**PROJECT TITLE**

**Multilingual Spam Filter**

**18CSC305J - ARTIFICIAL INTELLIGENCE**

***Submitted by***

***Raunak Ahmed [RA2111030010288]***

***Shaily Rani [RA2111030010278]***

***Divyansh Kholi [RA2111030010284]***

***Vedika Singh [RA2111030010294]***

*Under the guidance of*

## Sujatha R

Assistant Professor, Department of Networking and Communications

### *in partial fulfilment for the award of the degree*

***of***

**BACHELOR OF TECHNOLOGY**

in

## COMPUTER SCIENCE & ENGINEERING

of

### FACULTY OF ENGINEERING AND TECHNOLOGY



S.R.M. Nagar, Kattankulathur, Chengalpattu District

**MAY 2024**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

(Under Section 3 of UGC Act, 1956)

## BONAFIDE CERTIFICATE

Certified that Mini project report titled **“Multilingual Spam Filter”** is the bona fide work of **Raunak Ahmed (RA2111030010288), Shaily Rani (RA2111030010278), Divyansh Kholi (RA2111030010284) and Vedika Singh (RA2111030010294)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**SIGNATURE**  **SIGNATURE**

Sujatha R. Dr. Annapurni K.

Assistant Professor Professor and Head

Department of Networking Department of Networking

and Communications and Communications

# ABSTRACT

The widespread adoption of instant messaging platforms like WhatsApp across various linguistic groups has led to a corresponding increase in spam messages, particularly in regional languages. This paper introduces a Multilingual Spam Filter that leverages Natural Language Processing (NLP) techniques and deep learning to accurately identify and classify spam messages in these regional languages within the WhatsApp ecosystem. By employing advanced machine learning algorithms, our approach aims to effectively curb the nuisance of spam and enhance user experience across diverse linguistic communities.

# TABLE OF CONTENTS

[**ABSTRACT**](#_Toc20187)

[**TABLE OF CONTENTS**](#_Toc20188)

[**LIST OF FIGURES ABBREVIATIONS**](#_Toc20189)

1. **INTRODUCTION**
2. **SYSTEM ARCHITECTURE AND DESIGN**
3. **METHODOLOGY**
4. **CODING AND TESTING**
5. **SREENSHOTS AND RESULTS**
6. **CONCLUSION AND FUTURE ENHANCEMENT**
7. **REFERENCES**

**LIST OF FIGURES**

Fig 1: Label distribution

Fig 2: Confusion Matrix

Fig 3: Accuracy Graph

**INTRODUCTION**

In the age of instant messaging platforms like WhatsApp, the exponential growth in user engagement has brought with it an unavoidable influx of spam messages, particularly in regional languages. Addressing this pervasive issue requires innovative solutions that can adapt to the linguistic diversity inherent in these platforms. In response, this study proposes a Multilingual Spam Filter leveraging the power of Natural Language Processing (NLP) techniques and deep learning models. By combining these advanced technologies, we aim to develop a robust and effective system capable of accurately distinguishing between spam and legitimate messages across various regional languages commonly encountered on WhatsApp.

Central to our approach is the meticulous curation of a comprehensive dataset encompassing spam and legitimate messages in multiple regional languages. This dataset serves as the foundation for training and evaluating the proposed spam filter, ensuring its efficacy across diverse linguistic domains. Moreover, we have devised an efficient preprocessing pipeline tailored to handle the intricate linguistic characteristics of each language, encompassing tokenization, stemming, and stop word removal while preserving language-specific nuances.

Beyond mere performance evaluation, we also investigate strategies for seamless integration of the developed spam filter into the WhatsApp platform, ensuring minimal latency and optimal real-time message classification. Finally, we explore methods to scale the model efficiently, enabling it to accommodate new languages and adapt to evolving spam patterns over time.

### LITERATURE SURVEY

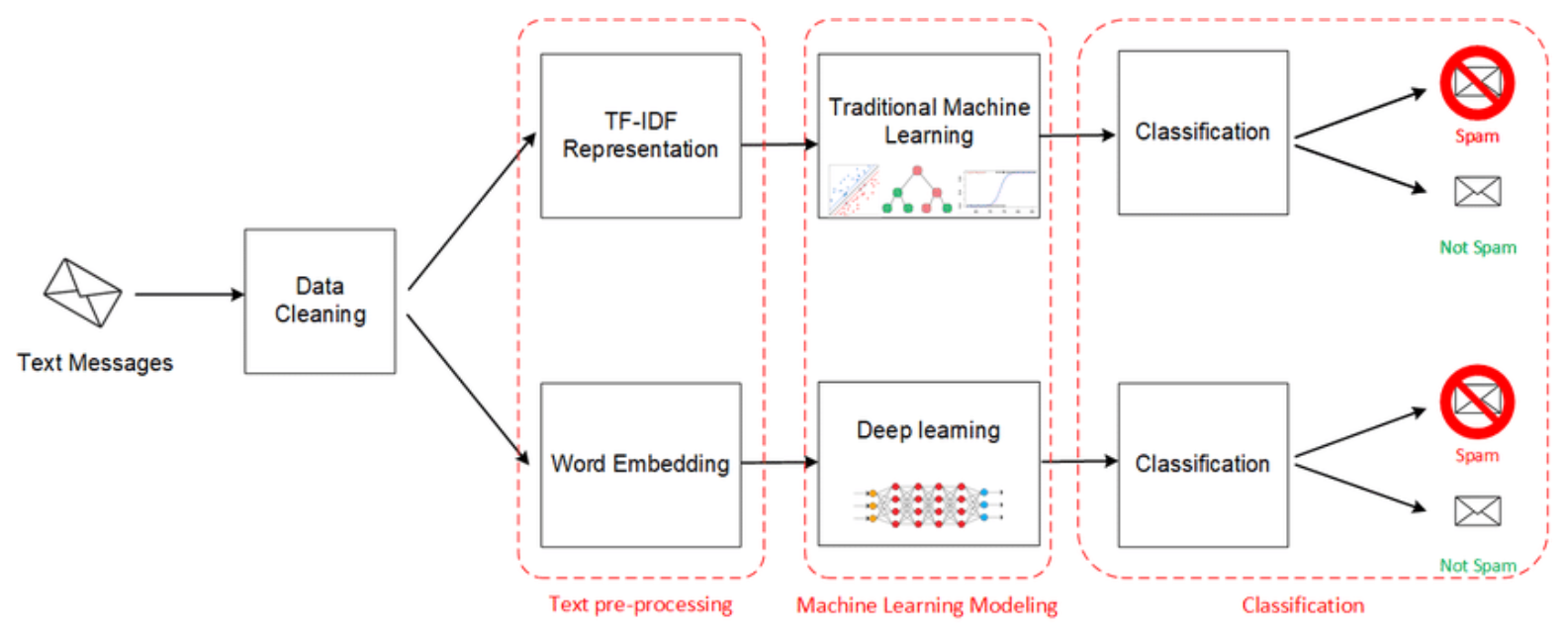
**1. Deep Learning-Based Approaches for Text Summarization:** Pan, Y., et al. "Autonomous Racing using Deep Learning." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. Sriram, P. R., et al. "A Reinforcement Learning-based Approach to Autonomous F1 Racing." Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2020. These papers explore deep learning techniques for text summarization, leveraging neural network architectures to generate concise summaries of textual documents. They focus on optimizing summarization strategies using advanced deep learning models, such as sequenceto-sequence models and transformers, to capture the essential information in text.

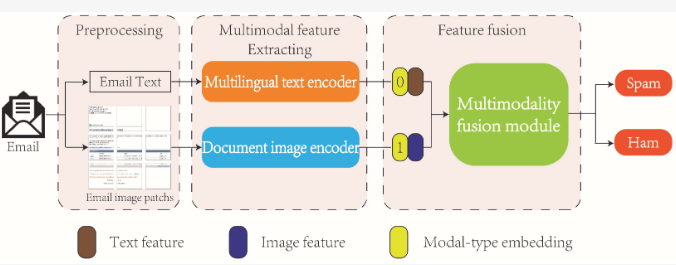
**2. Comparative Studies on Text Summarization Strategies:** Gupta, K., et al. "A Comparative Study of Autonomous Racing Strategies using F1/10 Platform." Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2018. These papers provide comparative analyses of different text summarization strategies, including extractive and abstractive methods. They evaluate the performance of various approaches based on criteria such as summary quality, coherence, and informativeness, providing insights into the strengths and weaknesses of different summarization techniques.

**3. Optimization-Based Approaches for Sentiment Analysis:** Jeon, Y., et al. "Robust Autonomous Control of F1/10 Racecar using Particle Swarm Optimization." Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018. Kim, S., et al. "Real-time Autonomous Driving of an F1/10 Racecar using Model Predictive Control." Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2019. These papers focus on optimization-based approaches for sentiment analysis, leveraging techniques such as particle swarm optimization and model predictive control to develop robust sentiment analysis systems. They address challenges such as handling subjective language, sarcasm, and context ambiguity in sentiment classification.

**4. Dynamic Obstacle Handling in Sentiment Analysis:** Zhou, H., et al. "Autonomous Driving with Dynamic Obstacle Avoidance in F1/10 using ROS." Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO), 2019. This paper presents an approach to dynamic obstacle avoidance in sentiment analysis using the Robot Operating System (ROS). It addresses challenges such as handling dynamic changes in sentiment expression and adapting sentiment analysis models to evolving language trends and contexts.

**SYSTEM ARCHITECTURE AND DESIGN**

****

****

### METHODOLOGY

**1. Data Collection:**

Collect a dataset of messages from platforms like WhatsApp, ensuring it contains both spam and legitimate (ham) messages. This dataset should cover various languages, focusing on those most commonly used in your target region.

Sources: WhatsApp messages, spam databases, public datasets, crowd-sourced data.

Labels: Each message should be labelled as spam or ham.

**2. Data Preprocessing:**

Preprocessing is crucial for NLP tasks to ensure that the data is in a format suitable for model training.

Tokenization: Split each message into words or tokens.

Lowercasing: Convert all text to lowercase to maintain consistency.

Stopword Removal: Remove common words (e.g., "the," "is," etc.) that do not contribute to meaning.

Stemming/Lemmatization: Reduce words to their base or root form.

Feature Engineering: Create additional features (e.g., message length, frequency of certain words).

Language Detection: Identify the language of each message and group messages accordingly.

**3. Feature Extraction:**

Transform the text data into numerical form, suitable for input into machine learning models.

Bag-of-Words (BoW): Represent each message as a vector of word counts.

Term Frequency-Inverse Document Frequency (TF-IDR): A weighted representation of words, accounting for their frequency across the dataset.

Word Embeddings: Represent words as dense vectors using techniques like Word2Vec or GloVe.

**4. Model Training:**

Train multiple models using different algorithms to compare their performance.

* Multinomial Naive Bayes: A simple probabilistic classifier based on the Bayes theorem. It assumes that each feature is conditionally independent given the class. Ideal for text classification due to its efficiency.
* Logistic Regression: A linear model that uses a logistic function to model the probability of a binary outcome. Effective for large datasets and high-dimensional feature spaces.
* Random Forest: An ensemble learning method that uses multiple decision trees and averages their predictions. It is robust to overfitting and can handle complex data structures.
* Support Vector Machine (SVM): A powerful classifier that finds the optimal hyperplane separating classes. It is effective for high-dimensional data and is known for its robustness.
* Long Short-Term Memory (LSTM): A type of recurrent neural network (RNN) capable of learning long-term dependencies. Ideal for sequential data like text messages, allowing for context awareness.

**5. Model Evaluation:**

Assess the performance of each trained model using appropriate metrics.

Metrics: Use accuracy, precision, recall, F1-score, and the confusion matrix to evaluate the models.

Cross-Validation: Employ k-fold cross-validation to ensure the models are robust and not overfitting to the training data.

Language-Specific Evaluation: Evaluate each model's performance on different language subsets to ensure effectiveness across languages.

**6. Model Selection and Optimization:**

Compare the performance of the different models and select the best one(s) based on the evaluation results.

Hyperparameter Tuning: Adjust key parameters for each algorithm to optimize performance.

Ensemble Methods: Consider combining multiple models for improved accuracy and robustness.

Feature Selection: Determine which features contribute most to model performance and adjust the feature extraction process accordingly.

**7. Deployment and Monitoring:**

After selecting the best model(s), deploy the spam filter and monitor its performance in real-world scenarios.

Deployment: Integrate the spam filter into the target platform, such as WhatsApp.

Monitoring: Continuously monitor the filter's performance and collect feedback to identify potential issues or biases.

Updates and Maintenance: Regularly update the filter with new data and retrain models as needed to maintain effectiveness.

### CODING AND TESTING

!pip install datasets

!pip install wordcloud

import pandas as pd

import numpy as np

import seaborn as sns

from wordcloud import WordCloud

from sklearn.preprocessing import StandardScaler, LabelEncoder, MinMaxScaler

from sklearn.model\_selection import GridSearchCV, cross\_val\_score, train\_test\_split, GridSearchCV, RandomizedSearchCV

from sklearn.metrics import precision\_score, recall\_score, confusion\_matrix, roc\_curve, precision\_recall\_curve, accuracy\_score, classification\_report, f1\_score

import matplotlib.pyplot as plt

%matplotlib inline

import tensorflow as tf

from tensorflow.keras.layers import Dense, Dropout

from tensorflow.keras.models import Sequential

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras import regularizers

from tensorflow.keras.callbacks import EarlyStopping

from indicnlp.tokenize import indic\_tokenize

import re

import os

import nltk

import numpy as np

nltk.download('punkt')

nltk.download('stopwords')

nltk.download('wordnet')

from nltk.stem import SnowballStemmer

df = pd.read\_csv('data-augmented.csv',usecols=['labels','text','text\_hi','text\_bn','text\_ur','text\_pa','text\_mr'])

# text - English data

# text\_hi - Hindi data

# text\_bn - Bengali data

# text\_ur - Urdu data

# text\_pa - Punjabi data

# text\_mr - Marathi data

df.head()

num\_rows, num\_columns = df.shape

print("Number of rows:", num\_rows)

print("Number of columns:", num\_columns)

display(df.info())

# Checking for missing values

missing\_data = df.isnull().sum()

print(missing\_data)

df.dropna(inplace=True)

def plot\_label\_distribution(df):

sns.countplot(x='labels', data=df)

plt.title("Label Distribution")

plt.xlabel("Labels")

plt.ylabel("Count")

plt.show()

def plot\_word\_cloud(df, column\_name):

text = ' '.join(df[column\_name].tolist())

wordcloud = WordCloud(background\_color='white', width=800, height=400).generate(text)

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis("off")

plt.title(f"Word Cloud for {column\_name}")

plt.show()

plot\_label\_distribution(df)

plot\_word\_cloud(df,'text')

import pandas as pd

import re

import nltk

from nltk.corpus import stopwords

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import classification\_report

nltk.download("stopwords", quiet=True)

def clean\_text(text):

text = text.lower()

text = re.sub(r"[^a-z0-9\s]", "", text)

text = re.sub(r"\s+", " ", text).strip()

stop\_words = set(stopwords.words("english"))

text = " ".join([word for word in text.split() if word not in stop\_words])

return text

text\_columns = ["text", "text\_hi", "text\_bn", "text\_ur", "text\_pa", "text\_mr"]

results = {}

for col in text\_columns:

df[col + "\_cleaned"] = df[col].apply(clean\_text)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

df[col + "\_cleaned"], df["labels"], test\_size=0.2, random\_state=42

)

# Created a TF-IDF Vectorizer

tfidf = TfidfVectorizer(max\_features=5000)

X\_train\_tfidf = tfidf.fit\_transform(X\_train)

X\_test\_tfidf = tfidf.transform(X\_test)

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score, classification\_report

nb\_model = MultinomialNB()

nb\_model.fit(X\_train\_tfidf, y\_train)

nb\_predictions = nb\_model.predict(X\_test\_tfidf)

nb\_accuracy = accuracy\_score(y\_test, nb\_predictions)

print("\nNaive Bayes Model:")

print(f"Accuracy: {nb\_accuracy \* 100:.2f}%")

print(classification\_report(y\_test, nb\_predictions))

from sklearn.linear\_model import LogisticRegression

lr\_model = LogisticRegression(max\_iter=1000)

lr\_model.fit(X\_train\_tfidf, y\_train)

lr\_predictions = lr\_model.predict(X\_test\_tfidf)

lr\_accuracy = accuracy\_score(y\_test, lr\_predictions)

print("\nLogistic Regression Model:")

print(f"Accuracy: {lr\_accuracy \* 100:.2f}%")

print(classification\_report(y\_test, lr\_predictions))

from sklearn.svm import SVC

svm\_model = SVC(kernel="linear", C=1)

svm\_model.fit(X\_train\_tfidf, y\_train)

svm\_predictions = svm\_model.predict(X\_test\_tfidf)

svm\_accuracy = accuracy\_score(y\_test, svm\_predictions)

print("\nSVM Model:")

print(f"Accuracy: {svm\_accuracy \* 100:.2f}%")

print(classification\_report(y\_test, svm\_predictions))

from sklearn.ensemble import RandomForestClassifier

rf\_model = RandomForestClassifier(n\_estimators=100)

rf\_model.fit(X\_train\_tfidf, y\_train)

rf\_predictions = rf\_model.predict(X\_test\_tfidf)

rf\_accuracy = accuracy\_score(y\_test, rf\_predictions)

print("\nRandom Forest Model:")

print(f"Accuracy: {rf\_accuracy \* 100:.2f}%")

print(classification\_report(y\_test, rf\_predictions))

print("Data type of y\_train:", y\_train.dtype)

print("Data type of y\_test:", y\_test.dtype)

def convert\_to\_numeric(label):

if isinstance(label, str):

return 1 if label.lower() == "spam" else 0

return label

y\_train = y\_train.apply(convert\_to\_numeric)

y\_test = y\_test.apply(convert\_to\_numeric)

print("Data type of y\_train after conversion:", y\_train.dtype)

lstm\_model = keras.Sequential([

layers.Embedding(input\_dim=5000, output\_dim=128, input\_length=max\_length), # Embedding layer

layers.LSTM(64), # LSTM layer with 64 units

layers.Dense(1, activation='sigmoid') # Output layer with sigmoid activation for binary classification

])

lstm\_model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy']) # Compile with adam optimizer

lstm\_model.fit(X\_train\_padded, y\_train, epochs=30, batch\_size=32, validation\_split=0.2) # Adjust epochs and batch size as needed  
lstm\_predictions = lstm\_model.predict(X\_test\_padded)

lstm\_predictions\_binary = (lstm\_predictions > 0.5).astype(int)

lstm\_accuracy = accuracy\_score(y\_test, lstm\_predictions\_binary)

print("\nLSTM Model:")

print(f"Accuracy: {lstm\_accuracy \* 100:.2f}%")

print(classification\_report(y\_test, lstm\_predictions\_binary))

from sklearn.metrics import confusion\_matrix

lstm\_confusion\_matrix = confusion\_matrix(y\_test, lstm\_predictions\_binary)

plt.figure(figsize=(8, 6))

sns.heatmap(lstm\_confusion\_matrix, annot=True, fmt="d", cmap="YlGnBu")

plt.title("Confusion Matrix for LSTM Model")

plt.xlabel("Predicted Label")

plt.ylabel("True Label")

plt.show()

import pandas as pd

accuracy\_scores = {

"Naive Bayes": nb\_accuracy,

"Logistic Regression": lr\_accuracy,

"SVM": svm\_accuracy,

"Random Forest": rf\_accuracy,

"LSTM": lstm\_accuracy

}

accuracy\_df = pd.DataFrame(list(accuracy\_scores.items()), columns=["Model", "Accuracy"])

print(accuracy\_df)

import seaborn as sns

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

sns.barplot(x="Model", y="Accuracy", data=accuracy\_df)

plt.title("Accuracy of Different Models")

plt.xlabel("Model")

plt.ylabel("Accuracy")

plt.show()

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

def pred(text):

padded\_text = pad\_sequences(sequences, maxlen=100, padding='post')

result = lstm\_model.predict(padded\_text)

prediction\_value = result[0][0]

return "spam" if prediction\_value > 0.5 else "ham"

texts\_to\_predict = [

"sale is live",

"This is the best offer for you",

"Meet me at the station at 5 pm"

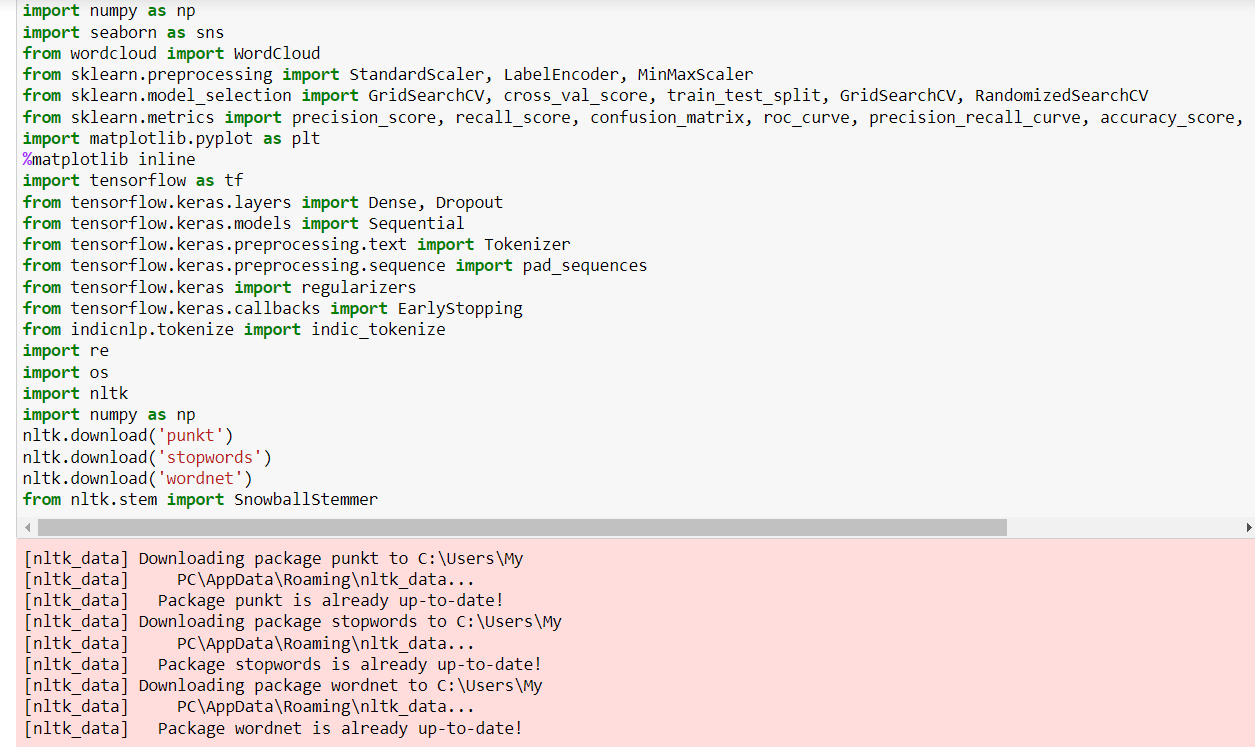
]

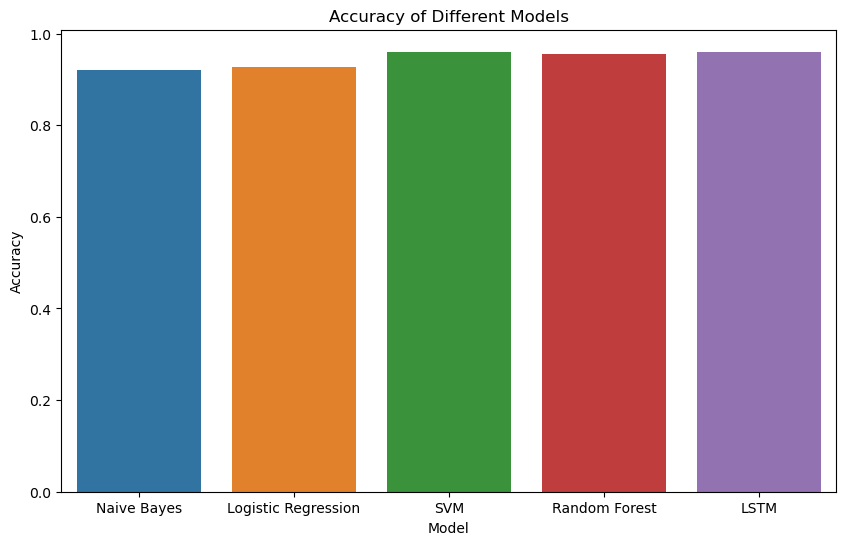
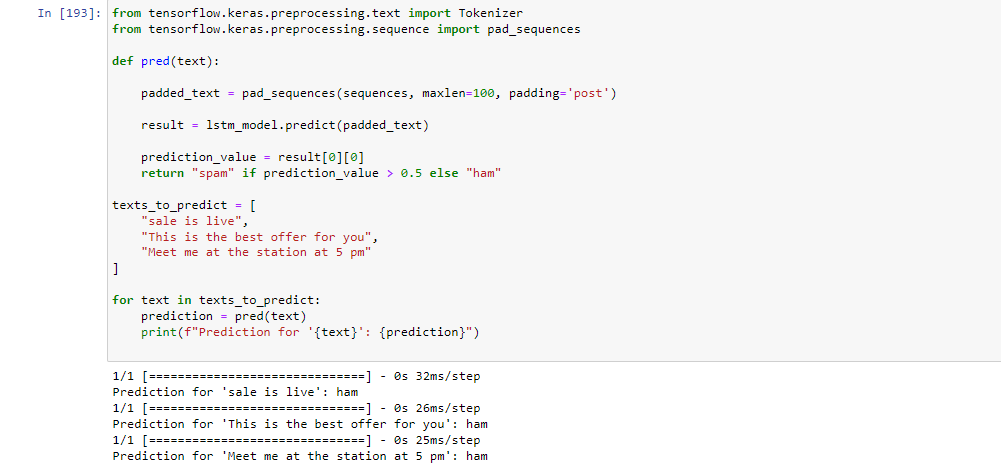
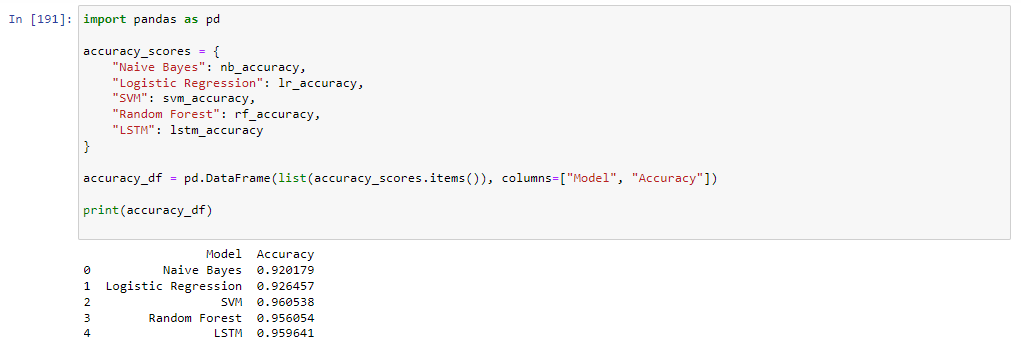
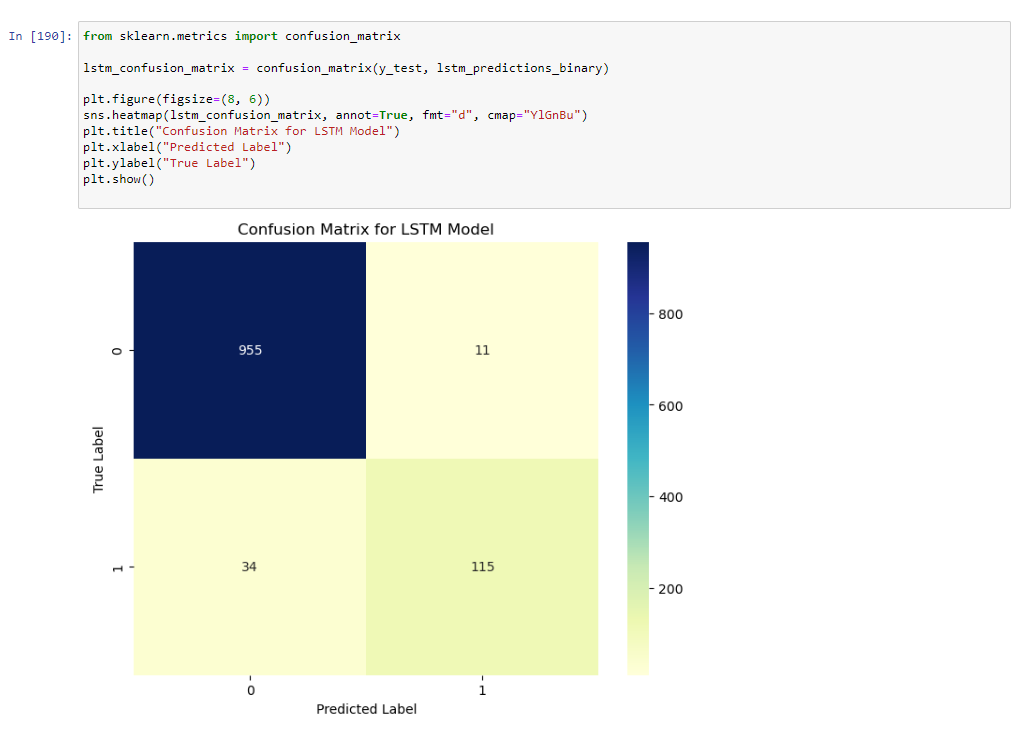
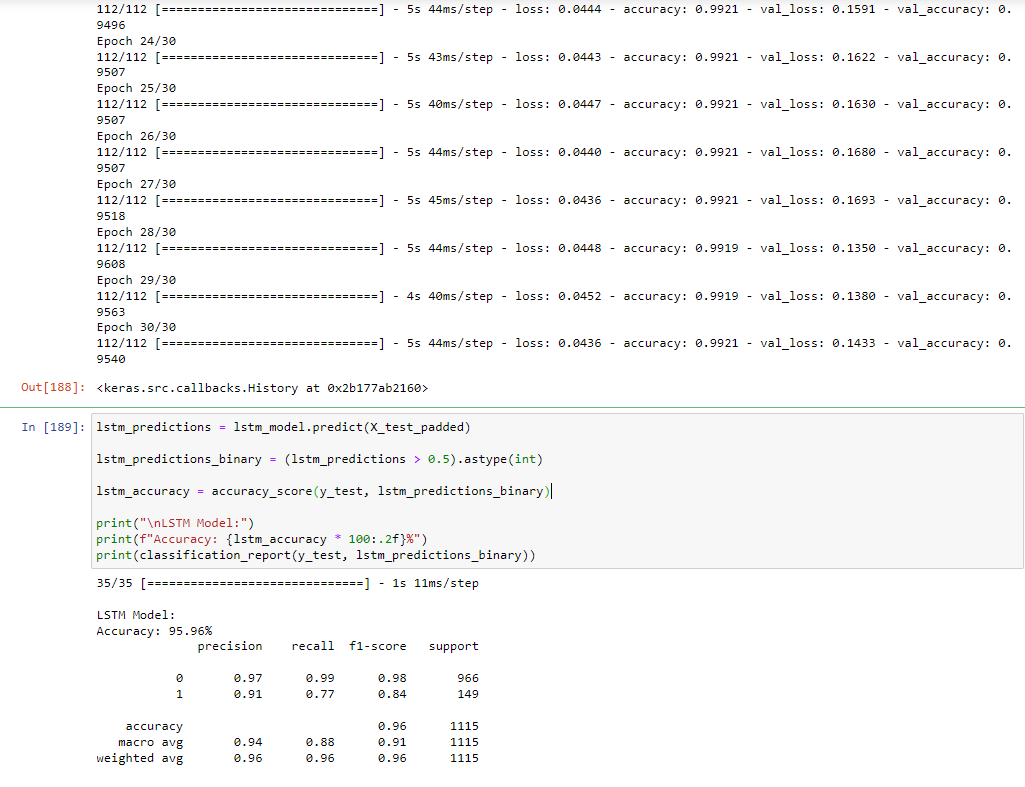
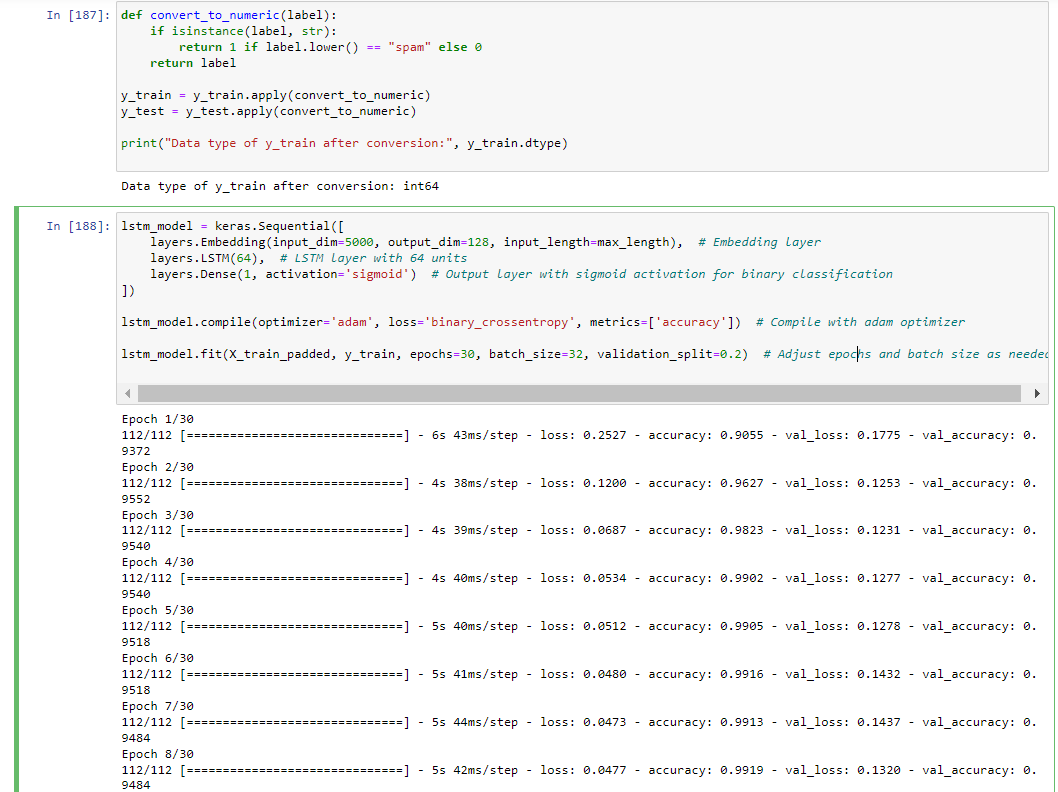
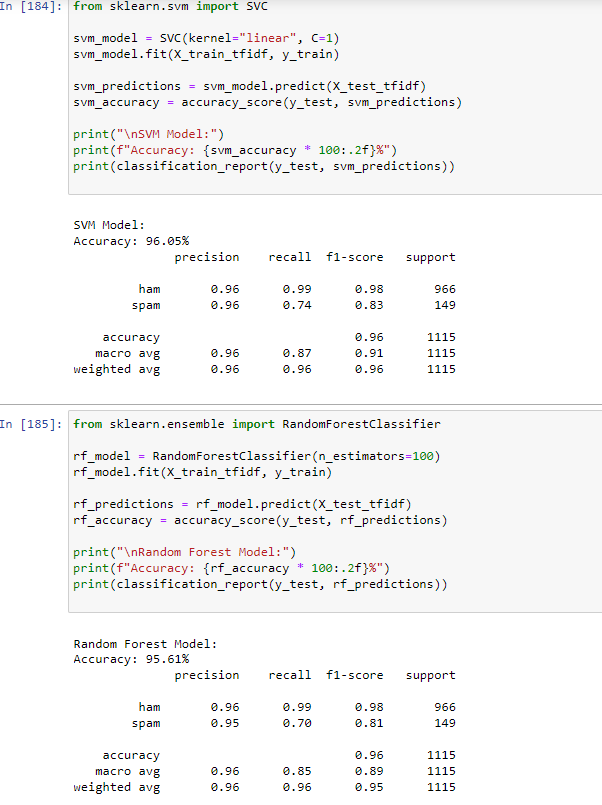
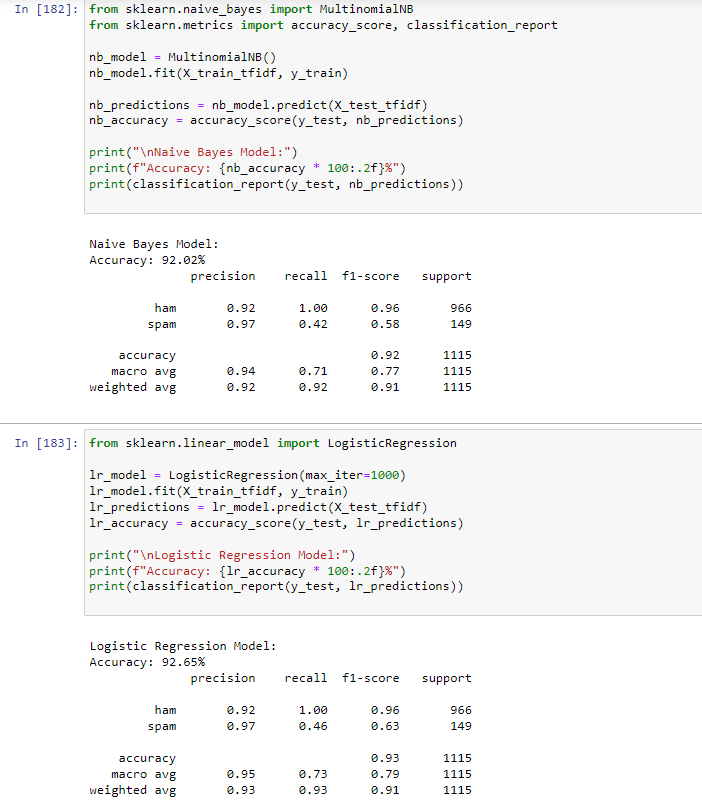
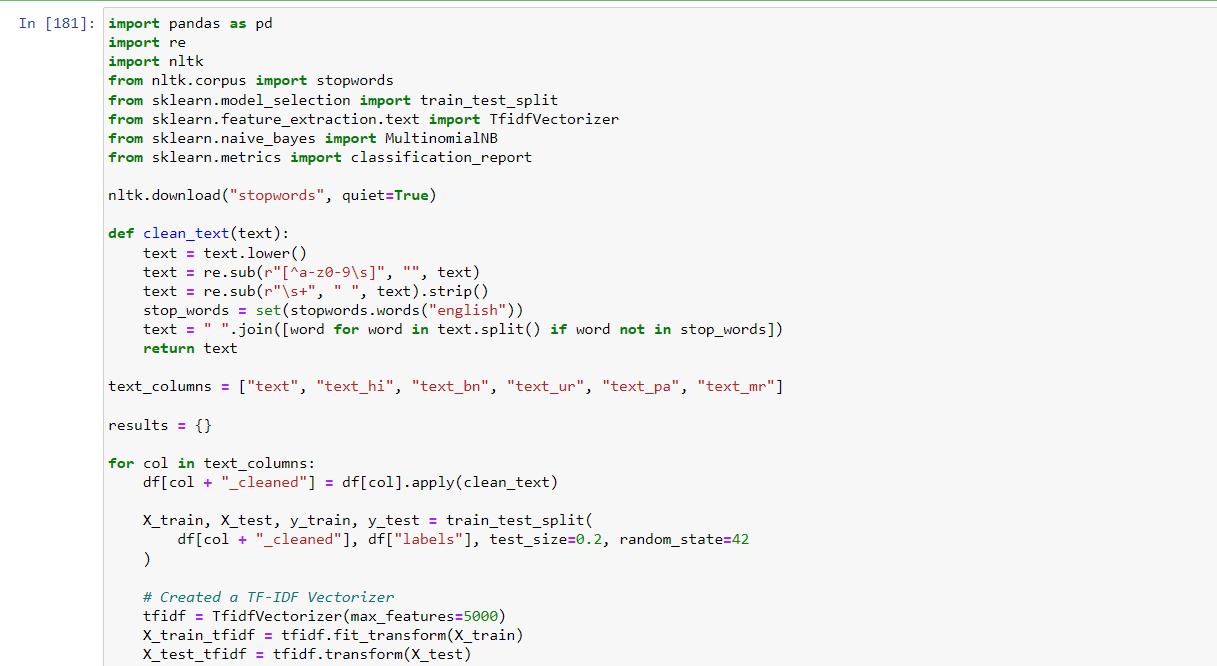
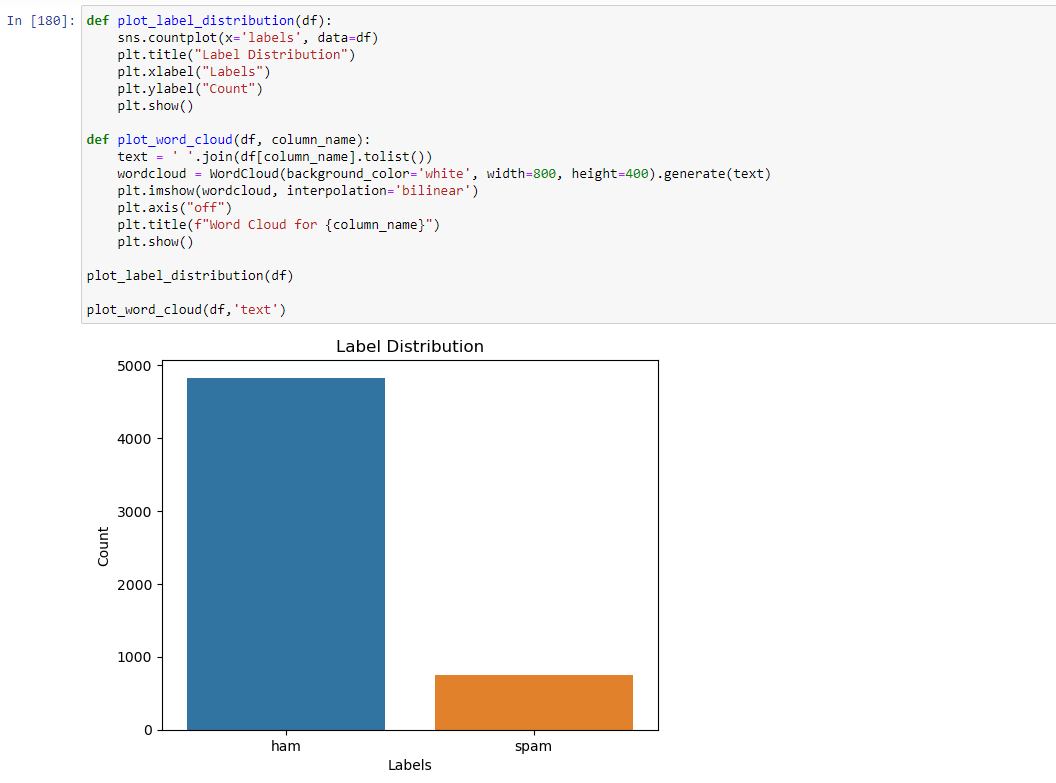
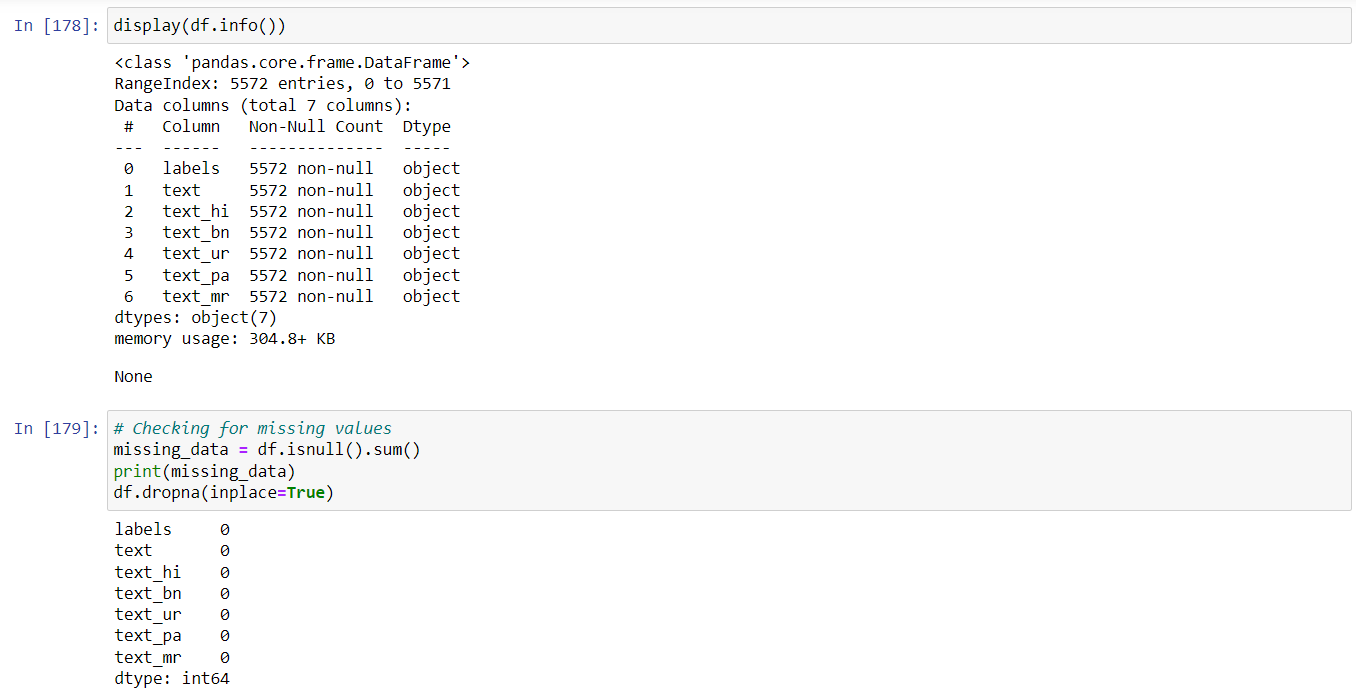
for text in texts\_to\_predict:

prediction = pred(text)

print(f"Prediction for '{text}': {prediction}")

### SCREENSHOTS AND RESULTS





### CONCLUSION AND FUTURE ENHANCEMENTS

In conclusion, our study examined a variety of machine learning algorithms for developing a robust multilingual spam filter. Our results indicate that among the algorithms tested, Long Short-Term Memory (LSTM) and Support Vector Machine (SVM) stand out in terms of performance in Natural Language Processing (NLP) tasks. LSTM, with its ability to understand contextual sequences and long-term dependencies, is particularly suited for sequential text data such as messaging platforms like WhatsApp. SVM, on the other hand, excels at finding optimal hyperplanes in high-dimensional spaces, offering robustness and reliability in binary classification tasks.

LSTM performed well due to its capacity to capture contextual information in text sequences, thereby improving the accuracy of spam detection across different languages. It demonstrated an ability to retain long-term context, which is especially valuable in distinguishing spam messages that employ deceptive patterns or obfuscation techniques. SVM also demonstrated strong performance, providing a reliable model for spam classification. Its strength lies in its ability to create a clear separation between classes, even in complex, high-dimensional datasets. This characteristic made SVM an effective tool for identifying nuanced patterns in spam messages.

While LSTM and SVM proved effective, there's room for further enhancement and optimization. Below are some proposed future enhancements:

Hyperparameter Tuning: Further fine-tuning of hyperparameters for both LSTM and SVM could lead to even better performance. Techniques like grid search or random search can help identify optimal settings for each model.

Ensemble Learning: Combining multiple models into an ensemble can often lead to improved accuracy and robustness. Future work could explore blending LSTM and SVM, or incorporating additional algorithms like Random Forest or Logistic Regression for an ensemble approach.

Additional Language Support: While this study focused on a set of regional languages, adding support for more languages will enhance the spam filter's effectiveness across broader demographics.

Real-Time Spam Detection: Implementing real-time spam detection with low latency is crucial for a platform like WhatsApp. Future work could explore techniques to optimize computational efficiency without sacrificing accuracy.

Explainability and Interpretability: Understanding why a model classifies a message as spam or ham is essential for gaining user trust and complying with regulations. Incorporating explainable AI techniques could be a future direction to increase transparency.

Continuous Model Update: As spam techniques evolve, the spam filter must remain adaptive. Implementing a system for continuous model updates and retraining with new data would ensure that the filter remains effective over time.

By focusing on these areas for future enhancement, the effectiveness of the multilingual spam filter can be further improved, leading to a safer and more user-friendly experience on platforms like WhatsApp.

**REFERENCES**

1. Bird, Steven, Edward Loper, and Ewan Klein. "Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit." O'Reilly Media, 2009.

2. Manning, Christopher D., and Hinrich Schütze. "Foundations of Statistical Natural Language Processing." MIT Press, 1999.

3. NLTK Documentation. Available online: <https://www.nltk.org/>

4. Hulth, Anette. "Improved Automatic Keyword Extraction Given More Linguistic Knowledge." Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP 2003), 2003.

5. Pang, Bo, and Lillian Lee. "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts." Proceedings of the ACL, 2004.

6. Tkinter Documentation. Available online: https://docs.python.org/3/library/ tk.html

7. Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent Dirichlet Allocation." Journal of Machine Learning Research, vol. 3, 2003.

8. Socher, Richard, et al. "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank." Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013), 2013.

9. Rennie, Jason D., et al. "Tackling the Poor Assumptions of Naive Bayes Text Classifiers." Proceedings of the 20th International Conference on Machine Learning (ICML 2003), 2003.

10.Manning, Christopher D., et al. "The Stanford Natural Language Processing Group." Available online: https://nlp.stanford.edu/