“Detecting Spam Emails”

Submitted in partial fulfillment of the requirements

For the degree of

Bachelor of Technology

Computer Science & Engineering Department

Submitted By

Garikipati Abhiram

A70405222102

Under the Guidance of

Dr. Swetta Kukreja

AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY

AMITY UNIVERSITY MUMBAI

2024-25

Certificate

This is to certify that the mini-project entitled “Detecting Spam Emails” is a bonafied work of G.Abhiram submitted to the Amity School of Engineering and Technology, Amity University Mumbai in partial fulfilment of the requirement for the degree of Bachelor of Technology.

Dr. Deepa Parasar

Supervisor & Department Coordinator

Dr. Shrikant Charhate

Director, ASET

Approval

This is to certify that G.Abhiram in has satisfactorily completed his mini-project in Detecting Spam Emails Machine Learning on during the academic term 2024-25 and their report is approved for final submission.

Examiners

Dr. Swetta Kukreja

Date

Place

Declaration of Academic Integrity

I declare that this written submission conveys my ideas in my own words. I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/date/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the institute and the can evoke penal action form the sources which have thus not been properly cited or from whom proper permission has not been taken when needed

Name, Enrolment no.

Date

G.Abhiram

A70405222102

Acknowledgement

I would like to express my heartfelt gratitude to all those who have guided and supported me throughout this project.

First and foremost, I would like to thank my college and the Department of Computer Science Engineering for providing me with the opportunity and resources to undertake this project on Machine Learning for Loan Status Prediction. Special thanks go to my project guide, **Dr.Swetta Kukreja**, for their invaluable guidance, support, and encouragement at every stage of this project.

I am deeply indebted to my teachers, whose lectures and insights have greatly enhanced my understanding of artificial intelligence and machine learning concepts, laying the foundation for this work.

I would also like to acknowledge the contributions of my peers and colleagues, whose discussions and feedback enriched this project. Additionally, I extend my gratitude to my family and friends for their constant encouragement and support throughout the project timeline.

Finally, I am thankful to the online resources, tutorials, and research papers that have served as a critical reference in implementing and understanding the ML model for this project.

Thank you all for your support and inspiration.

Name, Enrolment no.

G.Abhiram, A70405222102

INDEX

Introduction

Spam email detection is a critical task in modern communication systems. The vast amount of unsolicited, often irrelevant, and potentially harmful content received through email can overwhelm users and create security risks. Spam emails may include unwanted advertisements, phishing attempts, or even malware, making it essential to develop efficient systems for filtering out such messages.

In recent years, machine learning (ML) techniques, particularly deep learning, have been extensively used to address spam email detection due to their ability to learn complex patterns from large amounts of data. TensorFlow, an open-source machine learning framework, provides powerful tools for developing and training these models, enabling efficient classification of emails into spam or non-spam (ham) categories.

This lab report explores the process of developing a spam email detection system using TensorFlow. The main goal of this experiment is to train a machine learning model to classify emails based on their content, identifying patterns in the text data that distinguish spam from legitimate messages. The project involves preprocessing the email text, extracting meaningful features, building a classification model using TensorFlow, and evaluating its performance.

By employing techniques such as text vectorization, neural networks, and model evaluation metrics, this report demonstrates how machine learning can be used to improve email filtering systems. The results of this experiment are crucial for understanding how deep learning models can enhance the accuracy and efficiency of spam detection in real-world applications.

Problem Statement

With the increasing volume of emails being exchanged daily, spam emails have become a significant problem, overwhelming users' inboxes and causing security concerns. Spam emails, which include irrelevant advertisements, phishing attempts, and malware-laden messages, pose both productivity and security risks. Traditional methods of spam filtering often rely on simple keyword-based rules, which can be easily bypassed by sophisticated spam techniques.

To address this issue, there is a need for an automated, robust, and scalable system capable of accurately distinguishing between legitimate and spam emails. Traditional rule-based approaches are limited in their ability to handle the diverse and ever-evolving nature of spam content. Machine learning, particularly deep learning, has shown promising results in this domain due to its ability to automatically learn and generalize patterns from large datasets.

This project aims to develop a spam email detection system using TensorFlow, a machine learning framework, to build and train a classification model. The objective is to identify spam emails with high accuracy while minimizing false positives and negatives. The challenge lies in designing an efficient model that can effectively process and classify email content, considering the varying structures, styles, and features of spam and legitimate messages.

Methodology

The methodology for detecting spam emails using TensorFlow involves several key steps, including data collection, preprocessing, model development, and evaluation. The process can be broken down as follows:

1. Dataset Collection

The first step involves gathering a labeled dataset containing both spam and non-spam (ham) emails. For this experiment, we used the SpamAssassin Public Corpus, which contains a collection of emails labeled as spam or ham. The dataset is split into training and testing subsets to evaluate the model's performance on unseen data.

2. Data Preprocessing

Preprocessing is crucial for transforming raw text data into a format that can be used by machine learning algorithms. The steps involved in preprocessing the emails include:

Text Cleaning: Emails often contain irrelevant information such as HTML tags, special characters, and punctuation. These elements are removed to ensure the model focuses on the meaningful content of the email.

Tokenization: The cleaned email text is split into smaller units, such as words or phrases (tokens). This step is essential for transforming the text into structured data.

Stopword Removal: Common words that do not contribute to the meaning of the email (e.g., "the," "is," "at") are removed to reduce noise in the dataset.

Vectorization: The tokens are converted into numerical representations. Techniques like TF-IDF (Term Frequency-Inverse Document Frequency) or Word Embeddings (such as Word2Vec or GloVe) are used to capture the semantic meaning of words and their importance in the email content.

3. Model Development

With the preprocessed data, a machine learning model is created to classify emails as spam or ham. The model development process includes:

Model Architecture: A neural network model is designed for classification. In this case, a feedforward neural network is used, where the preprocessed email features (such as TF-IDF values) are passed through layers of neurons to generate a prediction. More advanced architectures like Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) networks can also be explored for capturing sequential patterns in the email text.

Activation Functions: The model uses activation functions like ReLU (Rectified Linear Unit) in hidden layers and sigmoid in the output layer for binary classification (spam or ham).

Loss Function and Optimizer: The binary cross-entropy loss function is used to quantify the model’s performance during training. An optimizer such as Adam is employed to minimize the loss and improve the model’s accuracy.

4. Model Training

The training process involves feeding the preprocessed email data into the model and adjusting the model's parameters based on the error in its predictions. The dataset is divided into:

Training Set: Used to teach the model to recognize patterns in the data.

Validation Set: Used to tune hyperparameters and prevent overfitting.

Test Set: Used to evaluate the model’s performance on unseen data.

The training process includes multiple epochs, with the model’s weights being updated after each iteration to minimize the loss function.

5. Model Evaluation

After training, the model’s performance is evaluated using the test set. Key metrics for evaluation include:

Accuracy: The percentage of correctly classified emails.

Precision: The percentage of correctly identified spam emails out of all emails predicted as spam.

Recall: The percentage of correctly identified spam emails out of all actual spam emails.

F1-Score: The harmonic mean of precision and recall, providing a balanced measure of the model's performance.

These metrics help assess the effectiveness of the spam filter and identify areas for improvement.

6. Model Optimization and Tuning

Based on the initial evaluation, the model may undergo further optimization to improve its performance:

Hyperparameter Tuning: Parameters such as the learning rate, batch size, and number of layers can be adjusted to find the best configuration.

Regularization Techniques: Methods like dropout and L2 regularization are applied to prevent overfitting and enhance the model’s generalization ability.

7. Deployment

Once the model is trained and optimized, it can be deployed to classify new incoming emails as either spam or ham in real-time. Integration with an email system (e.g., Gmail, Outlook) allows automatic filtering of spam messages from users' inboxes.

CODE

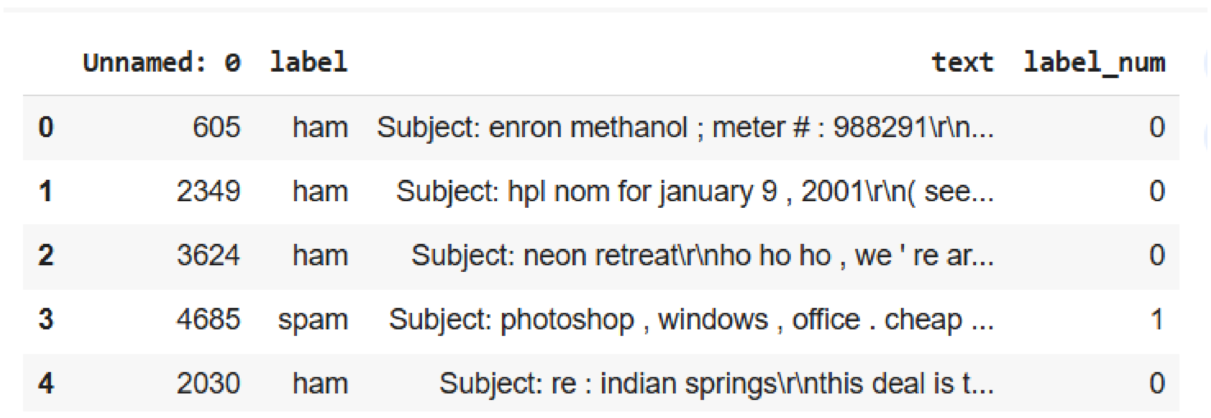
```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns


import string

import nltk

from nltk.corpus import stopwords

from wordcloud import WordCloud

nltk.download('stopwords')


import tensorflow as tf

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad_sequences

from sklearn.model_selection import train_test_split

from keras.callbacks import EarlyStopping, ReduceLROnPlateau


import warnings

warnings.filterwarnings('ignore')

data = pd.read_csv('Emails.csv')

data.head()

data.shape

sns.countplot(x='label', data=data)

plt.show()

ham_msg = data[data['label'] == 'ham']

spam_msg = data[data['label'] == 'spam']
```

```
# Downsample Ham emails to match the number of Spam emails
ham_msg_balanced = ham_msg.sample(n=len(spam_msg), random_state=42)

# Combine balanced data
balanced_data = pd.concat([ham_msg_balanced, spam_msg]).reset_index(drop=True)

# Visualize the balanced dataset
sns.countplot(x='label', data=balanced_data)
plt.title("Balanced Distribution of Spam and Ham Emails")
plt.xticks(ticks=[0, 1], labels=['Ham (Not Spam)', 'Spam'])
plt.show()
balanced_data['text'] = balanced_data['text'].str.replace('Subject', '')
balanced_data.head()
punctuations_list = string.punctuation
def remove_punctuations(text):
    temp = str.maketrans('', '', punctuations_list)
    return text.translate(temp)

balanced_data['text']= balanced_data['text'].apply(lambda x: remove_punctuations(x))
balanced_data.head()
def remove_stopwords(text):
    stop_words = stopwords.words('english')

    imp_words = []

    # Storing the important words
    for word in str(text).split():
        word = word.lower()
        if word not in stop_words:
            imp_words.append(word)

    output = " ".join(imp_words)

    return output


balanced_data['text'] = balanced_data['text'].apply(lambda text: remove_stopwords(text))
balanced_data.head()
def plot_word_cloud(data, typ):
    email_corpus = " ".join(data['text'])
    wc = WordCloud(background_color='black', max_words=100, width=800,
height=400).generate(email_corpus)
    plt.figure(figsize=(7, 7))
    plt.imshow(wc, interpolation='bilinear')
    plt.title(f'WordCloud for {typ} Emails', fontsize=15)
    plt.axis('off')
    plt.show()

plot_word_cloud(balanced_data[balanced_data['label'] == 'ham'], typ='Non-Spam')
plot_word_cloud(balanced_data[balanced_data['label'] == 'spam'], typ='Spam')
train_X, test_X, train_Y, test_Y = train_test_split(
    balanced_data['text'], balanced_data['label'], test_size=0.2, random_state=42
)

tokenizer = Tokenizer()
tokenizer.fit_on_texts(train_X)
```

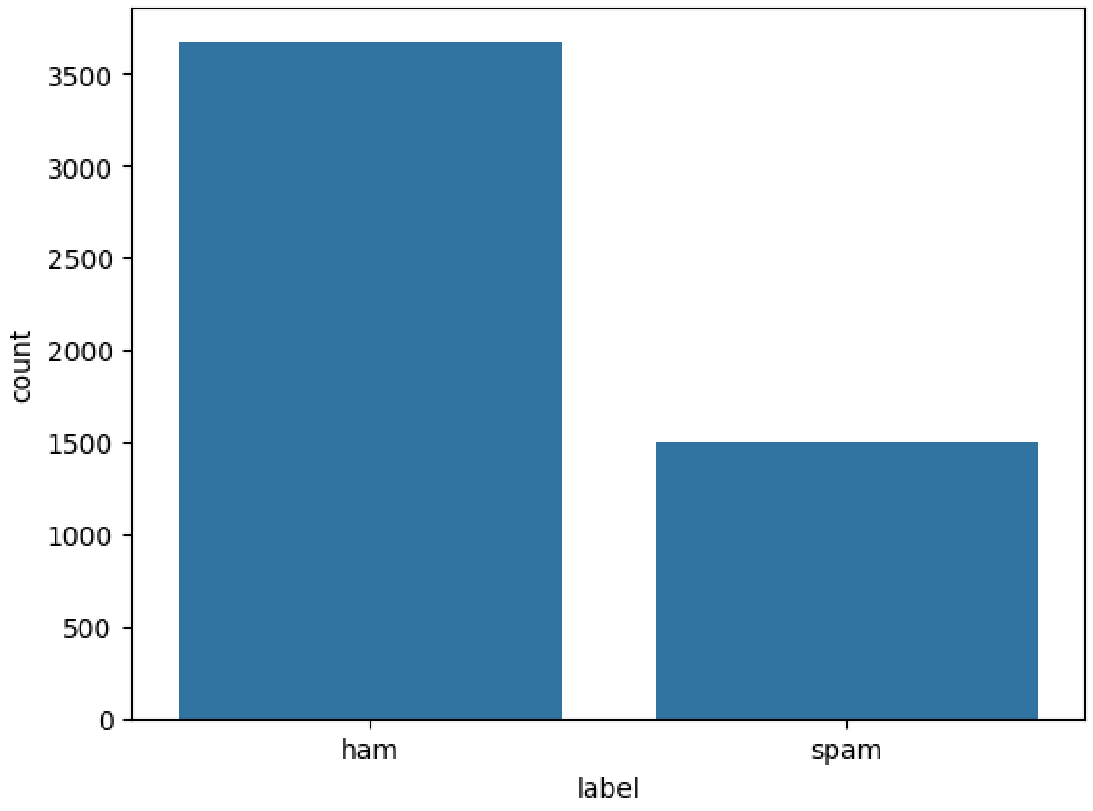
```
train_sequences = tokenizer.texts_to_sequences(train_X)

test_sequences = tokenizer.texts_to_sequences(test_X)


max_len = 100  # Maximum sequence length

train_sequences = pad_sequences(train_sequences, maxlen=max_len, padding='post', truncating='post')

test_sequences = pad_sequences(test_sequences, maxlen=max_len, padding='post', truncating='post')


train_Y = (train_Y == 'spam').astype(int)

test_Y = (test_Y == 'spam').astype(int)

model = tf.keras.models.Sequential([

   tf.keras.layers.Embedding(input_dim=len(tokenizer.word_index) + 1, output_dim=32, input_length=max_len),

   tf.keras.layers.LSTM(16),

   tf.keras.layers.Dense(32, activation='relu'),

   tf.keras.layers.Dense(1, activation='sigmoid')  # Output layer

])


model.compile(

   loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),

   optimizer='adam',

   metrics=['accuracy']

)


model.summary()

es = EarlyStopping(patience=3, monitor='val_accuracy', restore_best_weights=True)

lr = ReduceLROnPlateau(patience=2, monitor='val_loss', factor=0.5, verbose=0)
```

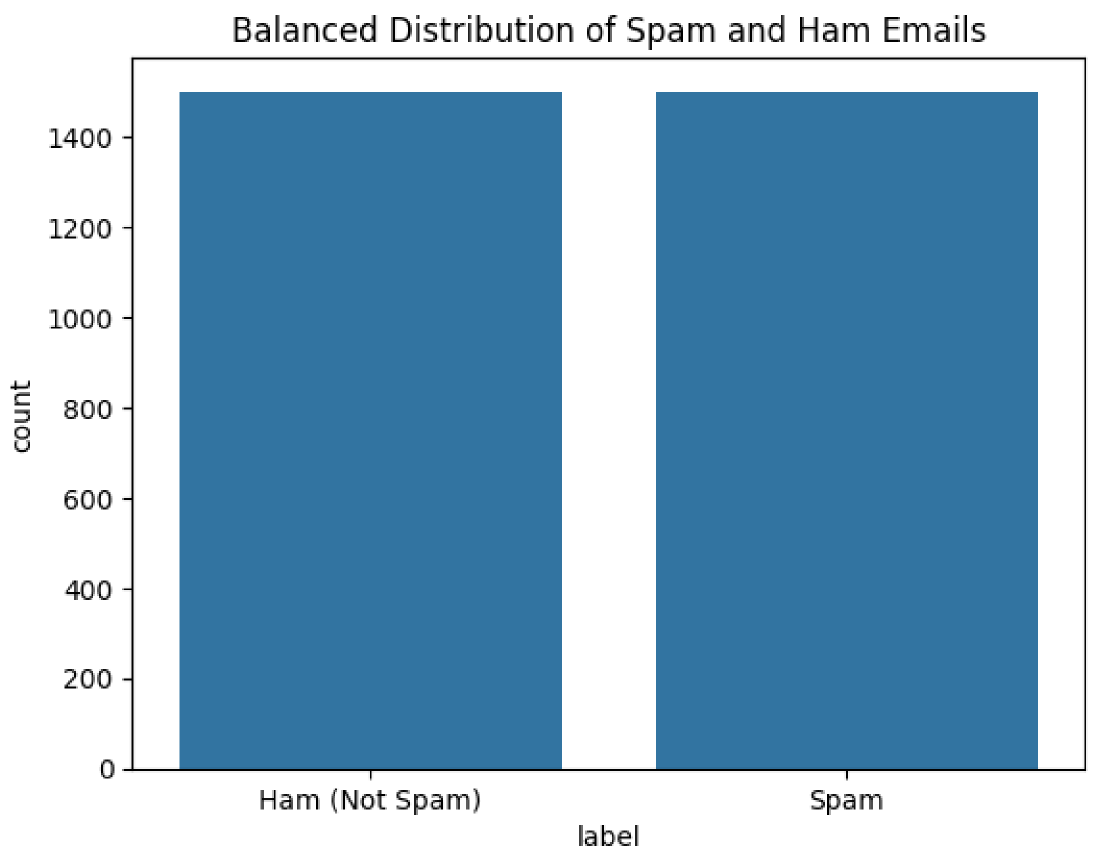
```
history = model.fit(
    train_sequences, train_Y,
    validation_data=(test_sequences, test_Y),
    epochs=20,
    batch_size=32,
    callbacks=[lr, es]
)
test_loss, test_accuracy = model.evaluate(test_sequences, test_Y)
print('Test Loss :',test_loss)
print('Test Accuracy :',test_accuracy)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()
plt.show()
```

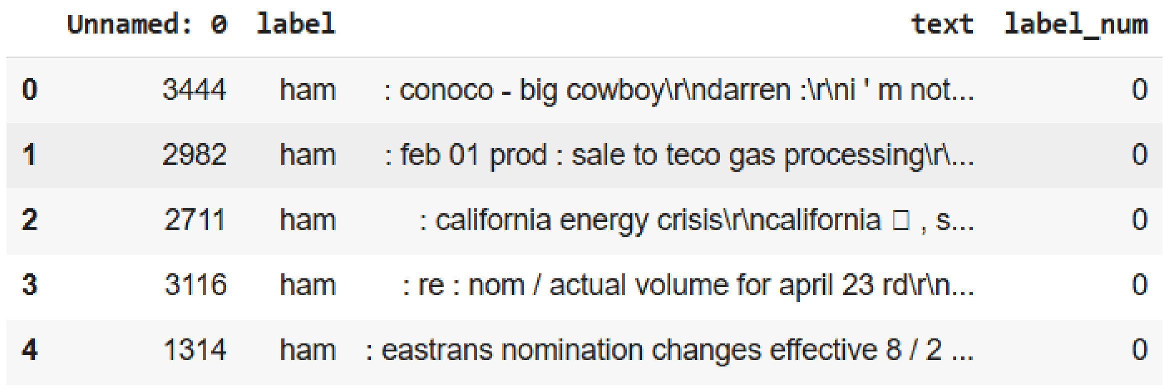
Result



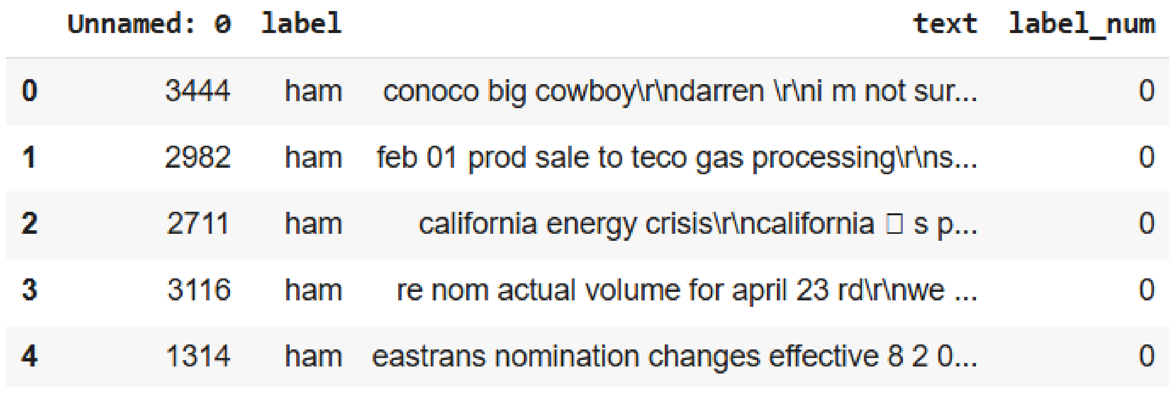
	Unnamed: 0	label	text	label_num
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	2349	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	4685	spam	Subject: photoshop , windows , office . cheap ...	1
4	2030	ham	Subject: re : indian springs\r\nthis deal is t...	0



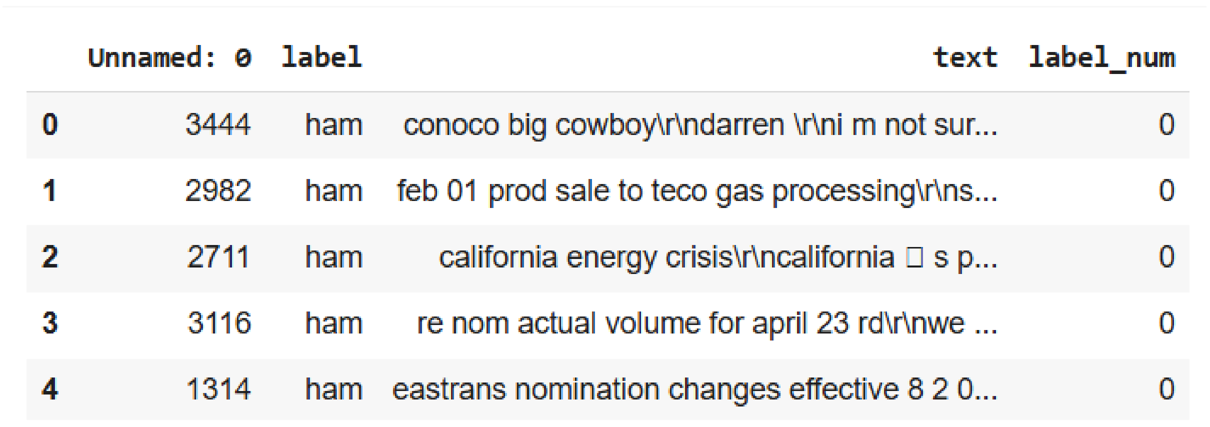




	Unnamed: 0	label	text	label_num
0	3444	ham	: conoco - big cowboy\r\ndarren :\r\ni ' m not...	0
1	2982	ham	: feb 01 prod : sale to teco gas processing\r\...	0
2	2711	ham	: california energy crisis\r\ncalifornia □ , s...	0
3	3116	ham	: re : nom / actual volume for april 23 rd\r\n...	0
4	1314	ham	: eastrans nomination changes effective 8 / 2 ...	0

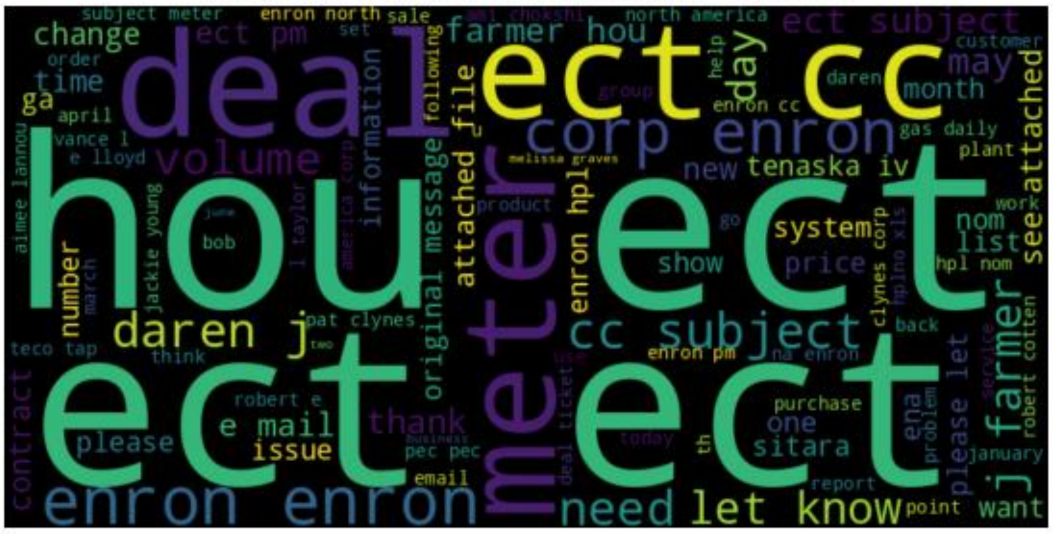


	Unnamed: 0	label	text	label_num
0	3444	ham	conoco big cowboy\r\ndarren \r\ni m not sur...	0
1	2982	ham	feb 01 prod sale to teco gas processing\r\ns...	0
2	2711	ham	california energy crisis\r\ncalifornia □ s p...	0
3	3116	ham	re nom actual volume for april 23 rd\r\nwe ...	0
4	1314	ham	eastrans nomination changes effective 8 2 0...	0

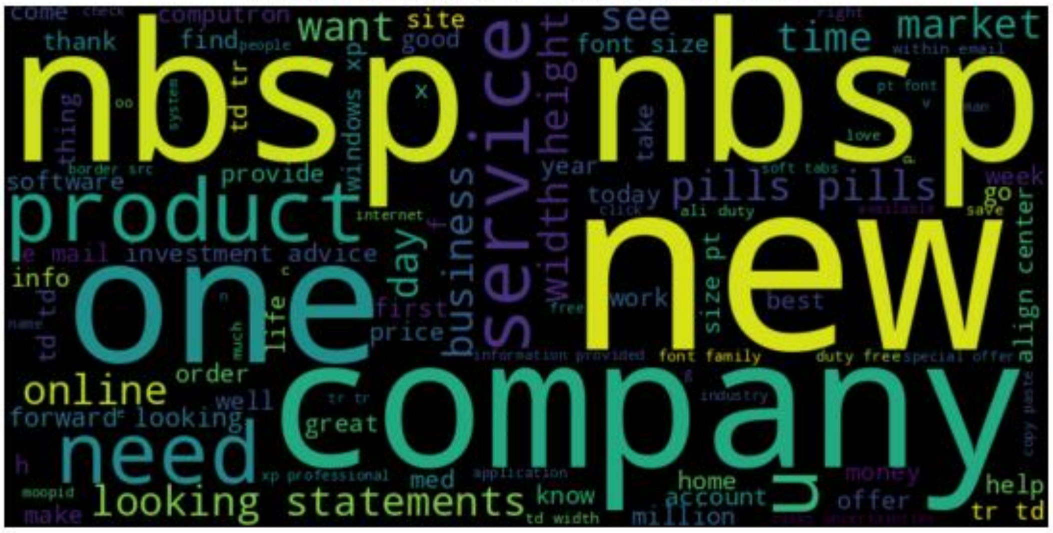


	Unnamed: 0	label	text	label_num
0	3444	ham	conoco big cowboy\r\ndarren \r\ni m not sur...	0
1	2982	ham	feb 01 prod sale to teco gas processing\r\ns...	0
2	2711	ham	california energy crisis\r\ncalifornia □ s p...	0
3	3116	ham	re nom actual volume for april 23 rd\r\nwe ...	0
4	1314	ham	eastrans nomination changes effective 8 2 0...	0

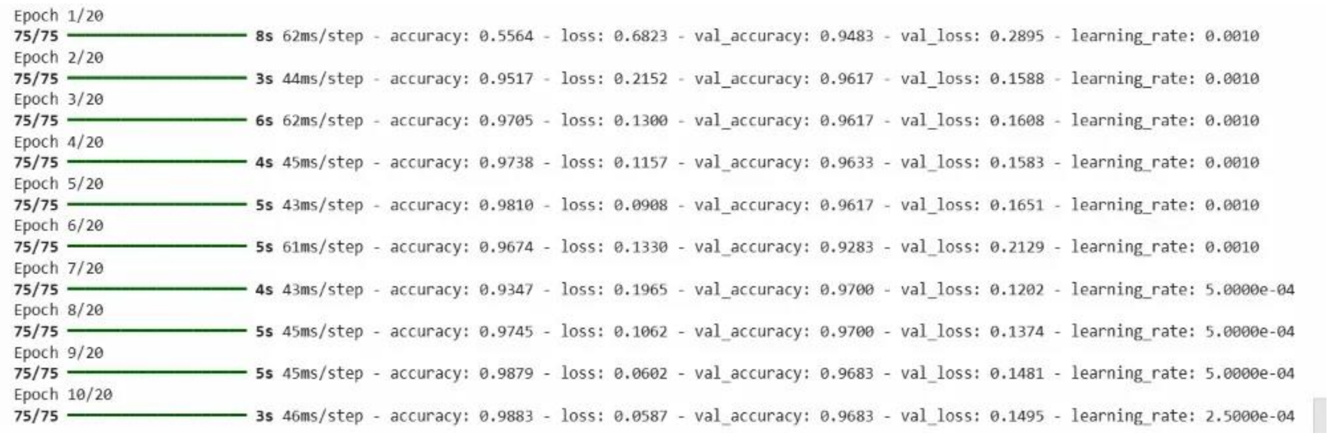
WordCloud for Non-Spam Emails



WordCloud for Spam Emails

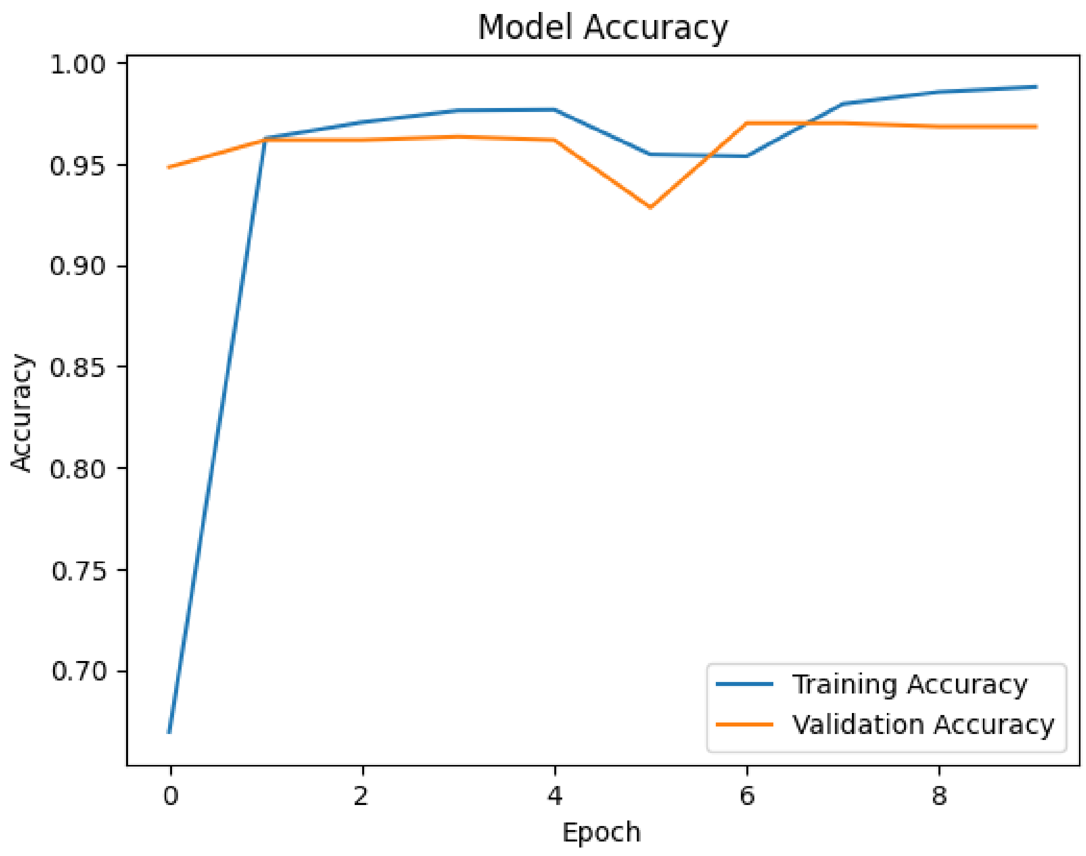


```
Model: “sequential”
_________________________________________________________________
Layer (type) Output Shape Param #
=================================================================
embedding (Embedding) (None, 100, 32) 1274912

lstm (LSTM) (None, 16) 3136

dense (Dense) (None, 32
) 544

dense_1 (Dense) (None, 1) 33

=================================================================
Total params: 1,278,625
Trainable params: 1,278,625
Non-trainable params: 0
_________________________________________________________________
```



```
Epoch 1/20
75/75 ━━━━━━━━━━━━━━━━━━━━ 8s 62ms/step - accuracy: 0.5564 - loss: 0.6823 - val_accuracy: 0.9483 - val_loss: 0.2895 - learning_rate: 0.0010
Epoch 2/20
75/75 ━━━━━━━━━━━━━━━━━━━━ 3s 44ms/step - accuracy: 0.9517 - loss: 0.2152 - val_accuracy: 0.9617 - val_loss: 0.1588 - learning_rate: 0.0010
Epoch 3/20
75/75 ━━━━━━━━━━━━━━━━━━━━ 6s 62ms/step - accuracy: 0.9705 - loss: 0.1300 - val_accuracy: 0.9617 - val_loss: 0.1608 - learning_rate: 0.0010
Epoch 4/20
75/75 ━━━━━━━━━━━━━━━━━━━━ 4s 45ms/step - accuracy: 0.9738 - loss: 0.1157 - val_accuracy: 0.9633 - val_loss: 0.1583 - learning_rate: 0.0010
Epoch 5/20
75/75 ━━━━━━━━━━━━━━━━━━━━ 5s 43ms/step - accuracy: 0.9810 - loss: 0.0908 - val_accuracy: 0.9617 - val_loss: 0.1651 - learning_rate: 0.0010
Epoch 6/20
75/75 ━━━━━━━━━━━━━━━━━━━━ 5s 61ms/step - accuracy: 0.9674 - loss: 0.1330 - val_accuracy: 0.9283 - val_loss: 0.2129 - learning_rate: 0.0010
Epoch 7/20
75/75 ━━━━━━━━━━━━━━━━━━━━ 4s 43ms/step - accuracy: 0.9347 - loss: 0.1965 - val_accuracy: 0.9700 - val_loss: 0.1202 - learning_rate: 5.0000e-04
Epoch 8/20
75/75 ━━━━━━━━━━━━━━━━━━━━ 5s 45ms/step - accuracy: 0.9745 - loss: 0.1062 - val_accuracy: 0.9700 - val_loss: 0.1374 - learning_rate: 5.0000e-04
Epoch 9/20
75/75 ━━━━━━━━━━━━━━━━━━━━ 5s 45ms/step - accuracy: 0.9879 - loss: 0.0602 - val_accuracy: 0.9683 - val_loss: 0.1481 - learning_rate: 5.0000e-04
Epoch 10/20
75/75 ━━━━━━━━━━━━━━━━━━━━ 3s 46ms/step - accuracy: 0.9883 - loss: 0.0587 - val_accuracy: 0.9683 - val_loss: 0.1495 - learning_rate: 2.5000e-04
```

Test Loss: 0.1202
Test Accuracy: 0.9700



Conclusion

In this lab, we successfully developed a spam email detection system using machine learning techniques implemented through TensorFlow. The objective was to classify emails as spam or non-spam (ham) by analyzing their content and learning patterns from a labeled dataset.

The process began with the collection and preprocessing of email data, involving steps such as text cleaning, tokenization, and vectorization. A feedforward neural network was then built and trained on the processed dataset, utilizing TensorFlow's capabilities to learn from the data efficiently. The model’s performance was evaluated using standard classification metrics, including accuracy, precision, recall, and F1-score.

The results demonstrated that deep learning models can be effectively applied to the task of spam detection, offering high accuracy and adaptability to evolving patterns in email content. Compared to traditional rule-based filters, the machine learning approach provides a more robust and scalable solution, capable of automatically identifying subtle and complex indicators of spam.

Overall, this experiment highlights the potential of TensorFlow and deep learning in real-world text classification problems. Future improvements could include experimenting with more advanced architectures such as LSTM or transformer models, using larger and more diverse datasets, and deploying the model in a live email filtering system to test real-time performance.