Spam Email Detection

Riya Roy
Gabriel Reganit
Neil Tejnani

Contents

Introduction

- A spam filter is a program that detects unsolicited and unwanted emails, and prevents those emails from getting to a user’s inbox.
- Spam emails can include advertisements, offers, and scams that try to take advantage of users. The emails can also sometimes be virus infected.

Data Sourcing

Kaggle dataset: **Preprocessed TREC 2007 Public Corpus Dataset**

- TREC 2007 Public Corpus Dataset is an email spam detection email. It contains 50199 spam emails and 25220 ham (not spam) emails for a total of 75,419 records.

Data Preprocessing & Cleaning

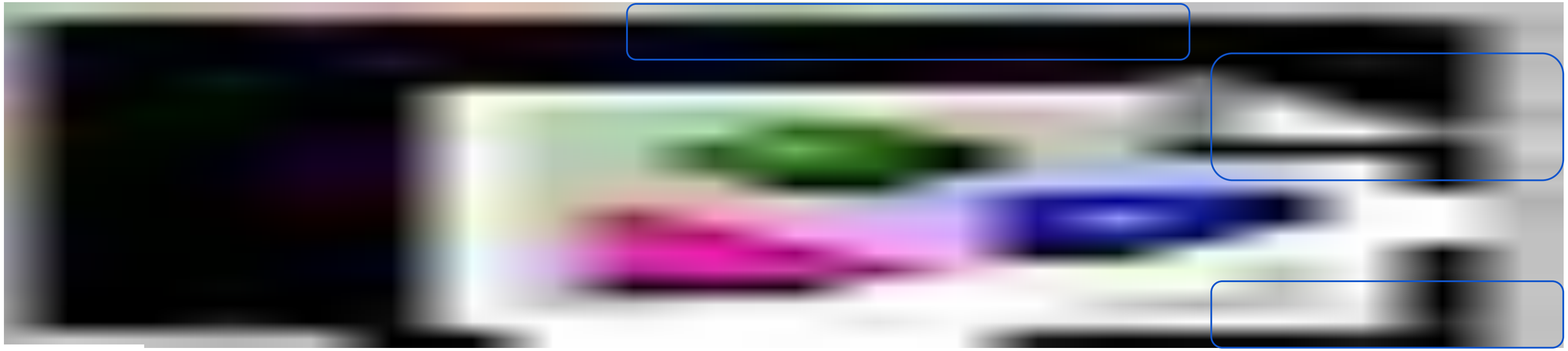
Data Preprocessing

```
(75419, 5)
```

1 = spam emails

0 = ham emails

Data Preprocessing



```
(75419, 5)
```

Dropped data:

1) 2280 rows with null values
2) 10,847 duplicate values, not including null values
3) 33,728 rows of message columns containing “Content-Type:”
4) Contains “email_to” and “email_from” columns

We decided to **focus on the subject and message**, and dropped both email address columns.

```
df_all.message.str.contains('Content-Type|Content-type').value_counts()
```

```
False    40204
True     33728
Name: message, dtype: int64
```

Preprocessed Data

Data Cleaning

Cleaning done on both “message” and “subject” columns:

1) Lowercase all the text
2) Replace all “\n” with “ “
3) Combining → Creating a new content column containing both the subject and message

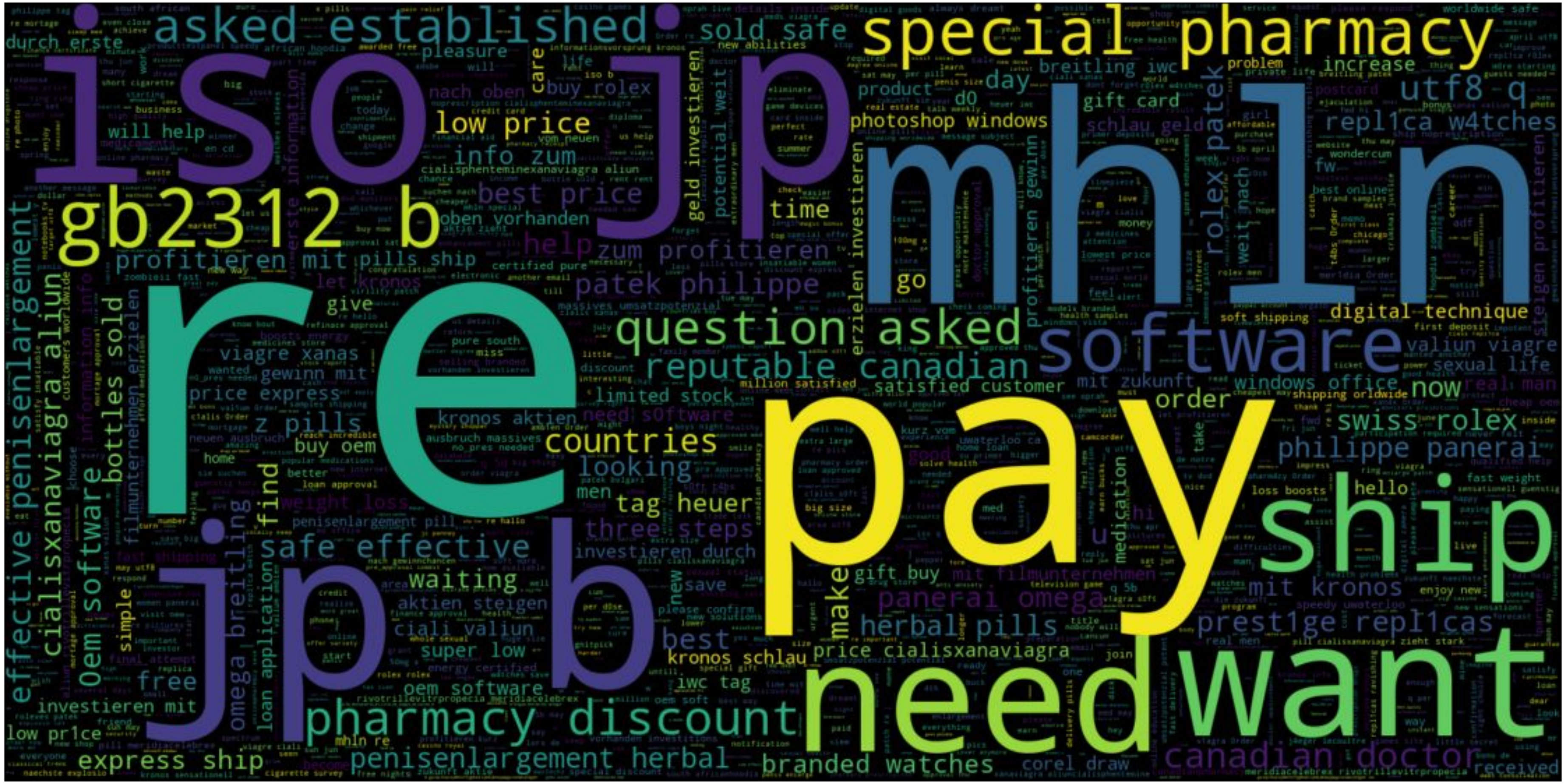
Cleaned Data

Exploring the Data

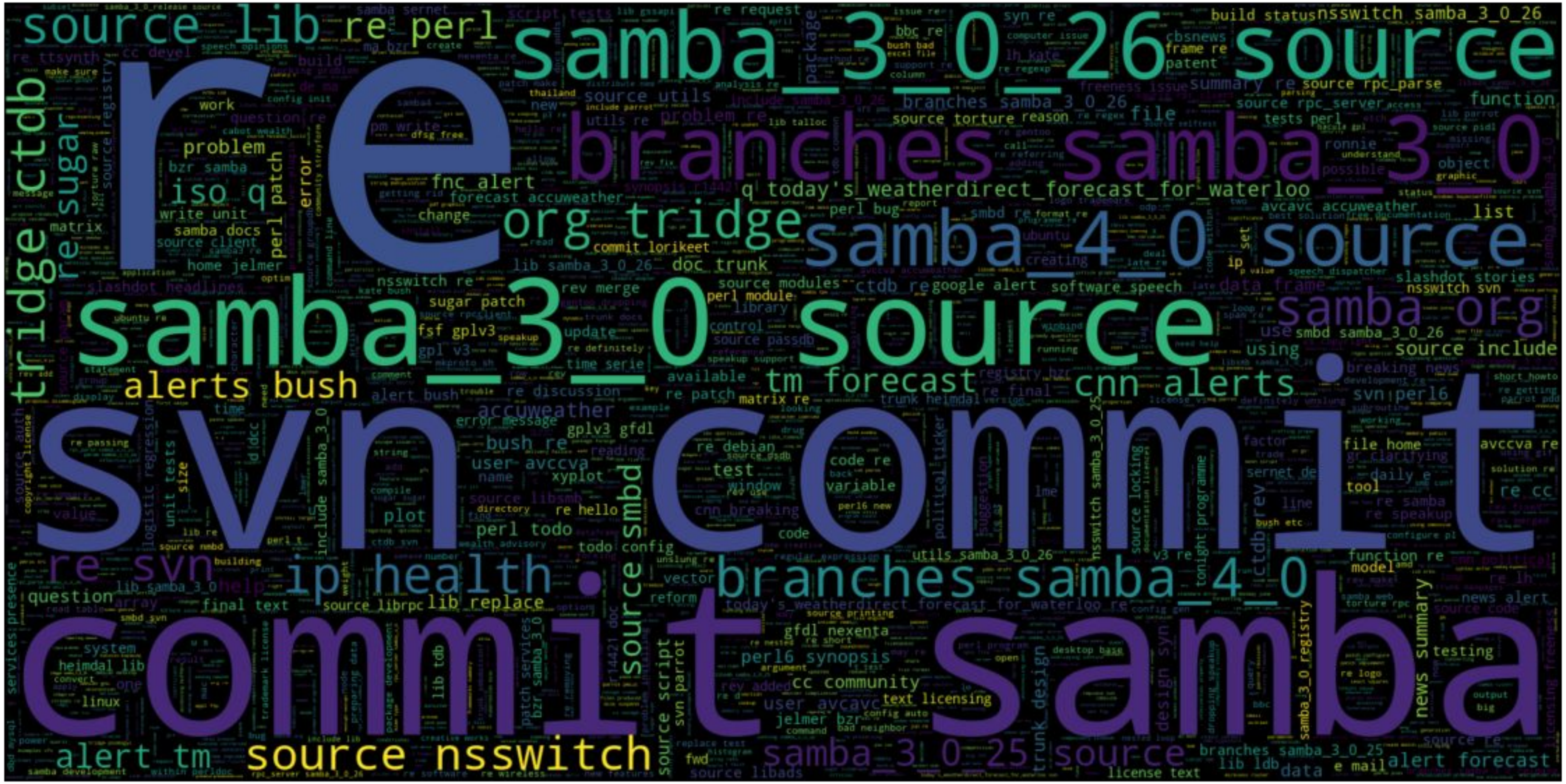
Spam Emails: Messages

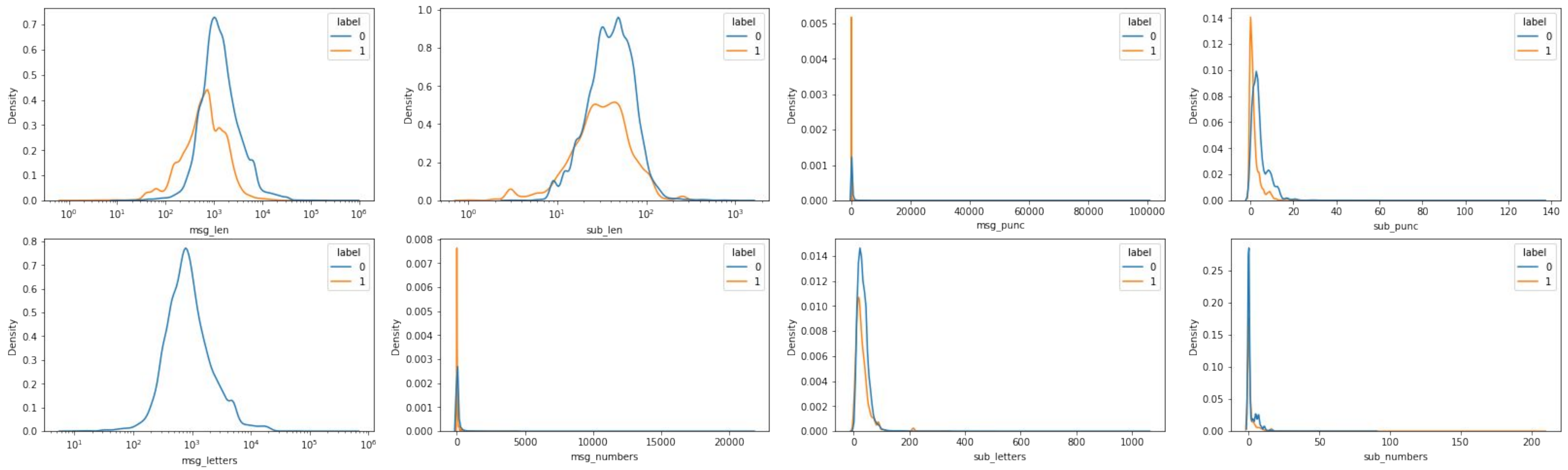
Non-Spam Emails: Messages

Spam Emails: Subjects



Non-Spam Emails: Subjects





Findings:

- ➔ There is significant overlap in the distributions between the numerical features of the messages and subjects, and would likely make differentiating them without using their textual contexts very difficult.
- ➔ There is a very wide distribution in message and subject length, punctuation counts, and the number of ASCII letters and numbers across both classes of emails. The vast majority of message lengths are concentrated are around 1000 in character length; whilst the majority of email subject line are between 100 to 1000 in characters in length.
- ➔ Punctuation and numeric characters have a very wide range, but are concentrated on the lower end for both classes (for both message and subject).

- ➔ Some rows contained significant message lengths, yet had no ASCII characters in them. These would be problematic when doing further textual processing such as stemming, and they were removed.

Data: Before and After Cleaning

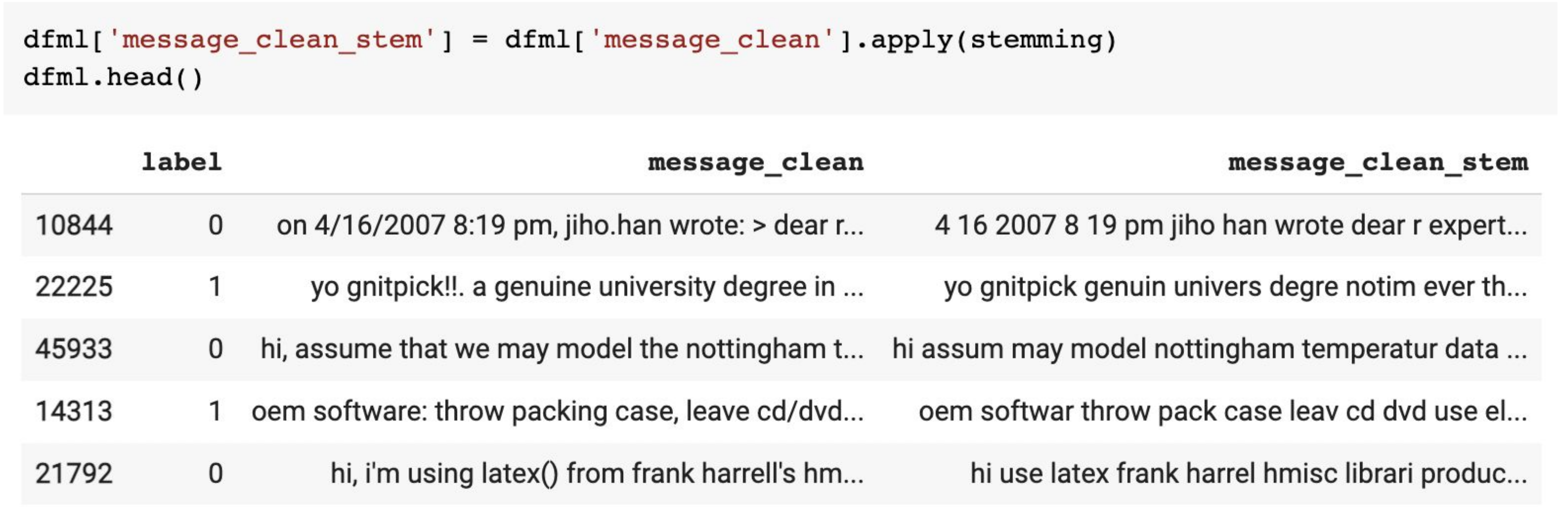
```
0     60.201263
1     39.798737
Name: label, dtype: float64
```

Preprocessing Text Data for ML Models

```
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
```

- ➔ As the dataset is quite large, we chose to use a stratified sample of 15% our data.
- ➔ Using regex to remove non-alphanumeric characters, then stemming and getting rid of stopwords, before joining the values back into strings from lists.

Preprocessing Text Data for ML Models



```
dfml['message_clean_stem'] = dfml['message_clean'].apply(stemming)
dfml.head()
```

	label	message_clean	message_clean_stem
10844	0	on 4/16/2007 8:19 pm, jiho.han wrote: > dear r...	4 16 2007 8 19 pm jiho han wrote dear r expert...
22225	1	yo gnitpick!!. a genuine university degree in ...	yo gnitpick genuin univers degre notim ever th...
45933	0	hi, assume that we may model the nottingham t...	hi assum may model nottingham temperatur data ...
14313	1	oem software: throw packing case, leave cd/dvd...	oem softwar throw pack case leav cd dvd use el...
21792	0	hi, i'm using latex() from frank harrell's hm...	hi use latex frank harrel hmisc librari produc...

Vectorization

- **Tf-idf (Term Frequency–Inverse Document Frequency) Vectorizer**
  - a numerical statistic that is intended to reflect **how important** a word is to a document in a collection or corpus.
  - It calculates the term frequency as well as calculates the inverse document frequency. It then assigns different levels of importance to difference words.

Tf-idf Matrix

```
(0, 44471)    0.04969181837359509
(0, 23349)    0.07488079099705662
(0, 9865)     0.0738888598952085
(0, 19266)    0.09189293099242675
(0, 33724)    0.0576302855845526
(0, 35683)    0.12558118296575002
(0, 14205)    0.044848487261753826
```

Machine Learning: Models

- **Classification Models**
    - Logistic Regression
    - Support Vector Machines
    - Naive Bayes → Used by Gmail in the early days of email for spam filtering, good for problems with multiple classes

Logistic Regression:

- Accuracy: 98.92%
- Precision: 99%
- Recall: 99%

➔ 7 ham emails predicted as spam (False positive)

➔ 4 spam emails predicted as ham (False negative)

Support Vector Machines:

- Accuracy: 99.31%
- Precision: 99%
- Recall: 99%

➔ 4 ham emails predicted as spam
➔ 3 spam emails predicted as ham

Naive Bayes:

- Accuracy: 96.65%
- Precision: 97%
- Recall: 97%

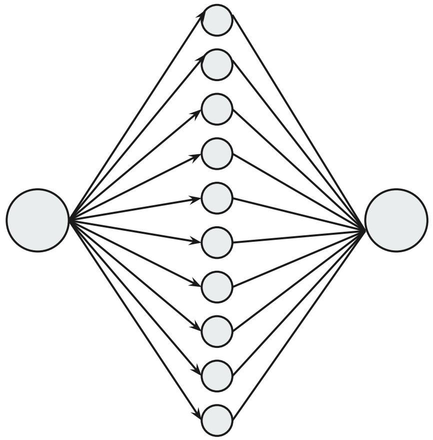
➔ 2 ham emails predicted as spam
➔ 32 spam emails predicted as ham
➔ Lowest accuracy of all ML models

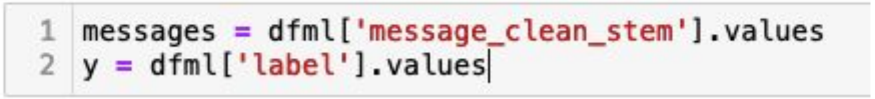
Conclusions on ML Model Performance

- ➔ Highest Accuracy ⟶ SVM
- ➔ Least False Positives ⟶ Naive Bayes
- ➔ Selecting only one model unreasonable

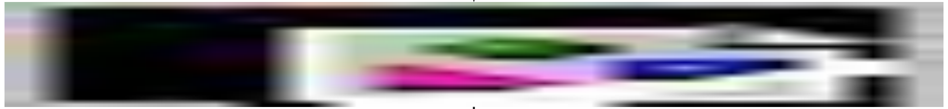
Deep Learning: Artificial Neural Network

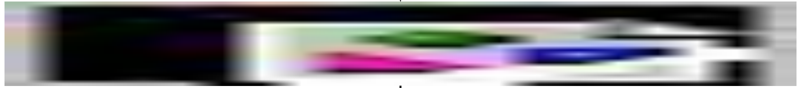
- In addition to the ML models created, a basic neural network was created for the purposes of classification. It consists of the input layer, and one dense layer with 10 neurons, followed by an output layer. Trained for 20 epochs.





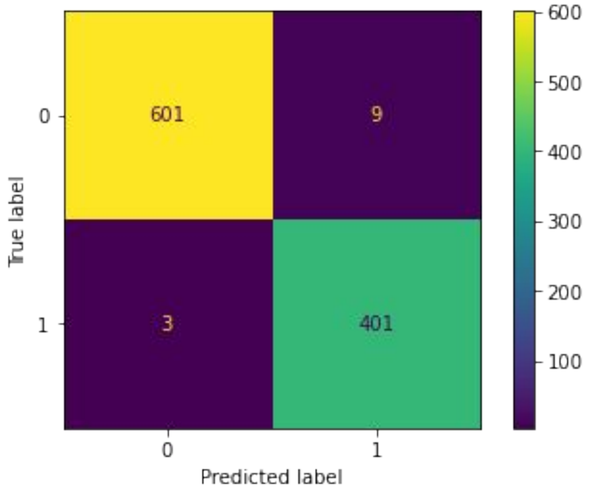
```python
messages = dfml['message_clean_stem'].values
y = dfml['label'].values
```





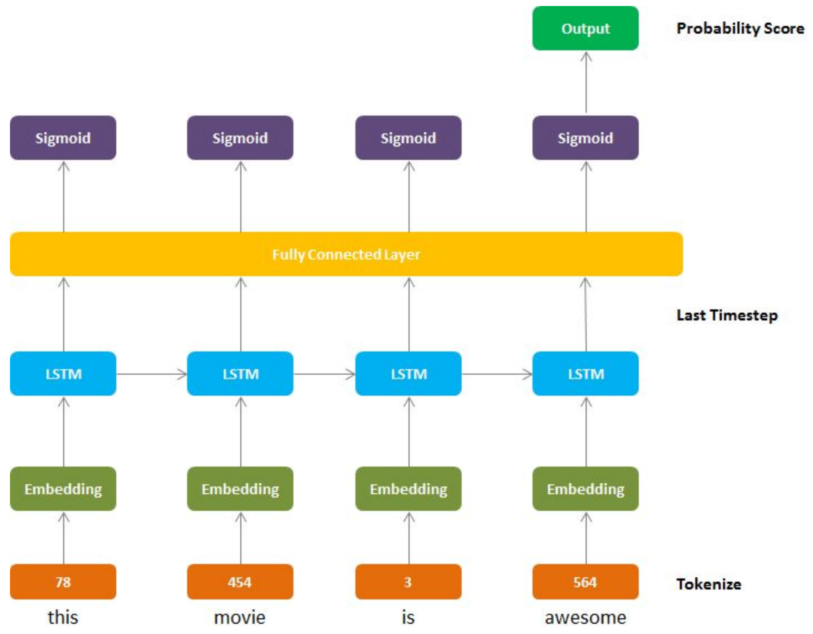
ANN Performance

- Training accuracy: 99.90%
- Validation accuracy: 98.82%
- Recall: 99%
- Precision: 99%



Long Short Term Memory

- Variant of RNN for contextualizing words
- Bidirectional parameter
    - Check the context early and later down the text
    - Choose the best ones
    - Merge them



Data Preparation for LSTM: Text Preprocessing

- Stopwords and any links (http/https) were removed in the text preprocessing steps.
- This was done to make the overall sequence length shorter and exclude non-real words (such as links/emails embedded into emails).

Finding Max Sequence Length

- The approximate word count for the stratified sample dataframe was found by finding the number of spaces inside each message. The mean + 1 std worth of spaces encompassed almost 93% of all emails (525 spaces). This was used as the maximum sequence length for the LSTM.

Step 1 Use a tokenizer and fit it on the training data messages.

Step 2 Pad the sequences so they are all the same length, up to a maximum of 525 words.

Step 3 Use Label Encoder to encode the y variables.

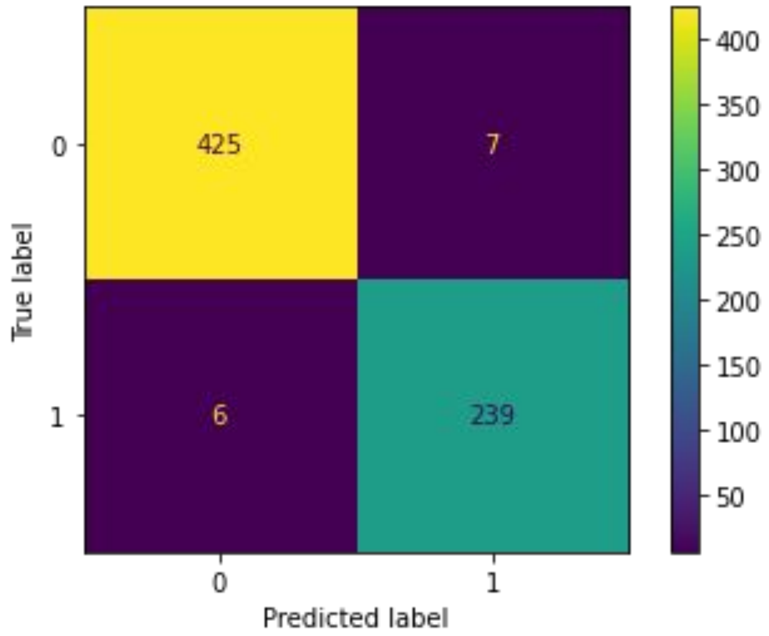
Step 4 Create an embedding index and matrix, use it to create an embedding layer.

Step 5 Create the LSTM model, train and test.

Long Short Term Memory:

- Accuracy: 98.07%
- Precision: 99%
- Recall: 98%

➔ 7 ham emails predicted as spam
➔ 6 spam emails predicted as ham



Challenges

- Configuring tensorflow on Jupyter notebook to use personal NVIDIA GPU in Windows
  - Kernel dies after numerous attempts
  - Minimize costs than buying Google Colab Pro
- Removing text with html code in order to not include them as part of the email’s content

Future Steps

- Testing with real emails
- Test without html code
- Test for overfitting

12/09/2022 meeting

1. Finalising the script
    a. Preprocessing and cleaning
    b. Train and test
    c. ML
    d. Small ANN
    e. LSTM attempt (note saying CPU/GPU not enough)
2. Presentation
    a. Word clouds
    b. EDA
3. Testing with real emails
    a. FINISH conf matrix send to gabe for LSTM
    b. ML + ANN and see how they perform
    c.

Neil: preprocessing and ML

Gabe: LSTM performance + real email performance

Riya: ANN + building the LSTM