```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models, preprocessing
```

```
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/ashwin-som/cs4372/main/SPAM%20text%20message%2020170820%20-%20Data.csv')
```

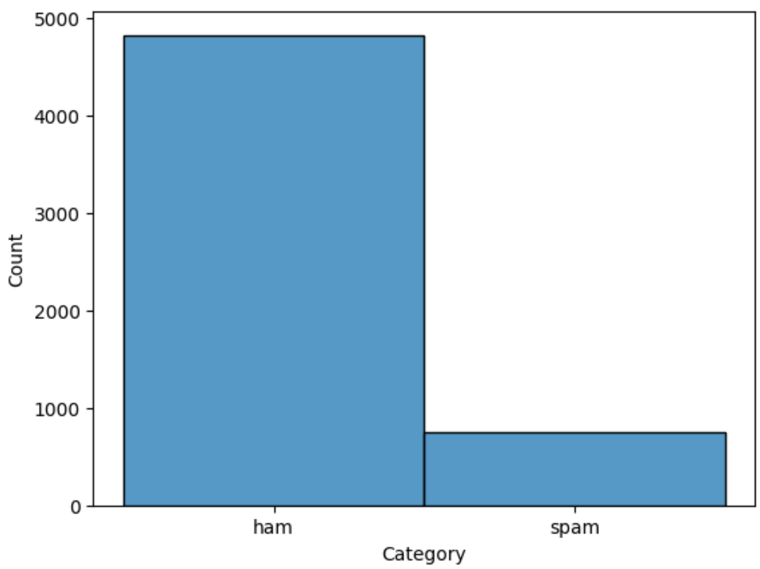
Dataset Description:

This is a dataset containing 5572 instances of data where each instance consists of a message and a category specifying whether that corresponding message is spam or not.

Model prediction goals:

The model should be able to differentiate between messages that are spam vs non-spam. The model will look at the word embeddings and pass them through different sequential layers. Some exmaples of this shown below are CNN, RNN, LSTM, etc. For each type of model created, I have included some sort of embedding to enchance the performance of the model.

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.histplot(data=df,x='Category')
plt.show()
```




```
df = df.replace('ham',0)
df = df.replace('spam',1)
df
```

	Category	Message
1	0	Ok lar... Joking wif u oni...
...	...	...
5568	0	Will u b going to esplanade fr home?
5571	0	Rofl. Its true to its name

```
from sklearn.model_selection import train_test_split
```

```
x = df.Message
y = df.Category
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, train_size=0.8, random_state=1234)
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np
```

```
vectorizer = TfidfVectorizer()
x_train = vectorizer.fit_transform(x_train)
x_test = vectorizer.transform(x_test)
```

```
x_train.shape
```

```
(4457, 7667)
```

```
x_train = x_train.toarray()
x_test = x_test.toarray()
```

```
seq_model = models.Sequential()
seq_model.add(layers.Dense(32, activation='relu', input_shape=(7667,)))
seq_model.add(layers.Dense(32, activation='relu'))
seq_model.add(layers.Dense(1, activation='sigmoid'))
```

Sequential Model

```
seq_model.compile(optimizer='rmsprop',loss='binary_crossentropy',metrics=['accuracy'])
```

```
history = seq_model.fit(x_train,y_train,epochs=20,batch_size=512,validation_data=(x_test, y_test))
```

```
Epoch 1/20
9/9 [==============================] - 6s 48ms/step - loss: 0.6553 - accuracy: 0.8649 - val_loss: 0.6100 - val_accuracy: 0.8
Epoch 2/20
9/9 [==============================] - 0s 18ms/step - loss: 0.5747 - accuracy: 0.8667 - val_loss: 0.5286 - val_accuracy: 0.8
Epoch 3/20
9/9 [==============================] - 0s 17ms/step - loss: 0.4928 - accuracy: 0.8667 - val_loss: 0.4517 - val_accuracy: 0.8
Epoch 4/20
9/9 [==============================] - 0s 16ms/step - loss: 0.4191 - accuracy: 0.8667 - val_loss: 0.3872 - val_accuracy: 0.8
Epoch 5/20
9/9 [==============================] - 0s 22ms/step - loss: 0.3572 - accuracy: 0.8667 - val_loss: 0.3345 - val_accuracy: 0.8
Epoch 6/20
9/9 [==============================] - 0s 18ms/step - loss: 0.3066 - accuracy: 0.8667 - val_loss: 0.2920 - val_accuracy: 0.8
Epoch 7/20
9/9 [==============================] - 0s 18ms/step - loss: 0.2662 - accuracy: 0.8667 - val_loss: 0.2580 - val_accuracy: 0.8
Epoch 8/20
9/9 [==============================] - 0s 20ms/step - loss: 0.2331 - accuracy: 0.8667 - val_loss: 0.2291 - val_accuracy: 0.8
Epoch 9/20
9/9 [==============================] - 0s 22ms/step - loss: 0.2050 - accuracy: 0.8793 - val_loss: 0.2041 - val_accuracy: 0.9
Epoch 10/20
9/9 [==============================] - 0s 17ms/step - loss: 0.1805 - accuracy: 0.9233 - val_loss: 0.1822 - val_accuracy: 0.9
Epoch 11/20
9/9 [==============================] - 0s 22ms/step - loss: 0.1588 - accuracy: 0.9473 - val_loss: 0.1637 - val_accuracy: 0.9
Epoch 12/20
9/9 [==============================] - 0s 17ms/step - loss: 0.1401 - accuracy: 0.9607 - val_loss: 0.1473 - val_accuracy: 0.9
Epoch 13/20
9/9 [==============================] - 0s 23ms/step - loss: 0.1239 - accuracy: 0.9695 - val_loss: 0.1325 - val_accuracy: 0.9
Epoch 14/20
9/9 [==============================] - 0s 25ms/step - loss: 0.1092 - accuracy: 0.9767 - val_loss: 0.1211 - val_accuracy: 0.9
Epoch 15/20
9/9 [==============================] - 0s 18ms/step - loss: 0.0966 - accuracy: 0.9800 - val_loss: 0.1088 - val_accuracy: 0.9
Epoch 16/20
9/9 [==============================] - 0s 19ms/step - loss: 0.0846 - accuracy: 0.9854 - val_loss: 0.0998 - val_accuracy: 0.9
Epoch 17/20
9/9 [==============================] - 0s 17ms/step - loss: 0.0740 - accuracy: 0.9877 - val_loss: 0.0903 - val_accuracy: 0.9
Epoch 18/20
9/9 [==============================] - 0s 17ms/step - loss: 0.0645 - accuracy: 0.9901 - val_loss: 0.0836 - val_accuracy: 0.9
Epoch 19/20
```

```
9/9 [==============================] - 0s 23ms/step - loss: 0.0560 - accuracy: 0.9912 - val_loss: 0.0769 - val_accuracy: 0.97
Epoch 20/20
9/9 [==============================] - 0s 16ms/step - loss: 0.0483 - accuracy: 0.9921 - val_loss: 0.0713 - val_accuracy: 0.97
```

```
from sklearn.metrics import classification_report
pred = seq_model.predict(x_test)
pred = [1.0 if p>= 0.5 else 0.0 for p in pred]   #Utilized from Professor Mazidi's resources
print(classification_report(y_test, pred))
```

```
35/35 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       962
           1       0.96      0.88      0.92       153

    accuracy                           0.98      1115
   macro avg       0.97      0.94      0.95      1115
weighted avg       0.98      0.98      0.98      1115
```

RNN Architecture

```
rnn_model = models.Sequential()
rnn_model.add(layers.Dense(32, activation='relu', input_shape=(7667,)))
rnn_model.add(layers.Embedding(1000, 32))
rnn_model.add(layers.SimpleRNN(32))
rnn_model.add(layers.Dense(1, activation='sigmoid'))
```

```
rnn_model.compile(optimizer='rmsprop',loss='binary_crossentropy',metrics=['accuracy'])
```

```
history_rnn = rnn_model.fit(x_train,y_train,epochs=20,batch_size=512,validation_data=(x_test, y_test))
```

```
Epoch 1/20
WARNING:tensorflow:Gradients do not exist for variables ['dense_3/kernel:0', 'dense_3/bias:0'] when minimizing the loss. If y
WARNING:tensorflow:Gradients do not exist for variables ['dense_3/kernel:0', 'dense_3/bias:0'] when minimizing the loss. If y
WARNING:tensorflow:Gradients do not exist for variables ['dense_3/kernel:0', 'dense_3/bias:0'] when minimizing the loss. If y
WARNING:tensorflow:Gradients do not exist for variables ['dense_3/kernel:0', 'dense_3/bias:0'] when minimizing the loss. If y
9/9 [==============================] - 2s 86ms/step - loss: 0.4414 - accuracy: 0.8667 - val_loss: 0.4001 - val_accuracy: 0.86
Epoch 2/20
9/9 [==============================] - 1s 66ms/step - loss: 0.3929 - accuracy: 0.8667 - val_loss: 0.3999 - val_accuracy: 0.86
Epoch 3/20
9/9 [==============================] - 1s 97ms/step - loss: 0.3929 - accuracy: 0.8667 - val_loss: 0.4005 - val_accuracy: 0.86
Epoch 4/20
9/9 [==============================] - 1s 100ms/step - loss: 0.3935 - accuracy: 0.8667 - val_loss: 0.3999 - val_accuracy: 0.8
Epoch 5/20
9/9 [==============================] - 1s 61ms/step - loss: 0.3927 - accuracy: 0.8667 - val_loss: 0.4021 - val_accuracy: 0.86
Epoch 6/20
9/9 [==============================] - 0s 47ms/step - loss: 0.3932 - accuracy: 0.8667 - val_loss: 0.4005 - val_accuracy: 0.86
Epoch 7/20
9/9 [==============================] - 0s 44ms/step - loss: 0.3928 - accuracy: 0.8667 - val_loss: 0.4025 - val_accuracy: 0.86
Epoch 8/20
9/9 [==============================] - 0s 49ms/step - loss: 0.3938 - accuracy: 0.8667 - val_loss: 0.4006 - val_accuracy: 0.86
Epoch 9/20
9/9 [==============================] - 0s 53ms/step - loss: 0.3937 - accuracy: 0.8667 - val_loss: 0.3999 - val_accuracy: 0.86
Epoch 10/20
9/9 [==============================] - 0s 46ms/step - loss: 0.3918 - accuracy: 0.8667 - val_loss: 0.4042 - val_accuracy: 0.86
Epoch 11/20
9/9 [==============================] - 0s 46ms/step - loss: 0.3948 - accuracy: 0.8667 - val_loss: 0.4004 - val_accuracy: 0.86
Epoch 12/20
9/9 [==============================] - 0s 47ms/step - loss: 0.3934 - accuracy: 0.8667 - val_loss: 0.3999 - val_accuracy: 0.86
Epoch 13/20
9/9 [==============================] - 0s 46ms/step - loss: 0.3929 - accuracy: 0.8667 - val_loss: 0.4006 - val_accuracy: 0.86
Epoch 14/20
9/9 [==============================] - 0s 47ms/step - loss: 0.3928 - accuracy: 0.8667 - val_loss: 0.4005 - val_accuracy: 0.86
Epoch 15/20
9/9 [==============================] - 0s 54ms/step - loss: 0.3931 - accuracy: 0.8667 - val_loss: 0.3999 - val_accuracy: 0.86
Epoch 16/20
9/9 [==============================] - 0s 47ms/step - loss: 0.3932 - accuracy: 0.8667 - val_loss: 0.3999 - val_accuracy: 0.86
Epoch 17/20
9/9 [==============================] - 0s 44ms/step - loss: 0.3927 - accuracy: 0.8667 - val_loss: 0.3999 - val_accuracy: 0.86
Epoch 18/20
9/9 [==============================] - 0s 46ms/step - loss: 0.3928 - accuracy: 0.8667 - val_loss: 0.4002 - val_accuracy: 0.86
Epoch 19/20
9/9 [==============================] - 0s 50ms/step - loss: 0.3930 - accuracy: 0.8667 - val_loss: 0.4023 - val_accuracy: 0.86
Epoch 20/20
9/9 [==============================] - 0s 49ms/step - loss: 0.3932 - accuracy: 0.8667 - val_loss: 0.4000 - val_accuracy: 0.86
```

```
from sklearn.metrics import classification_report
pred = rnn_model.predict(x_test)
pred = [1.0 if p>= 0.5 else 0.0 for p in pred] #Utilized from Professor Mazidi's resources
print(classification_report(y_test, pred))
```

```
35/35 [==============================] - 0s 4ms/step
              precision    recall  f1-score   support

           0       0.86      1.00      0.93       962
           1       0.00      0.00      0.00       153

    accuracy                           0.86      1115
   macro avg       0.43      0.50      0.46      1115
weighted avg       0.74      0.86      0.80      1115

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score
  _warn_prf(average, modifier, msg_start, len(result))
```

LSTM architecture

```
lstm_model = models.Sequential()
#lstm_model.add(layers.Dense(32, activation='relu', input_shape=(7667,)))
lstm_model.add(layers.Embedding(1000, 32))
lstm_model.add(layers.LSTM(32))
lstm_model.add(layers.Dense(1, activation='sigmoid'))
```

```
lstm_model.compile(optimizer='rmsprop',loss='binary_crossentropy',metrics=['accuracy'])
```

```
history_lstm = lstm_model.fit(x_train,y_train,epochs=20,batch_size=512,validation_data=(x_test, y_test))
```

```
Epoch 1/20
9/9 [==============================] - 7s 386ms/step - loss: 0.6323 - accuracy: 0.7792 - val_loss: 0.5254 - val_accuracy: 0.8
Epoch 2/20
9/9 [==============================] - 3s 333ms/step - loss: 0.4253 - accuracy: 0.8667 - val_loss: 0.4003 - val_accuracy: 0.8
Epoch 3/20
9/9 [==============================] - 3s 339ms/step - loss: 0.3932 - accuracy: 0.8667 - val_loss: 0.4014 - val_accuracy: 0.8
Epoch 4/20
9/9 [==============================] - 3s 338ms/step - loss: 0.3939 - accuracy: 0.8667 - val_loss: 0.4003 - val_accuracy: 0.8
Epoch 5/20
9/9 [==============================] - 3s 333ms/step - loss: 0.3933 - accuracy: 0.8667 - val_loss: 0.4000 - val_accuracy: 0.8
Epoch 6/20
9/9 [==============================] - 3s 337ms/step - loss: 0.3931 - accuracy: 0.8667 - val_loss: 0.3999 - val_accuracy: 0.8
Epoch 7/20
9/9 [==============================] - 3s 380ms/step - loss: 0.3935 - accuracy: 0.8667 - val_loss: 0.4001 - val_accuracy: 0.8
Epoch 8/20
9/9 [==============================] - 3s 379ms/step - loss: 0.3941 - accuracy: 0.8667 - val_loss: 0.4004 - val_accuracy: 0.8
Epoch 9/20
9/9 [==============================] - 3s 339ms/step - loss: 0.3927 - accuracy: 0.8667 - val_loss: 0.4004 - val_accuracy: 0.8
Epoch 10/20
9/9 [==============================] - 3s 336ms/step - loss: 0.3936 - accuracy: 0.8667 - val_loss: 0.3999 - val_accuracy: 0.8
Epoch 11/20
9/9 [==============================] - 3s 344ms/step - loss: 0.3936 - accuracy: 0.8667 - val_loss: 0.4009 - val_accuracy: 0.8
Epoch 12/20
9/9 [==============================] - 3s 343ms/step - loss: 0.3931 - accuracy: 0.8667 - val_loss: 0.4029 - val_accuracy: 0.8
Epoch 13/20
9/9 [==============================] - 3s 340ms/step - loss: 0.3932 - accuracy: 0.8667 - val_loss: 0.4038 - val_accuracy: 0.8
Epoch 14/20
9/9 [==============================] - 3s 338ms/step - loss: 0.3934 - accuracy: 0.8667 - val_loss: 0.4013 - val_accuracy: 0.8
Epoch 15/20
9/9 [==============================] - 3s 388ms/step - loss: 0.3930 - accuracy: 0.8667 - val_loss: 0.4007 - val_accuracy: 0.8
Epoch 16/20
9/9 [==============================] - 3s 336ms/step - loss: 0.3929 - accuracy: 0.8667 - val_loss: 0.4003 - val_accuracy: 0.8
Epoch 17/20
9/9 [==============================] - 3s 340ms/step - loss: 0.3926 - accuracy: 0.8667 - val_loss: 0.4003 - val_accuracy: 0.8
Epoch 18/20
9/9 [==============================] - 3s 343ms/step - loss: 0.3938 - accuracy: 0.8667 - val_loss: 0.4010 - val_accuracy: 0.8
Epoch 19/20
9/9 [==============================] - 3s 353ms/step - loss: 0.3930 - accuracy: 0.8667 - val_loss: 0.4024 - val_accuracy: 0.8
Epoch 20/20
9/9 [==============================] - 3s 341ms/step - loss: 0.3935 - accuracy: 0.8667 - val_loss: 0.4042 - val_accuracy: 0.8
```

```
from sklearn.metrics import classification_report
pred = lstm_model.predict(x_test)
pred = [1.0 if p>= 0.5 else 0.0 for p in pred] #Utilized from Professor Mazidi's resources
print(classification_report(y_test, pred))
```

```
35/35 [==============================] - 3s 70ms/step
              precision    recall  f1-score   support

           0       0.86      1.00      0.93       962
           1       0.00      0.00      0.00       153

    accuracy                           0.86      1115
   macro avg       0.43      0.50      0.46      1115
weighted avg       0.74      0.86      0.80      1115

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score
  _warn_prf(average, modifier, msg_start, len(result))
```

CNN

```
cnn_model = models.Sequential()
cnn_model.add(layers.Dense(32, activation='relu', input_shape=(7667,)))
cnn_model.add(layers.Embedding(1000, 32))
cnn_model.add(layers.Conv1D(32, 7, activation='relu'))
cnn_model.add(layers.MaxPooling1D(5))
cnn_model.add(layers.Dense(1, activation='sigmoid'))
```

```
cnn_model.compile(optimizer='rmsprop',loss='binary_crossentropy',metrics=['accuracy'])
```

```
history_cnn = cnn_model.fit(x_train,y_train,epochs=20,batch_size=512,validation_data=(x_test, y_test))
```

```
Epoch 1/20
WARNING:tensorflow:Gradients do not exist for variables ['dense_5/kernel:0', 'dense_5/bias:0'] when minimizing the loss. If y
WARNING:tensorflow:Gradients do not exist for variables ['dense_5/kernel:0', 'dense_5/bias:0'] when minimizing the loss. If y
WARNING:tensorflow:Gradients do not exist for variables ['dense_5/kernel:0', 'dense_5/bias:0'] when minimizing the loss. If y
WARNING:tensorflow:Gradients do not exist for variables ['dense_5/kernel:0', 'dense_5/bias:0'] when minimizing the loss. If y
9/9 [==============================] - 2s 53ms/step - loss: 0.6454 - accuracy: 0.7864 - val_loss: 0.6010 - val_accuracy: 0.86
Epoch 2/20
9/9 [==============================] - 0s 19ms/step - loss: 0.5683 - accuracy: 0.8667 - val_loss: 0.5286 - val_accuracy: 0.86
Epoch 3/20
9/9 [==============================] - 0s 24ms/step - loss: 0.4980 - accuracy: 0.8667 - val_loss: 0.4667 - val_accuracy: 0.86
Epoch 4/20
9/9 [==============================] - 0s 20ms/step - loss: 0.4436 - accuracy: 0.8667 - val_loss: 0.4262 - val_accuracy: 0.86
Epoch 5/20
9/9 [==============================] - 0s 19ms/step - loss: 0.4106 - accuracy: 0.8667 - val_loss: 0.4065 - val_accuracy: 0.86
Epoch 6/20
9/9 [==============================] - 0s 18ms/step - loss: 0.3965 - accuracy: 0.8667 - val_loss: 0.4001 - val_accuracy: 0.86
Epoch 7/20
9/9 [==============================] - 0s 23ms/step - loss: 0.3930 - accuracy: 0.8667 - val_loss: 0.3999 - val_accuracy: 0.86
Epoch 8/20
9/9 [==============================] - 0s 24ms/step - loss: 0.3927 - accuracy: 0.8667 - val_loss: 0.3999 - val_accuracy: 0.86
Epoch 9/20
9/9 [==============================] - 0s 23ms/step - loss: 0.3929 - accuracy: 0.8667 - val_loss: 0.3999 - val_accuracy: 0.86
Epoch 10/20
9/9 [==============================] - 0s 19ms/step - loss: 0.3927 - accuracy: 0.8667 - val_loss: 0.4000 - val_accuracy: 0.86
Epoch 11/20
9/9 [==============================] - 0s 18ms/step - loss: 0.3928 - accuracy: 0.8667 - val_loss: 0.4000 - val_accuracy: 0.86
Epoch 12/20
9/9 [==============================] - 0s 17ms/step - loss: 0.3926 - accuracy: 0.8667 - val_loss: 0.3999 - val_accuracy: 0.86
Epoch 13/20
9/9 [==============================] - 0s 18ms/step - loss: 0.3927 - accuracy: 0.8667 - val_loss: 0.4004 - val_accuracy: 0.86
Epoch 14/20
9/9 [==============================] - 0s 19ms/step - loss: 0.3928 - accuracy: 0.8667 - val_loss: 0.4004 - val_accuracy: 0.86
Epoch 15/20
9/9 [==============================] - 0s 18ms/step - loss: 0.3929 - accuracy: 0.8667 - val_loss: 0.4001 - val_accuracy: 0.86
Epoch 16/20
9/9 [==============================] - 0s 19ms/step - loss: 0.3931 - accuracy: 0.8667 - val_loss: 0.3999 - val_accuracy: 0.86
Epoch 17/20
9/9 [==============================] - 0s 23ms/step - loss: 0.3930 - accuracy: 0.8667 - val_loss: 0.4002 - val_accuracy: 0.86
Epoch 18/20
9/9 [==============================] - 0s 18ms/step - loss: 0.3926 - accuracy: 0.8667 - val_loss: 0.4014 - val_accuracy: 0.86
Epoch 19/20
9/9 [==============================] - 0s 23ms/step - loss: 0.3930 - accuracy: 0.8667 - val_loss: 0.3999 - val_accuracy: 0.86
```

```
Epoch 20/20
9/9 [==============================] - 0s 18ms/step - loss: 0.3932 - accuracy: 0.8667 - val_loss: 0.3999 - val_accuracy: 0.8
```

Performance Analysis of the approaches

For the purpose of this analysis, we can consider the 4 models created above: Sequential, RNN, LSTM, and CNN.

- The best performance was observed by the sequential model without involving any embedding or other RNN/CNN layers with an accuracy of 0.99.

- I attempted to increase the number of sequential layers for this model, however, the accuracy only seemed decrease. This can be in part due to the lack of a diverse training dataset, as the class labels of most datapoints are heavily skewed.

- I tried multiply methods for the CNN and RNN and the ones above performed the best. While the results were very high, I recognize that it could be because of the input data set. We can see that for the CNN, the accuracy is 86%. However, when taking a look at the training data, 13% of the text input is spam. That gives a small amount of spam emails to work with for training, validation, and testing, compared to non-spam. Therefore, when we consider the test results of all models, they succeeed. However, there is the concern that these models may not be robust to spam introduced as testing, if it is an unseen concept.

- The high accuracy of 87% can also be observed in the RNN and LSTM models as well. Now, the reason for this becomes clear as a case of overfitting. Since, the model is encountering 87% of non-spam messages during training, just specifing every single message as non-spam will itself guarantee a 87% accuracy which is what we see. This is confirmed by the recall rate specified in the classification report/confusion matrix. WE can see that the recall is extremely high for non-spam (=1) and extremely low (=0) for spam messages.

- The sequential model on the other hand, has a high recall for both spam and non-spam (.99 and .88). Thus, in terms of performance, the sequential model performs the best.

- I used a validation set as well on all models (taken from the testing set)

Note for CNN,RNN,LSTM:

Increasing the number of layers for CNN, (conv1D), actually had a detrimental effect in terms of the models accuracy. Removing the singular densd layer for input at the begining also greatly slowed down the training process along with the embedding layer as well.

Future Work: I would like to focus on incorporating GloVe pretrained embeddings into my model and check if this increases the accuracy of the model. The nature of the dataset might also impact the accuracy in this manner.