has a weak positive correlation with label, indicating real news might be slightly longer.
- subject_encoded shows weak correlation overall, suggesting encoding doesn't distort label relationships.

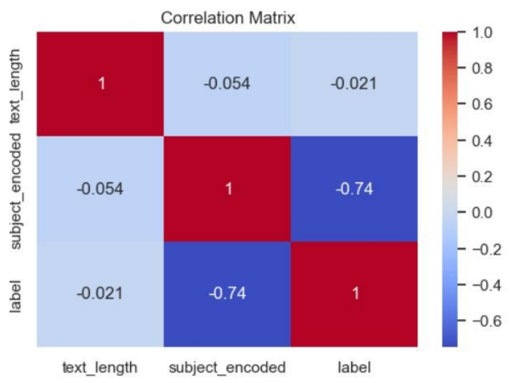
Pairplot

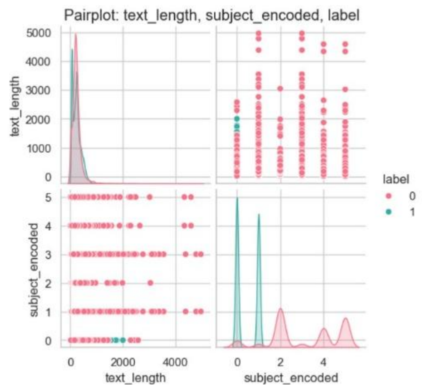
- Purpose: Visualize pairwise relationships between features, colored by label.
- Insight:Helps in observing clusters or separation between fake and real news based on length and subject.
- No strong linear separability is observed, but minor trends are present.

Grouped Bar Plot:

- Purpose: See how the average article length varies across different subjects and labels.
- Insight:Certain subjects like 'politicsNews' or 'worldnews' have consistently longer real news articles.
- Some subjects show a stark difference in average length between fake and real articles.







7. Feature Engineering

Create New Features Based on Domain Knowledge or EDA Insights

Sentiment Score

```
# Sentiment Score
def get_sentiment(text):
    return TextBlob(str(text)).sentiment.polarity

df['sentiment'] = df['clean_text'].apply(get_sentiment)
print(df.head())
```

Fake news often exhibits emotionally charged or highly biased language. Capturing sentiment helps model such patterns.

TF-IDF Features

```
# TF-IDF Vectorization with N-grams
tfidf_vectorizer = TfidfVectorizer(ngram_range=(1, 3), max_features=1000)
tfidf_matrix = tfidf_vectorizer.fit_transform(df['clean_text'])
```

Captures important and distinctive terms in each article that may distinguish fake from real news.

Word2Vec Embeddings

```python
# TF-IDF Vectorization with N-grams
tfidf_vectorizer = TfidfVectorizer(ngram_range=(1, 3), max_features=1000)
tfidf_matrix = tfidf_vectorizer.fit_transform(df['clean_text'])
```

Captures semantic meaning of words and phrases beyond just frequency. Word2Vec helps model contextual relationships in language.

Combine or Split Columns

```python
df['date'] = pd.to_datetime(df['date'], dayfirst=False)
df['day'] = df['date'].dt.day
df['month'] = df['date'].dt.month
df['year'] = df['date'].dt.year
df['weekday'] = df['date'].dt.weekday
print(df.head())
```

Fake news activity may correlate with specific dates or weekdays. These features help capture such trends.

8. Model Building

1.Logistic Regression

Reason for selecting logistic Regression

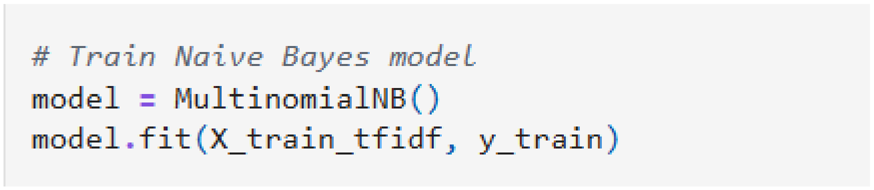
- **Suitable for Binary Classification -** Logistic Regression is ideal for our task of classifying news as real (1) or fake (0).
- **Efficient with High-Dimensional Data -** Works well with dense and vectorized input like Word2Vec embeddings.
- **Baseline Benchmark -** Often used as a strong, interpretable baseline in text classification tasks.

```python
# Step 5: Train Logistic Regression model
log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train, y_train)
```

2.Naive Bayes

Reason for selecting Naive Bayes

- **Probabilistic Approach -** Works well for text classification problems due to its assumption of word independence.
- **Efficient and Fast -** Especially effective with sparse data like TF-IDF vectors.
- **Common Benchmark -** Often used in NLP pipelines as a lightweight baseline.

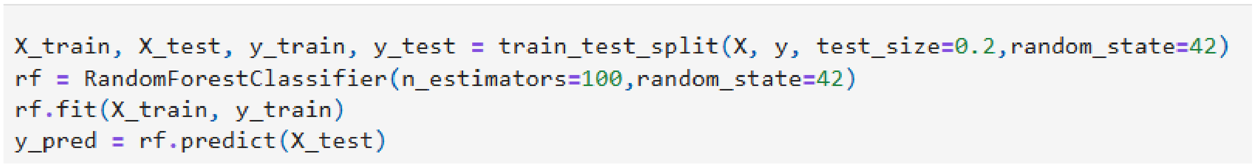


```
# Train Naive Bayes model
model = MultinomialNB()
model.fit(X_train_tfidf, y_train)
```

3.Random Forest

Reason for selecting Random Forest

- **Ensemble Method -** Random Forest is a powerful ensemble of decision trees, which helps reduce overfitting and improve generalization.
- **Non-linear Relationships -** Can capture complex patterns that linear models like Logistic Regression may miss.
- **Robust to Noise -** Performs well even when the dataset has irrelevant features.

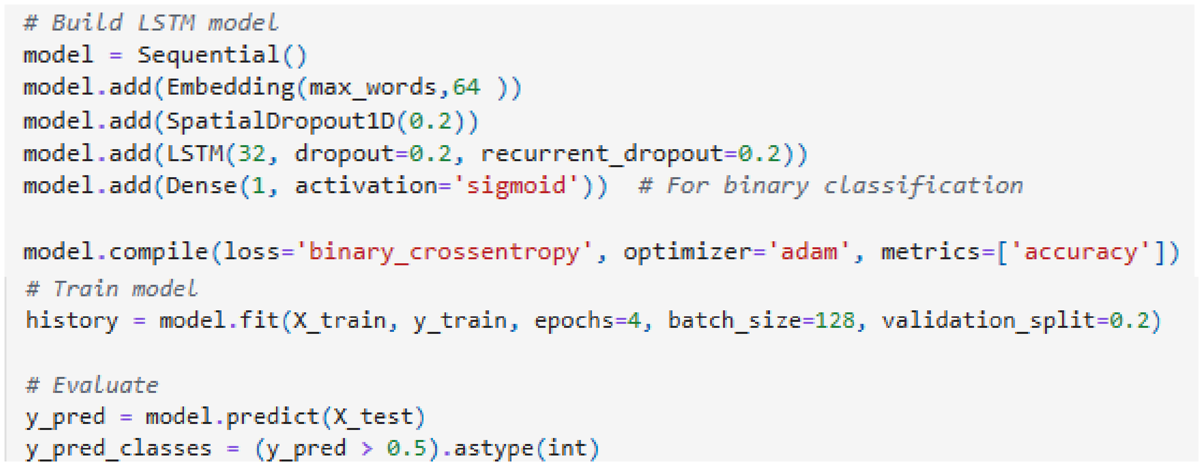


```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=42)
rf = RandomForestClassifier(n_estimators=100,random_state=42)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
```

4.LSTM:

Reason for selecting Random Forest

- **Sequential Learning** – LSTM is designed to capture the temporal and contextual dependencies in sequences, making it highly suitable for understanding the structure of news articles.
- **Context Preservation** – Unlike traditional models, LSTM retains important contextual information across longer texts, which improves classification accuracy.
- **Deep Learning Advantage** – As a deep learning model, LSTM can automatically learn complex patterns in text without manual feature engineering.



```
# Build LSTM model
model = Sequential()
model.add(Embedding(max_words,64 ))
model.add(SpatialDropout1D(0.2))
model.add(LSTM(32, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))  # For binary classification

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
# Train model
history = model.fit(X_train, y_train, epochs=4, batch_size=128, validation_split=0.2)

# Evaluate
y_pred = model.predict(X_test)
y_pred_classes = (y_pred > 0.5).astype(int)
```

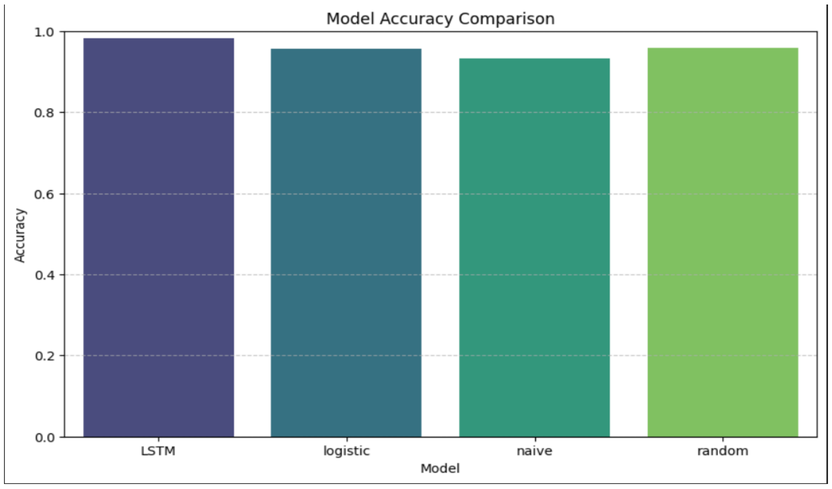
9. Visualization of Results & Model Insights

1. Accuracy Comparison (Bar Plot)

A bar plot was generated to compare the accuracy of the LSTM, Logistic Regression, Naive Bayes, and Random Forest models.

Insight:

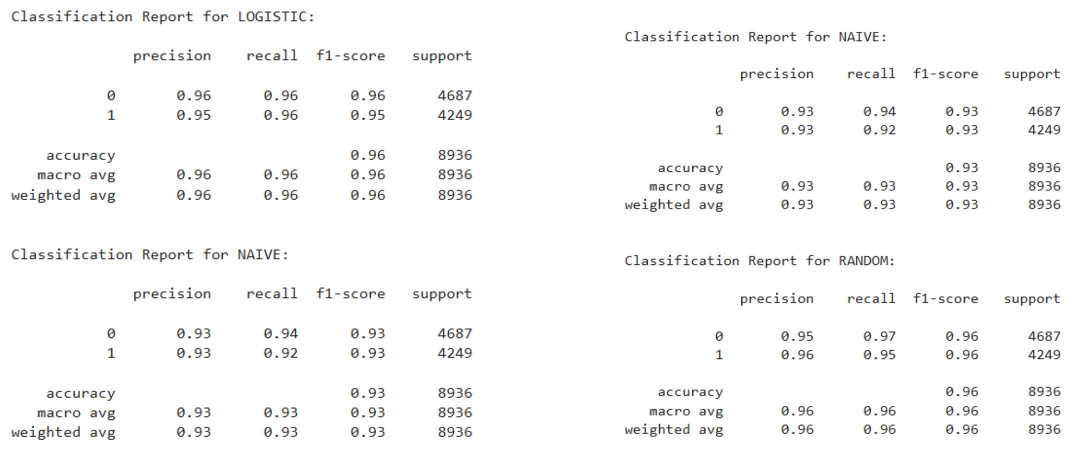
The LSTM model achieved the highest accuracy (0.98), indicating superior performance over traditional machine learning models. Logistic Regression and Random Forest followed closely with 0.96, while Naive Bayes had the lowest at 0.93.



2. Classification Report (Printed Table)

For each model, the full classification report was printed, showing precision, recall, and F1-score for both classes.

Insight:
These metrics provide a more granular view beyond accuracy. For instance, LSTM shows a nearly perfect balance between precision and recall for both classes, indicating strong classification performance.



```
Classification Report for LOGISTIC:

              precision    recall  f1-score   support

           0       0.96      0.96      0.96      4687
           1       0.95      0.96      0.95      4249

    accuracy                           0.96      8936
   macro avg       0.96      0.96      0.96      8936
weighted avg       0.96      0.96      0.96      8936
```

```
Classification Report for NAIVE:

              precision    recall  f1-score   support

           0       0.93      0.94      0.93      4687
           1       0.93      0.92      0.93      4249

    accuracy                           0.93      8936
   macro avg       0.93      0.93      0.93      8936
weighted avg       0.93      0.93      0.93      8936
```

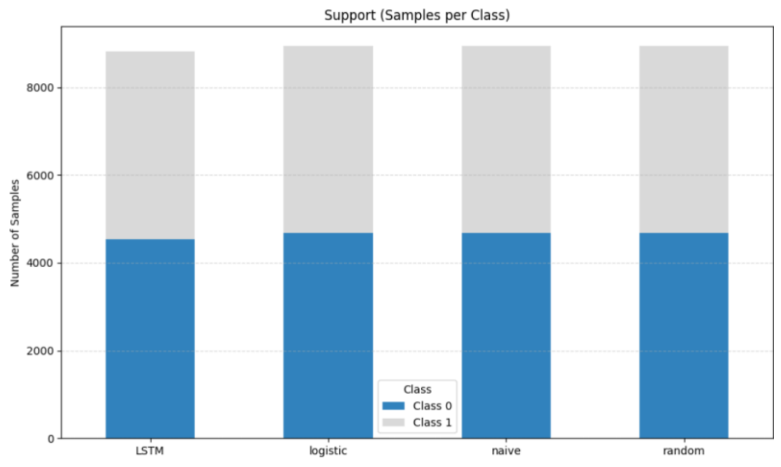
```
Classification Report for NAIVE:

              precision    recall  f1-score   support

           0       0.93      0.94      0.93      4687
           1       0.93      0.92      0.93      4249

    accuracy                           0.93      8936
   macro avg       0.93      0.93      0.93      8936
weighted avg       0.93      0.93      0.93      8936
```

```
Classification Report for RANDOM:

              precision    recall  f1-score   support

           0       0.95      0.97      0.96      4687
           1       0.96      0.95      0.96      4249

    accuracy                           0.96      8936
   macro avg       0.96      0.96      0.96      8936
weighted avg       0.96      0.96      0.96      8936
```

3. Class Support Distribution (Stacked Bar Plot)

A stacked bar plot visualized the number of samples (support) per class for each model's test predictions.

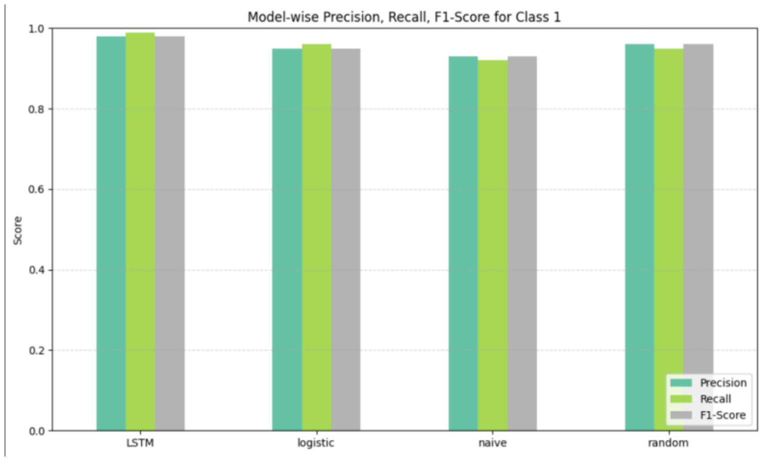
Insight:
The class distribution was relatively balanced across all models, ensuring no class imbalance bias. Each model processed similar numbers of instances in Class 0 and Class 1.



4. Precision, Recall, F1-Score (Bar Plot for Class 1)

A grouped bar chart was used to compare **precision, recall, and F1-score** specifically for **Class 1** across all models.

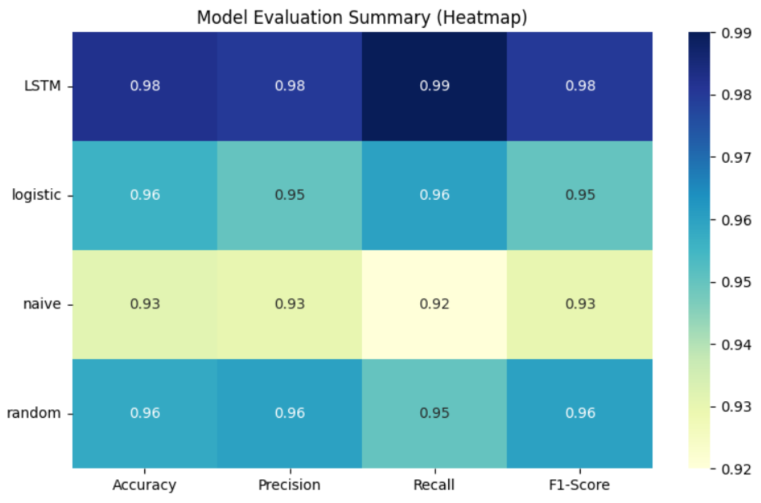
Insight:
This plot highlights model effectiveness in identifying the positive class. LSTM consistently performed best, followed by Random Forest. Naive Bayes lagged behind slightly, especially in recall.



5. Heatmap: Model Evaluation Summary

A heatmap was generated to show a quick comparative summary of all key metrics (Accuracy, Precision, Recall, F1-Score) for each model.

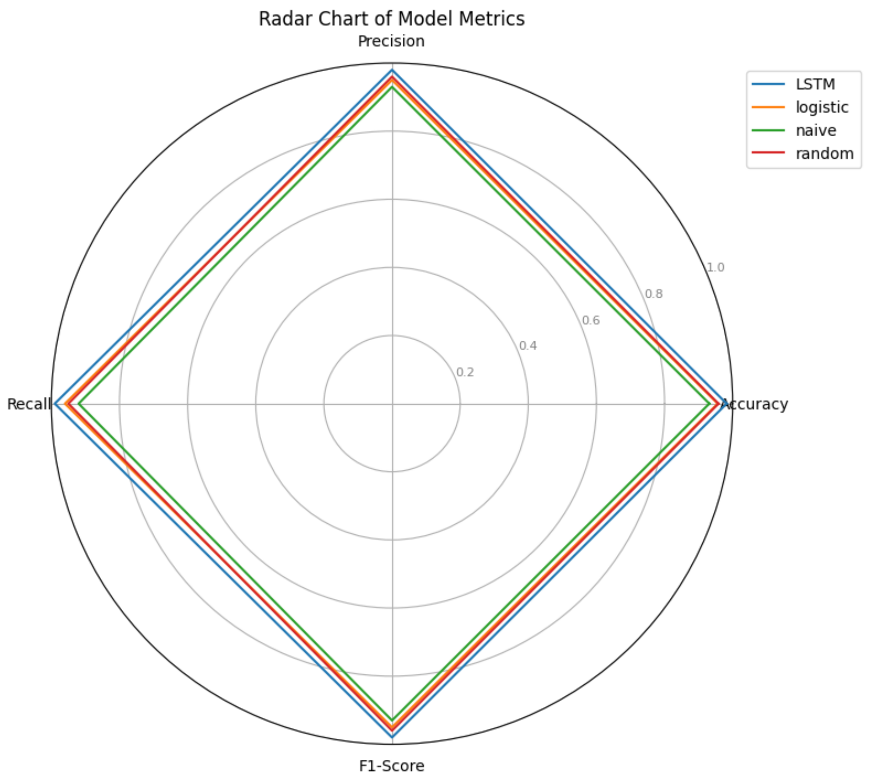
Insight:
The heatmap visually reinforces that LSTM is the top performer across all metrics. It also shows that Random Forest and Logistic Regression perform similarly, whereas Naive Bayes is less optimal.



6. Radar Chart (Polar Plot of Model Metrics)

A radar chart was plotted to visualize the overall performance of each model across all four metrics simultaneously.

Insight:
The shape and spread of each radar polygon illustrate how balanced or skewed a model is across evaluation metrics. LSTM's polygon forms a large, nearly circular shape—indicating well-rounded performance. Naive Bayes, however, shows a more irregular shape due to comparatively lower scores.



9.Tools and Technologies Used

- **Programming Language:** Python 3.8+
- **IDE/Notebook:** Jupyter Notebook
- **NLP Libraries:** NLTK, spaCy, Hugging Face Transformers
- **Data Handling:** Pandas, NumPy
- **Visualization:** Matplotlib, Seaborn

10. Team Members and Contributions

Bhavadharani M

- Co-leads data collection and cleaning.
- Writes sections of the final project report.
- Participates in model evaluation and visualization.

Dhanya P

- Manages the GitHub repo and version control.
- Contributes to model training.
- Coordinates team meetings and milestone check-ins.

Gayathri S

- Co-develops ML/NLP models.
- Helps with project documentation and dataset description.
- Helps create evaluation metrics and results tables.

Rinchin Pelton

- Works on text preprocessing and tokenization.
- Integrates modules from all team members.
- Assists in writing code explanations and documentation.

Varshini R

- Assists in hyperparameter tuning and training scripts.
- Leads testing and validation (cross-validation, confusion matrix, etc.).
- Contributes to performance analysis and interprets model output.