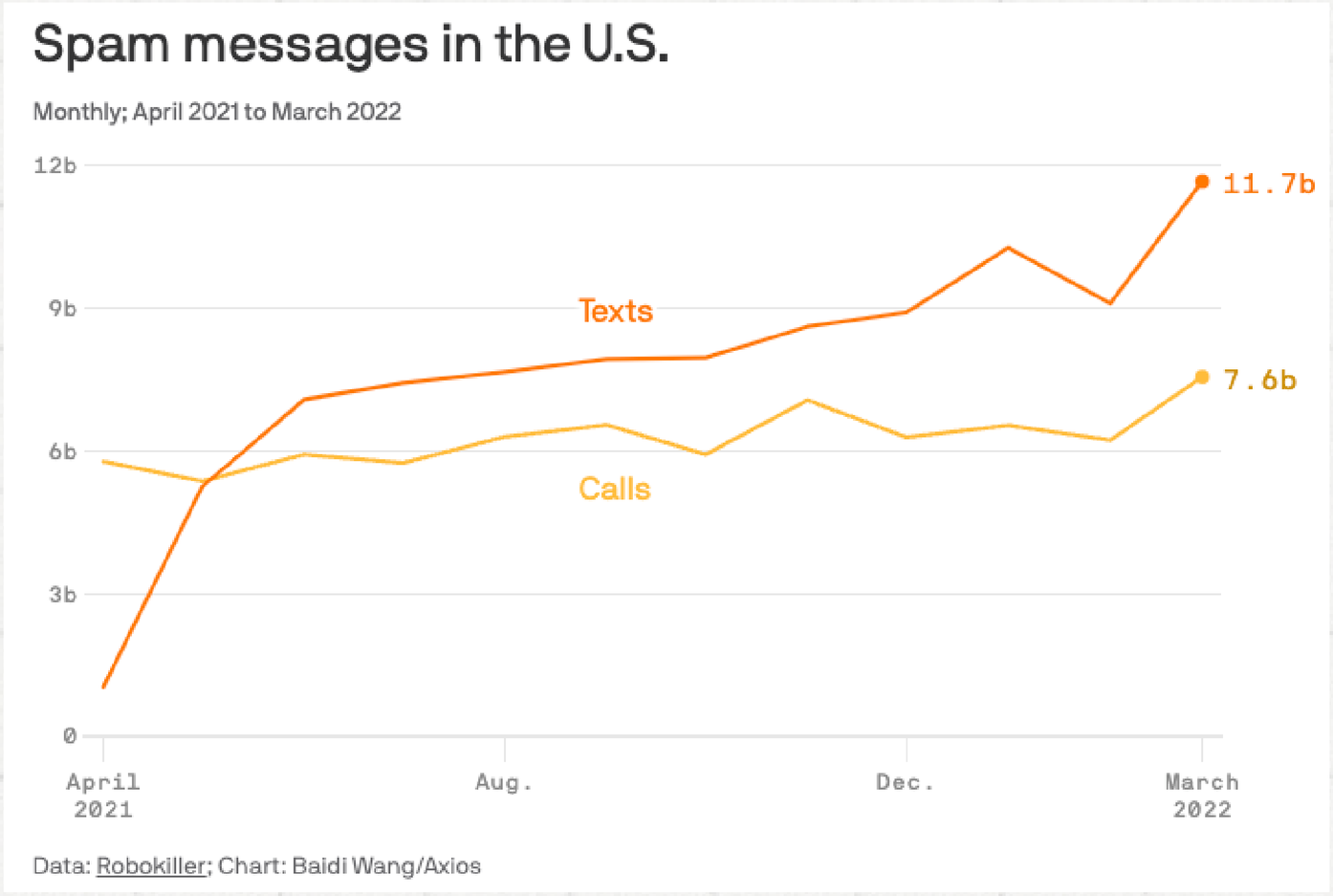
SC1015 Spam or Ham Email classifier project

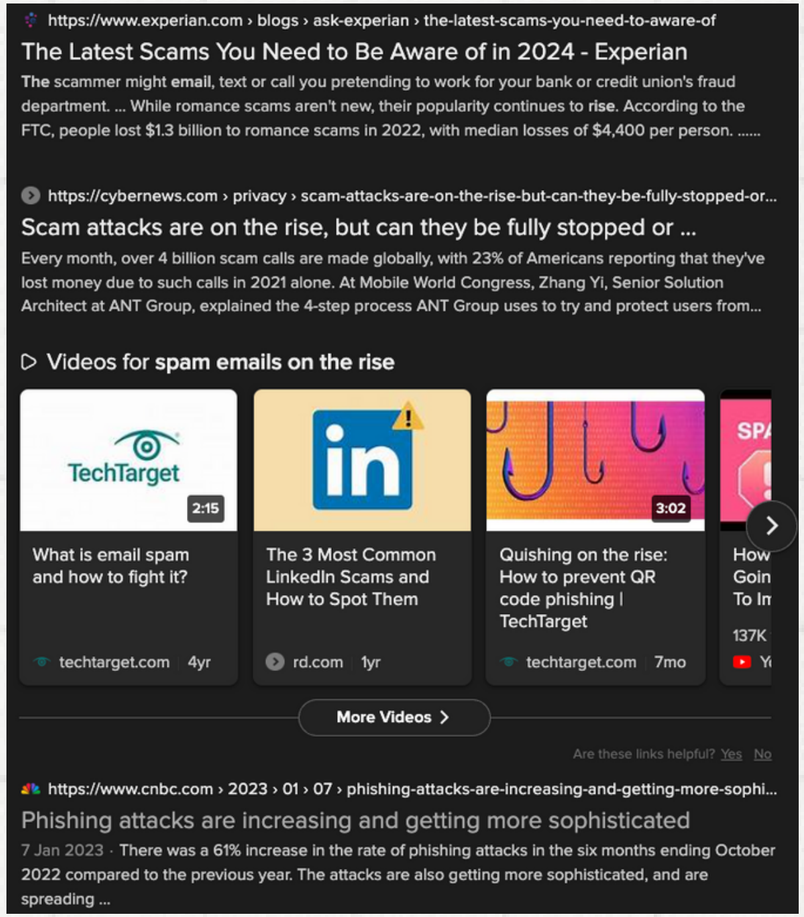
FDDA Team 9

Presented by Napatr, Wen Rong, Ivan

Content

Spam emails on the rise





Our Goal

To leverage on machine learning and data analysis to increase accurate spam or ham email classification

Our Data set

SHANTANU DHAKAD · UPDATED 2 YEARS AGO

62 | New Notebook | Download (216 kB)

Email Spam Detection Dataset (classification)

Spam/Ham Detection Dataset

Data Card | Code (38) | Discussion (1) | Suggestions (0)

About Dataset

Context

TO CLASSIFY THE MAIL AS SPAM OR HAM BY USING MACHINE OR DEEP LEARNING MODEL.

Content

This is the dataset in which some randomly mails are collected and classified as spam or ham .1st column contains spam/ham classification resr column have the mail itself

Usability 9.41

License
Other (specified in description)

Expected update frequency
Never

Tags

Our Data set

A1 v1

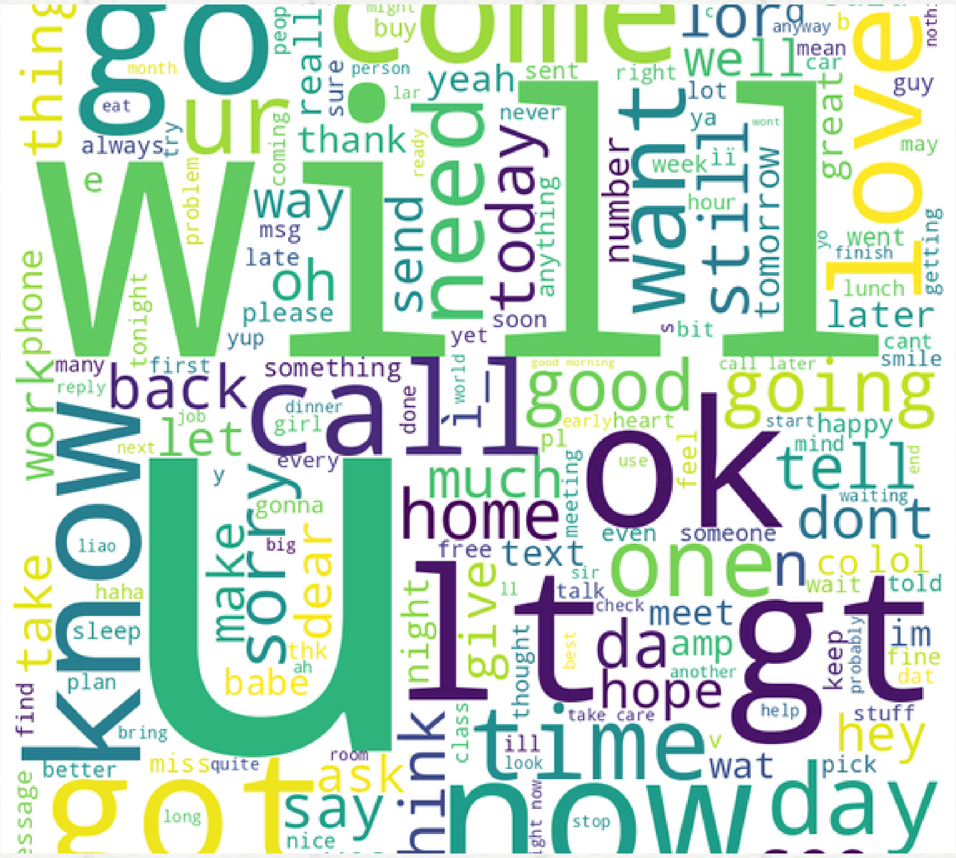
	A	B
1	v1	v2
2	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
3	ham	Ok lar... Joking wif u oni...
4	spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
5	ham	U dun say so early hor... U c already then say...
6	ham	Nah I don't think he goes to usf, he lives around here though
7	spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, å£1.50 to rcv
8	ham	Even my brother is not like to speak with me. They treat me like aids patent.
9	ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune
10	spam	WINNER!! As a valued network customer you have been selected to receivea å£900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.
11	spam	Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update Co FREE on 08002986030
12	ham	I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.
13	spam	SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info
14	spam	URGENT! You have won a 1 week FREE membership in our å£100,000 Prize Jackpot! Txt the word: CLAIM to No: 81010 T&C www.dbuk.net LCCLTD POBOX 4403LDNW1A7RW18
15	ham	I've been searching for the right words to thank you for this breather. I promise i wont take your help for granted and will fulfil my promise. You have been wonderful and a blessing at all times.
16	ham	I HAVE A DATE ON SUNDAY WITH WILL!!
17	spam	XXXMobileMovieClub: To use your credit, click the WAP link in the next txt message or click here>> http://wap. xxxmobilemovieclub.com?n=QJKGIGHJJGCBL
18	ham	Oh k...i'm watching here:)
19	ham	Eh u remember how 2 spell his name... Yes i did. He v naughty make until i v wet.
20	ham	Fine if thatåÕs the way u feel. ThatåÕs the way its gota b
21	spam	England v Macedonia - dont miss the goals/team news. Txt ur national team to 87077 eg ENGLAND to 87077 Try:WALES, SCOTLAND 4txt/Ì¼1.20 POBOXox36504W45WQ 16+
22	ham	Is that seriously how you spell his name?
23	ham	I‰Û÷m going to try for 2 months ha ha only joking
24	ham	So Ì_ pay first lar... Then when is da stock comin...
25	ham	Aft i finish my lunch then i go str down lor. Ard 3 smth lor. U finish ur lunch already?
26	ham	Ffffffffff. Alright no way I can meet up with you sooner?
27	ham	Just forced myself to eat a slice. I'm really not hungry tho. This sucks. Mark is getting worried. He knows I'm sick when I turn down pizza. Lol

Data Preparation: Cleaning

The three unnamed columns ('Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4') seem to be mostly empty and may not be relevant for our classification task.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB
(     v1                                                 v2 Unnamed: 2  \
 0   ham  Go until jurong point, crazy.. Available only ...         NaN
 1   ham                      Ok lar... Joking wif u oni...         NaN
 2  spam  Free entry in 2 a wkly comp to win FA Cup fina...         NaN
 3   ham  U dun say so early hor... U c already then say...         NaN
 4   ham  Nah I don't think he goes to usf, he lives aro...         NaN

  Unnamed: 3 Unnamed: 4
0        NaN        NaN
1        NaN        NaN
2        NaN        NaN
3        NaN        NaN
4        NaN        NaN
None)
```

Exploratory Data Analysis- Word Cloud



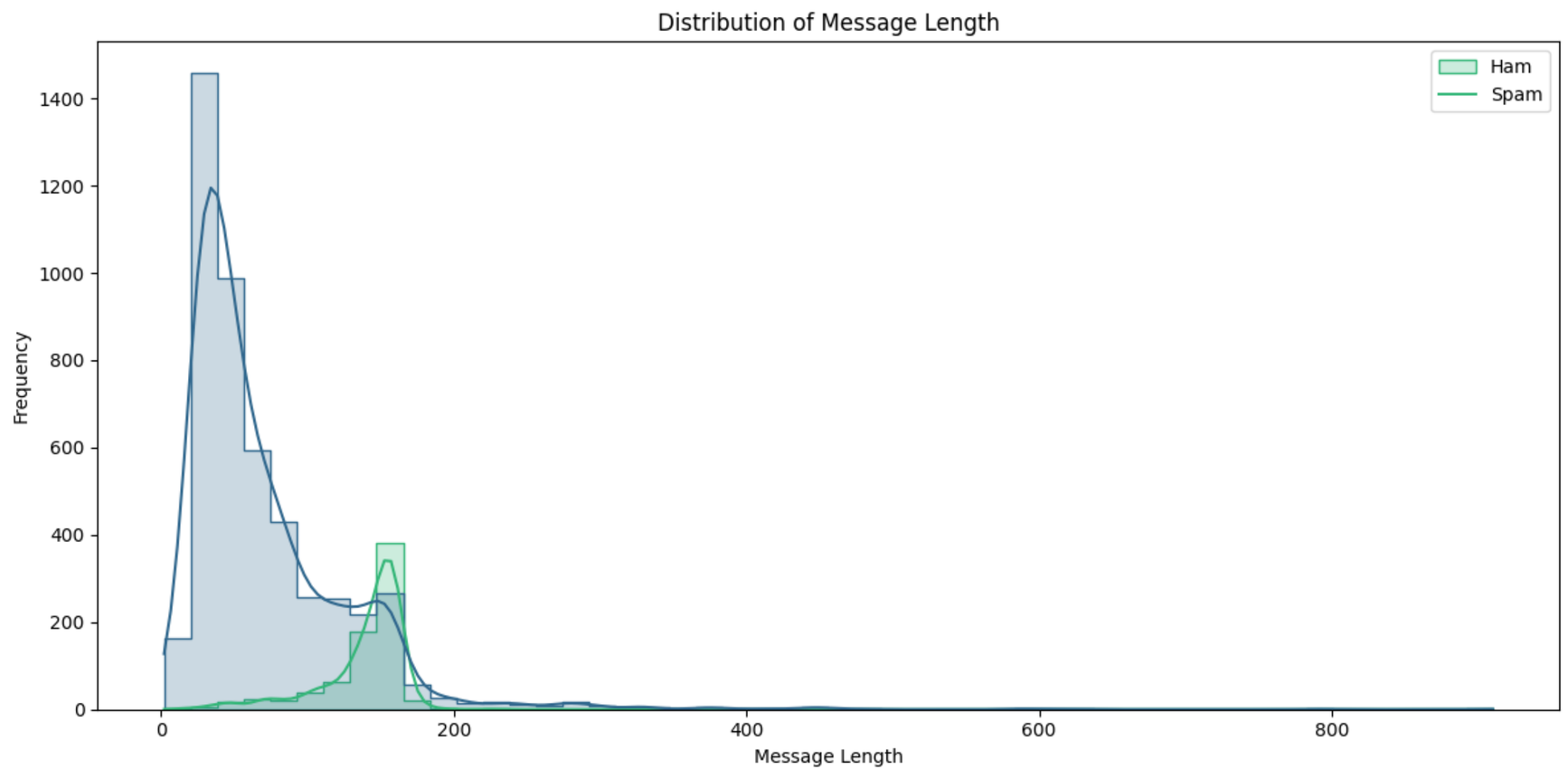
What insights can we derive based on the most frequently appearing words?

1. Spam emails has spelling errors
2. Action words, sense of urgency (Call, Now)
3. Some overlapping words (Now, Call)
4. Inconclusive based on a single word

Question:

How to increase the classification accuracy?
What are other exploration methods to use?

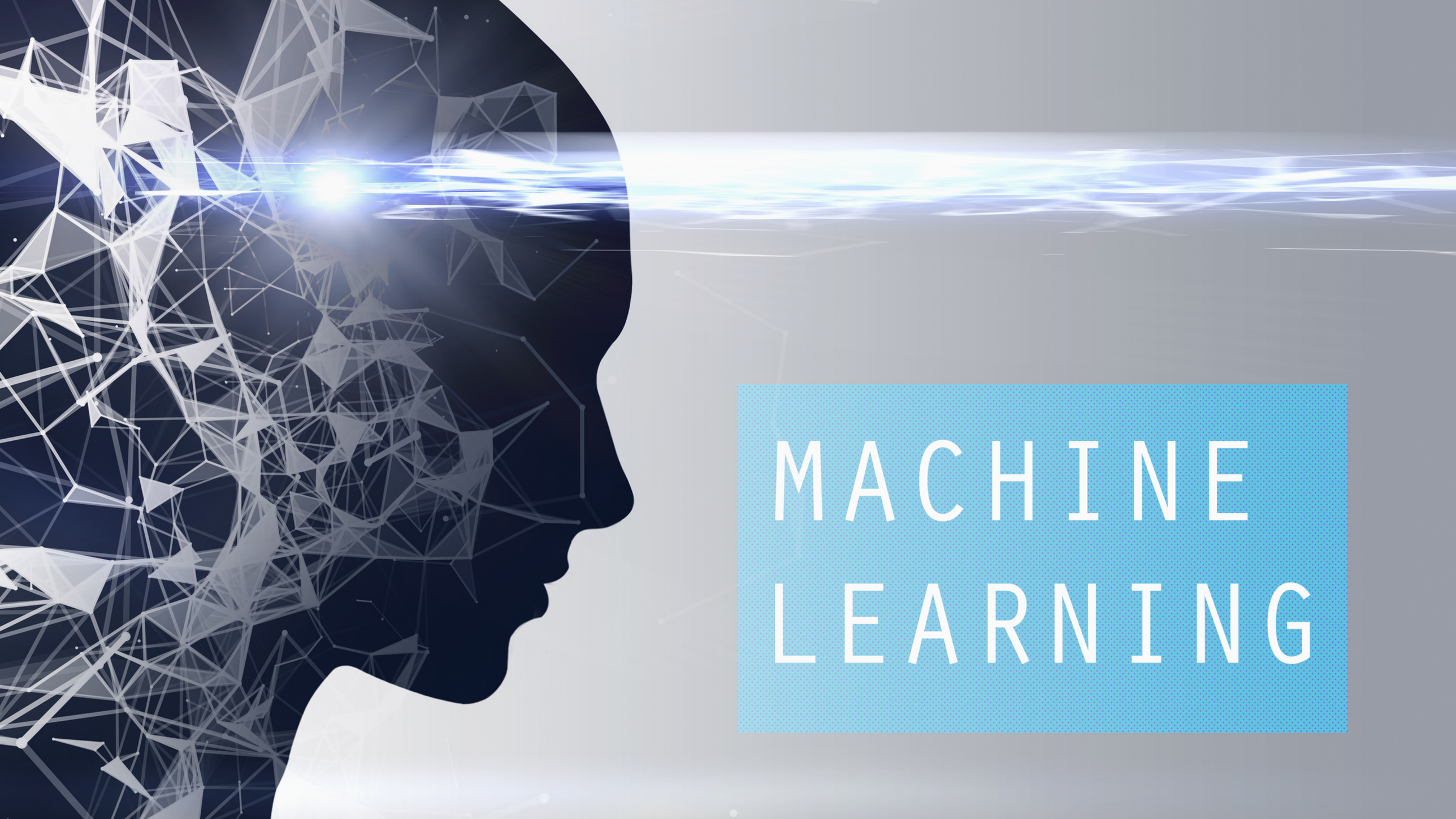
Exploratory Data Analysis- Email length



Exploratory Data Analysis- TF-IDF

```
free        0.060179
txt         0.045359
mobile      0.039427
text        0.038554
stop        0.037292
claim       0.036077
ur          0.033210
reply       0.032289
prize       0.032068
www         0.031167
new         0.027198
uk          0.026345
won         0.025745
cash        0.025137
150p        0.025043
service     0.024639
urgent      0.022786
win         0.022625
50          0.021397
nokia       0.021276
dtype: float64
```

1. Higher value == stronger indicator of Spam content
2. correlation ≠ causation
3. furthur analysis must be performed to increase confidence level of findings


MACHINE
LEARNING

Machine Learning - splitting dataset

1. Use train test split:

```
# Split the dataset into training and testing sets, set seed = 42
X_train, X_test, y_train, y_test = train_test_split(data_clean['text'], data_clean['label'], test_size=0.2, random_state=42)
```

2. The test set had Ham count of 965 and Spam count of 150, making it unbalanced.

```
#check balance of test set
def check(y_test):
  ham_count = 0
  spam_count = 0
  for label in y_test:
    if label == 'ham':
      ham_count += 1
    else:
      spam_count += 1
  print("Ham count:",ham_count, "Spam count:",spam_count)

check(y_test)
```

```
Ham count: 965 Spam count: 150
```

3. Removed the last 815 Ham values from test set and added it back into train set, new test set has an equal Ham and Spam count of 150.

```
Ham count: 150 Spam count: 150
y_test:
8      spam
12     spam
15     spam
23      ham
33      ham
Name: label, dtype: object
X_test:
8      winner!! as a valued network customer you have...
12     urgent! you have won a 1 week free membership ...
15     xxxmobilemovieclub: to use your credit, click ...
23     aft i finish my lunch then i go str down lor. ...
33     for fear of fainting with the of all that hous...
```

Machine Learning - Vectorization

1. Count vectorization (each sentence is represented as a vector where each dimension corresponds to the count of a specific word):

```
# Vectorize the text data
vectorizer = CountVectorizer()
X_train_count = vectorizer.fit_transform(X_train)
X_test_count = vectorizer.transform(X_test)

print(X_train_count.toarray())
```

```
[[0 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

2. TF-IDF vectorization (each sentence is represented as a vector where each dimension corresponds to the TF-IDF score of a specific word):

```
# Vectorize the text data using TF-IDF
vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

print(X_train_tfidf.toarray())
```

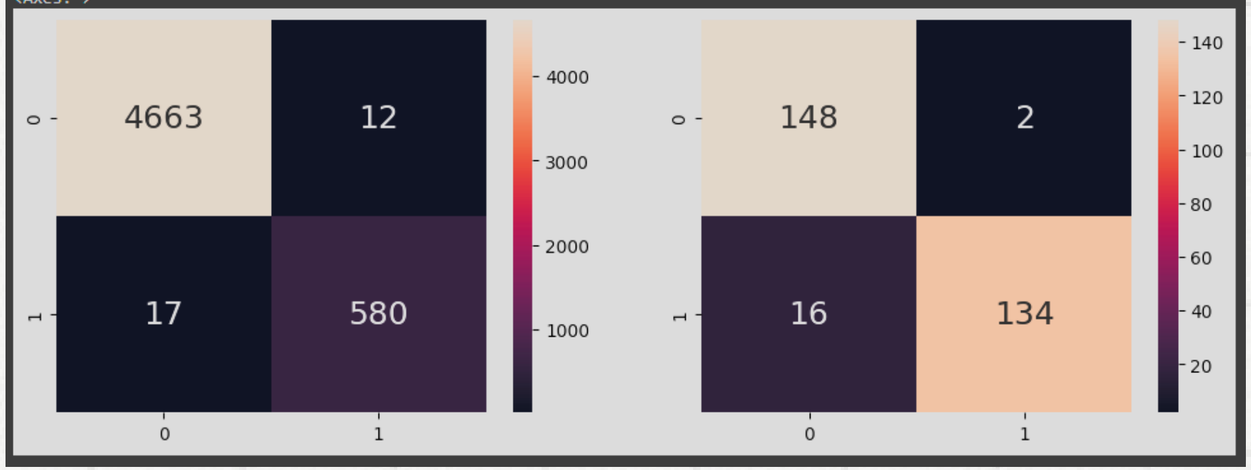
```
[[0.         0.         0.         ... 0.         0.         0.        ]
 [0.24509223 0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 ...
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]]
```

Machine Learning - Naive Bayes Classifier

- Naive Bayes classification uses Bayes' theorem to calculate the conditional probability of a document being either "ham" or "spam" given its vectorized representation. By Naively assuming independence between features (words), Naive Bayes efficiently computes the likelihood of each class and assigns the class with the highest probability to the document.
- Used MultinomialNB()

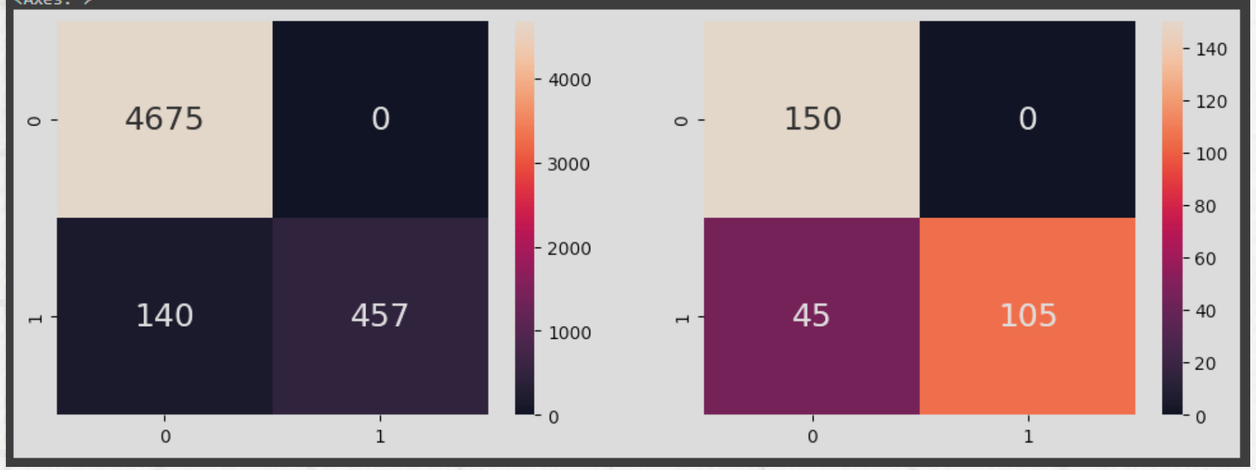
Count Vectorization

```
Accuracy on train set: 0.9944992412746586
Accuracy on test set: 0.94

Classification Report on test set:
              precision    recall  f1-score   support

         ham       0.90      0.99      0.94       150
        spam       0.99      0.89      0.94       150

    accuracy                           0.94       300
   macro avg       0.94      0.94      0.94       300
weighted avg       0.94      0.94      0.94       300

<Axes: >
<Axes: >
```



TF-IDF Vectorization

```
Accuracy on train set: 0.9734446130500759
Accuracy on test set: 0.85

Classification Report on test set:
              precision    recall  f1-score   support

         ham       0.77      1.00      0.87       150
        spam       1.00      0.70      0.82       150

    accuracy                           0.85       300
   macro avg       0.88      0.85      0.85       300
weighted avg       0.88      0.85      0.85       300

<Axes: >
<Axes: >
```

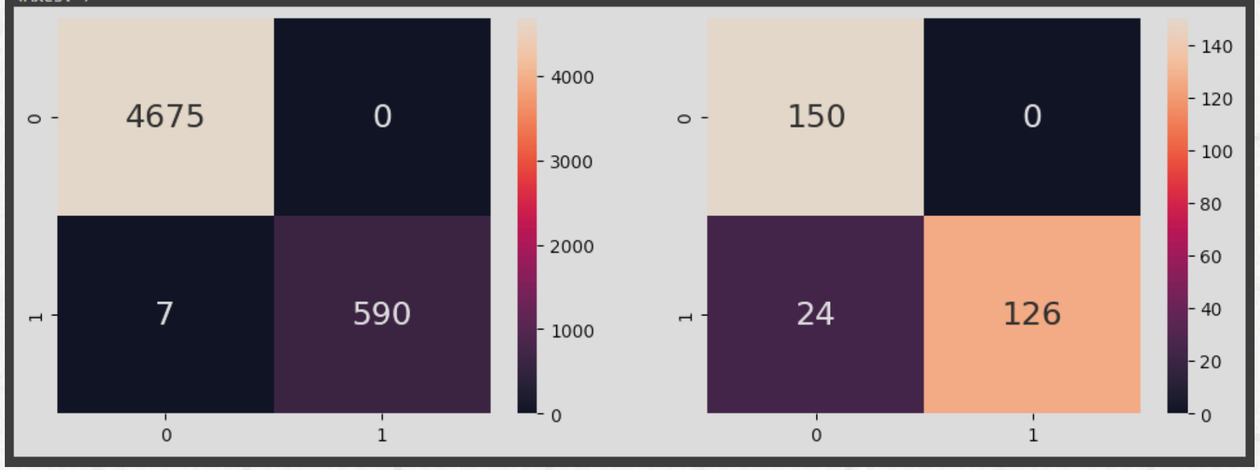


Machine Learning - Logistic Regression Classifier

- Logistic regression classifies vectorized text as "ham" or "spam" by fitting a logistic function to the vectorized features, estimating the probability of each class. The model then assigns the document to the class with the highest predicted probability, typically using a threshold of 0.5.
- Used LogisticRegression()

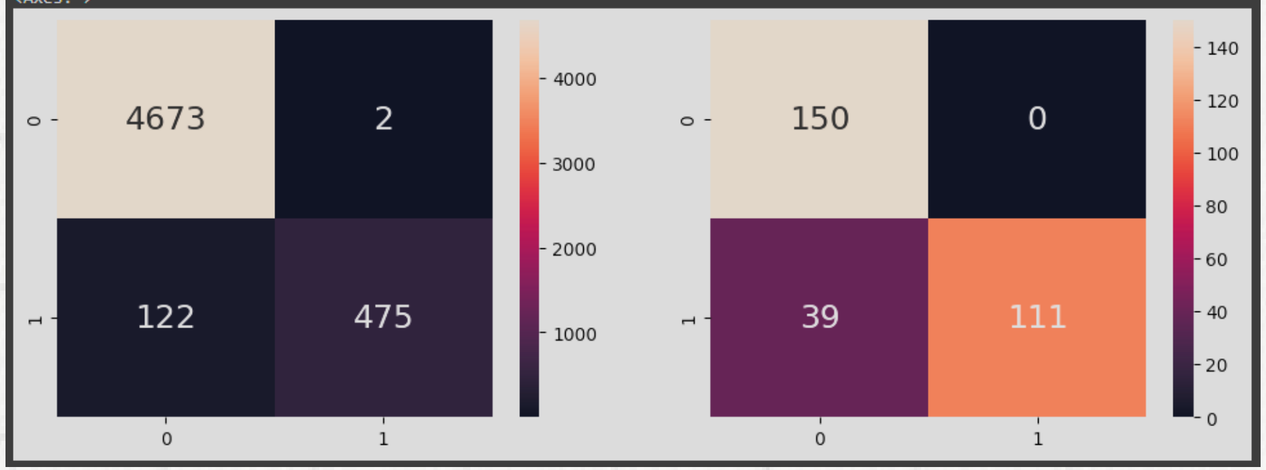
Count Vectorization

```
Accuracy on train set: 0.9986722306525038
Accuracy on test set: 0.92

Classification Report on test set:
              precision    recall  f1-score   support

         ham       0.86      1.00      0.93       150
        spam       1.00      0.84      0.91       150

    accuracy                           0.92       300
   macro avg       0.93      0.92      0.92       300
weighted avg       0.93      0.92      0.92       300

<Axes: >
<Axes: >
```



TF-IDF Vectorization

```
Accuracy on train set: 0.9764795144157815
Accuracy on test set: 0.87

Classification Report on test set:
              precision    recall  f1-score   support

         ham       0.79      1.00      0.88       150
        spam       1.00      0.74      0.85       150

    accuracy                           0.87       300
   macro avg       0.90      0.87      0.87       300
weighted avg       0.90      0.87      0.87       300

<Axes: >
<Axes: >
```

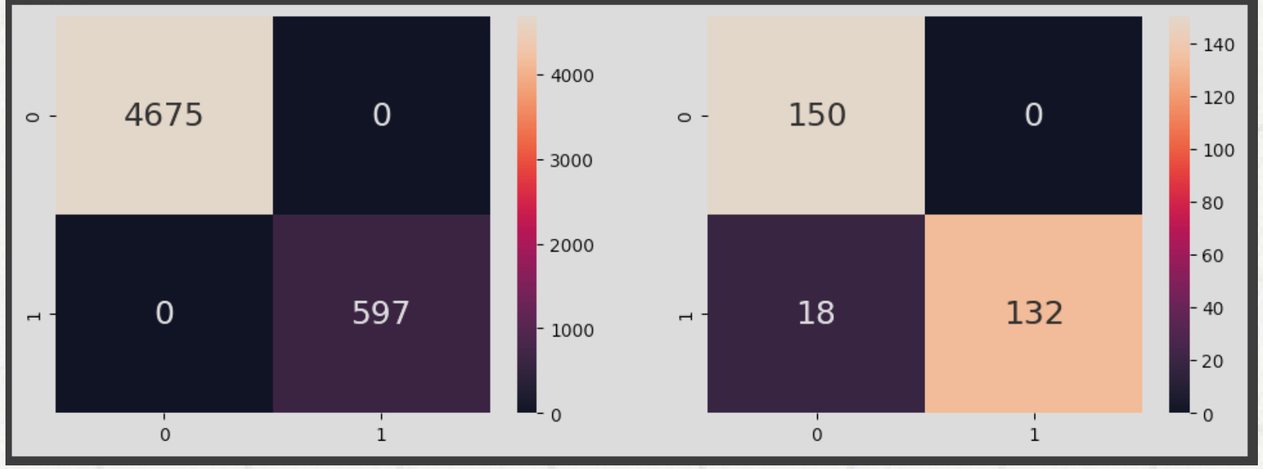


Machine Learning - SVM Classifier

- Support Vector Machine (SVM) classifies vectorized text as "ham" or "spam" by finding the hyperplane that best separates the two classes in the high-dimensional space defined by the vectorized features. The document is then assigned to a class based on which side of the hyperplane it falls on.
- Used SVC(kernel='linear')

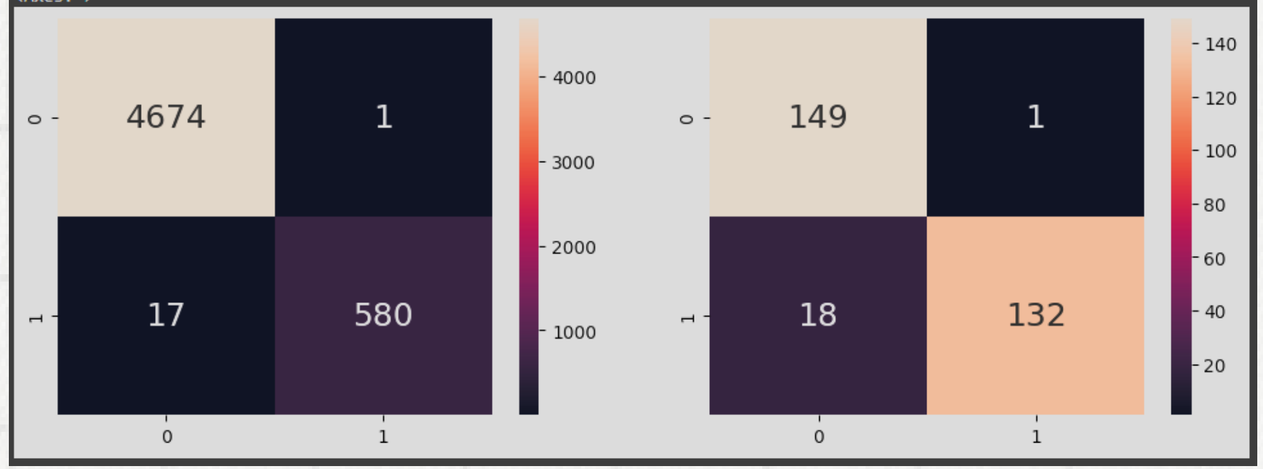
Count Vectorization

```
Accuracy on train set: 1.0
Accuracy on test set: 0.94

Classification Report on test set:
              precision    recall  f1-score   support

         ham       0.89      1.00      0.94       150
        spam       1.00      0.88      0.94       150

    accuracy                           0.94       300
   macro avg       0.95      0.94      0.94       300
weighted avg       0.95      0.94      0.94       300

<Axes: >
<Axes: >
```



TF-IDF Vectorization

```
Accuracy on train set: 0.9965857359635811
Accuracy on test set: 0.9366666666666666

Classification Report on test set:
              precision    recall  f1-score   support

         ham       0.89      0.99      0.94       150
        spam       0.99      0.88      0.93       150

    accuracy                           0.94       300
   macro avg       0.94      0.94      0.94       300
weighted avg       0.94      0.94      0.94       300

<Axes: >
<Axes: >
```

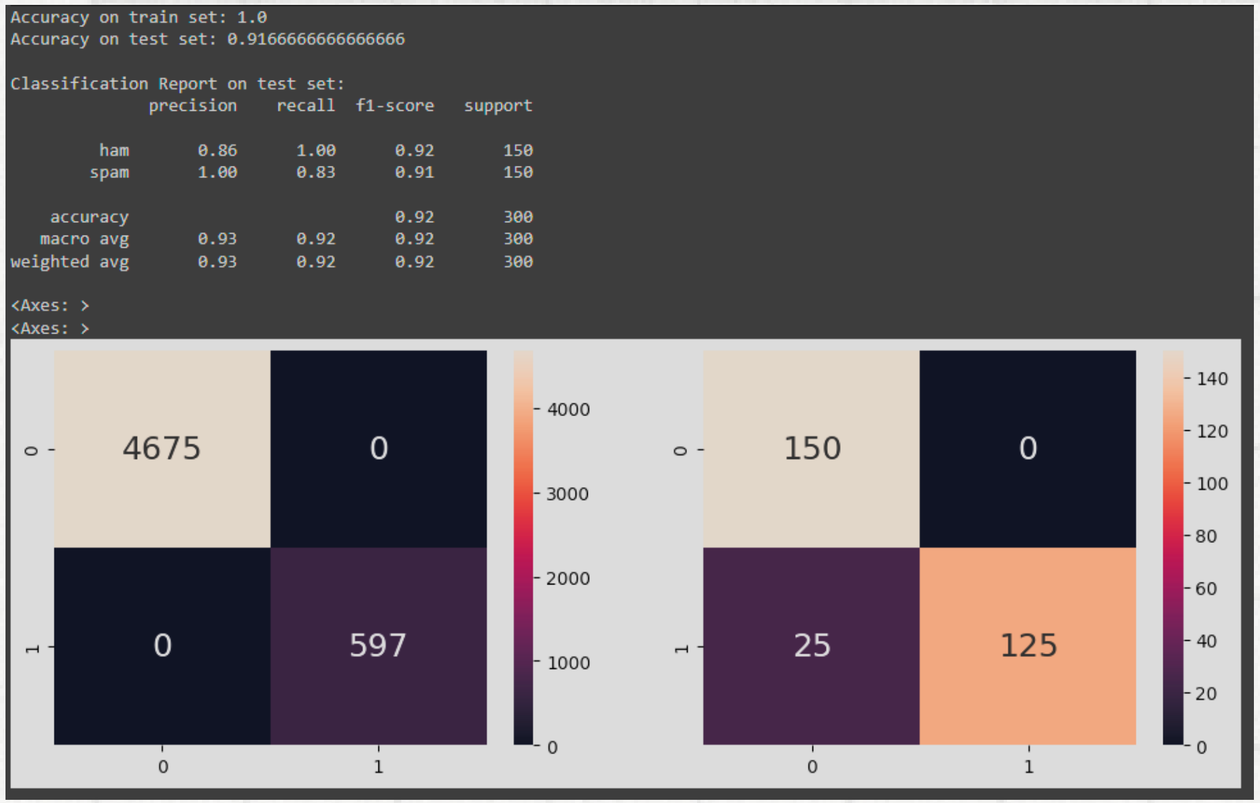


Machine Learning - Random Forest Classifier

- Random Forest classifies vectorized text as "ham" or "spam" by constructing an ensemble of decision trees, where each tree is trained on a random subset of features and data samples. The final classification is determined by a majority vote of all the trees in the forest.
- Used RandomForestClassifier(n_estimators=100, random_state=42)

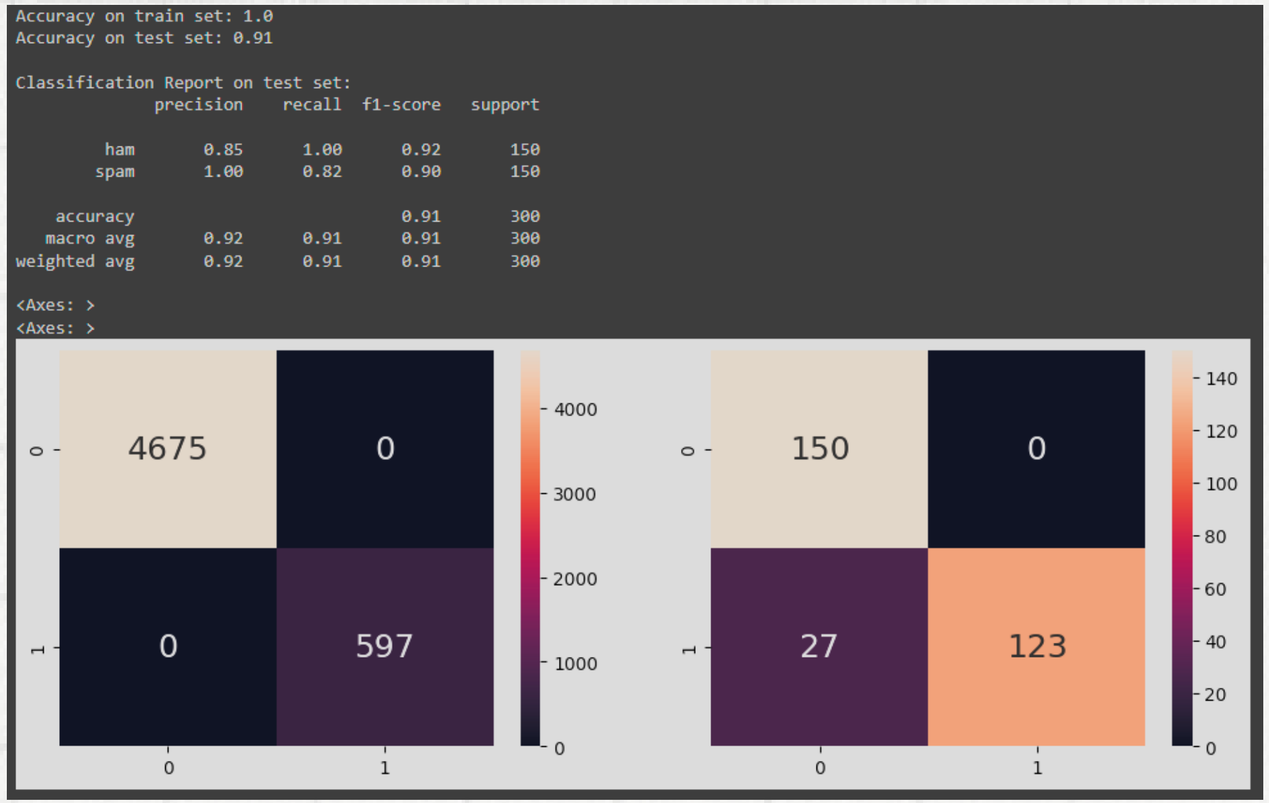
Count Vectorization

```
Accuracy on train set: 1.0
Accuracy on test set: 0.9166666666666666

Classification Report on test set:
              precision    recall  f1-score   support

         ham       0.86      1.00      0.92       150
        spam       1.00      0.83      0.91       150

    accuracy                           0.92       300
   macro avg       0.93      0.92      0.92       300
weighted avg       0.93      0.92      0.92       300

<Axes: >
<Axes: >
```



TF-IDF Vectorization

```
Accuracy on train set: 1.0
Accuracy on test set: 0.91

Classification Report on test set:
              precision    recall  f1-score   support

         ham       0.85      1.00      0.92       150
        spam       1.00      0.82      0.90       150

    accuracy                           0.91       300
   macro avg       0.92      0.91      0.91       300
weighted avg       0.92      0.91      0.91       300

<Axes: >
<Axes: >
```



Machine Learning - Neural Network (2 layers)

- The first layer has 64 neurons with a ReLU activation function and expects input data with a dimension equal to the number of features in the training data (`input_dim=X_train_count.shape[1]`).
- The second layer has 2 neurons with a sigmoid activation function, which is commonly used for binary classification tasks. The sigmoid activation function outputs probabilities for each class, typically representing the probability of the document belonging to each class ("ham" or "spam").

Count Vectorization

```
Epoch 1/10
165/165 [==============================] - 2s 9ms/step - loss: 0.2913 - accuracy: 0.9435
Epoch 2/10
165/165 [==============================] - 2s 9ms/step - loss: 0.0490 - accuracy: 0.9930
Epoch 3/10
165/165 [==============================] - 2s 10ms/step - loss: 0.0168 - accuracy: 0.9979
Epoch 4/10
165/165 [==============================] - 2s 11ms/step - loss: 0.0078 - accuracy: 0.9992
Epoch 5/10
165/165 [==============================] - 2s 9ms/step - loss: 0.0044 - accuracy: 0.9998
Epoch 6/10
165/165 [==============================] - 3s 15ms/step - loss: 0.0028 - accuracy: 1.0000
Epoch 7/10
165/165 [==============================] - 3s 18ms/step - loss: 0.0019 - accuracy: 1.0000
Epoch 8/10
165/165 [==============================] - 2s 13ms/step - loss: 0.0014 - accuracy: 1.0000
Epoch 9/10
165/165 [==============================] - 2s 12ms/step - loss: 0.0011 - accuracy: 1.0000
Epoch 10/10
165/165 [==============================] - 3s 18ms/step - loss: 8.4417e-04 - accuracy: 1.0000
<keras.src.callbacks.History at 0x787076b68730>

Train set:
Train Loss: 0.0007148910663090646
Train Accuracy: 1.0

Test set:
Test Loss: 0.376123309135437
Test Accuracy: 0.9266666769981384
```

TF-IDF Vectorization

```
Epoch 1/10
165/165 [==============================] - 2s 8ms/step - loss: 0.3038 - accuracy: 0.9131
Epoch 2/10
165/165 [==============================] - 2s 10ms/step - loss: 0.0561 - accuracy: 0.9879
Epoch 3/10
165/165 [==============================] - 3s 16ms/step - loss: 0.0213 - accuracy: 0.9954
Epoch 4/10
165/165 [==============================] - 2s 11ms/step - loss: 0.0102 - accuracy: 0.9983
Epoch 5/10
165/165 [==============================] - 1s 8ms/step - loss: 0.0056 - accuracy: 0.9996
Epoch 6/10
165/165 [==============================] - 1s 8ms/step - loss: 0.0035 - accuracy: 0.9996
Epoch 7/10
165/165 [==============================] - 1s 9ms/step - loss: 0.0023 - accuracy: 1.0000
Epoch 8/10
165/165 [==============================] - 1s 8ms/step - loss: 0.0017 - accuracy: 1.0000
Epoch 9/10
165/165 [==============================] - 2s 9ms/step - loss: 0.0012 - accuracy: 1.0000
Epoch 10/10
165/165 [==============================] - 3s 15ms/step - loss: 9.5389e-04 - accuracy: 1.0000
<keras.src.callbacks.History at 0x787048279150>

Train set:
Train Loss: 0.0007859576726332307
Train Accuracy: 1.0

Test set:
Test Loss: 0.2467339038848877
Test Accuracy: 0.9366666674613953
```

Machine Learning - Neural Network with dropout

- Uses the same 2 layers as before with an additional dropout layer in-between
- The dropout layer randomly drops a fraction of input units during training, helping to prevent overfitting by promoting the learning of more robust features.

Count Vectorization

```
Epoch 1/10
165/165 [==============================] - 4s 12ms/step - loss: 0.3168 - accuracy: 0.9274
Epoch 2/10
165/165 [==============================] - 1s 9ms/step - loss: 0.0716 - accuracy: 0.9877
Epoch 3/10
165/165 [==============================] - 2s 10ms/step - loss: 0.0335 - accuracy: 0.9937
Epoch 4/10
165/165 [==============================] - 2s 12ms/step - loss: 0.0187 - accuracy: 0.9973
Epoch 5/10
165/165 [==============================] - 2s 9ms/step - loss: 0.0097 - accuracy: 0.9992
Epoch 6/10
165/165 [==============================] - 1s 7ms/step - loss: 0.0061 - accuracy: 0.9992
Epoch 7/10
165/165 [==============================] - 1s 8ms/step - loss: 0.0044 - accuracy: 1.0000
Epoch 8/10
165/165 [==============================] - 1s 8ms/step - loss: 0.0030 - accuracy: 0.9998
Epoch 9/10
165/165 [==============================] - 1s 8ms/step - loss: 0.0024 - accuracy: 1.0000
Epoch 10/10
165/165 [==============================] - 1s 7ms/step - loss: 0.0023 - accuracy: 0.9996
<keras.src.callbacks.History at 0x787076dfdde0>

Train set:
Train Loss: 0.000983307370916009
Train Accuracy: 1.0

Test set:
Test Loss: 0.34099963307380676
Test Accuracy: 0.9366666674613953
```

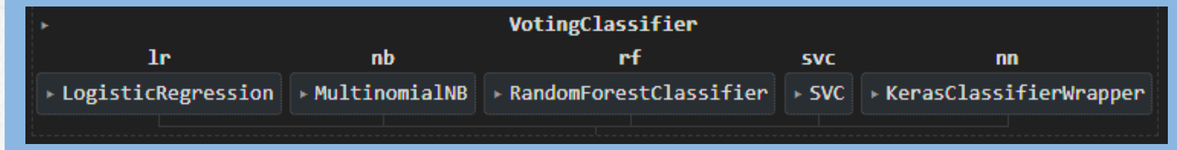
TF-IDF Vectorization

```
Epoch 1/10
165/165 [==============================] - 2s 9ms/step - loss: 0.3359 - accuracy: 0.8932
Epoch 2/10
165/165 [==============================] - 1s 8ms/step - loss: 0.0927 - accuracy: 0.9752
Epoch 3/10
165/165 [==============================] - 1s 8ms/step - loss: 0.0400 - accuracy: 0.9913
Epoch 4/10
165/165 [==============================] - 1s 9ms/step - loss: 0.0247 - accuracy: 0.9947
Epoch 5/10
165/165 [==============================] - 2s 12ms/step - loss: 0.0156 - accuracy: 0.9968
Epoch 6/10
165/165 [==============================] - 2s 9ms/step - loss: 0.0105 - accuracy: 0.9981
Epoch 7/10
165/165 [==============================] - 1s 9ms/step - loss: 0.0066 - accuracy: 0.9992
Epoch 8/10
165/165 [==============================] - 1s 8ms/step - loss: 0.0049 - accuracy: 0.9991
Epoch 9/10
165/165 [==============================] - 1s 9ms/step - loss: 0.0036 - accuracy: 0.9996
Epoch 10/10
165/165 [==============================] - 2s 12ms/step - loss: 0.0026 - accuracy: 1.0000
<keras.src.callbacks.History at 0x78703af70730>

Train set:
Train Loss: 0.0014252258697524667
Train Accuracy: 1.0

Test set:
Test Loss: 0.24432456493377686
Test Accuracy: 0.9366666674613953
```

Machine Learning - Combining models

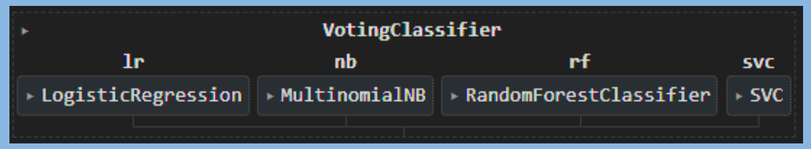
- VotingClassifier combines the predictions of multiple individual classifiers by majority vote (hard voting) or averaging of probabilities (soft voting), typically resulting in improved overall performance and robustness.

TF-IDF Vectorization

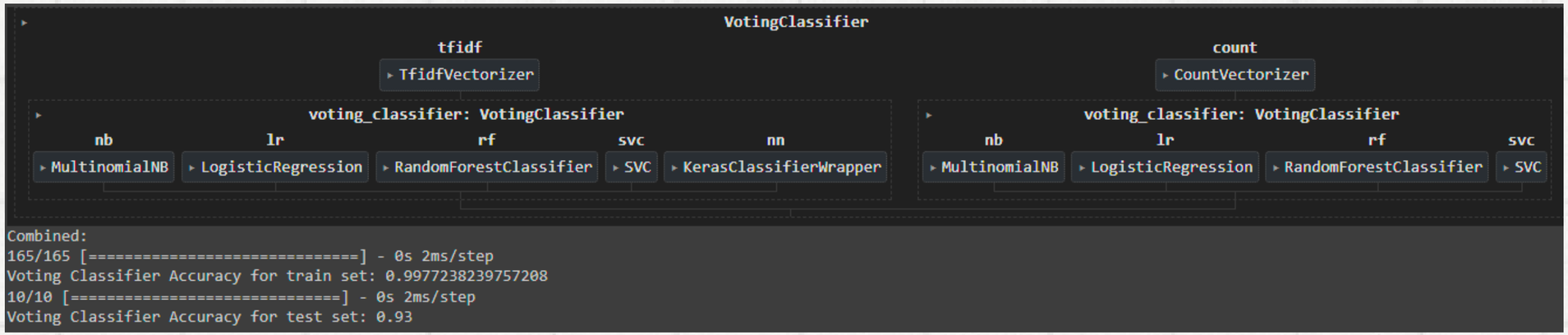



```
TF-IDF:
165/165 [==============================] - 0s 2ms/step
Voting Classifier Accuracy for train set: 0.9963960546282246
10/10 [==============================] - 0s 2ms/step
Voting Classifier Accuracy for test set: 0.9266666666666666
```

Count Vectorization




```
Count Vectorization:
Voting Classifier Accuracy for train set: 1.0
Voting Classifier Accuracy for test set: 0.9366666666666666
```



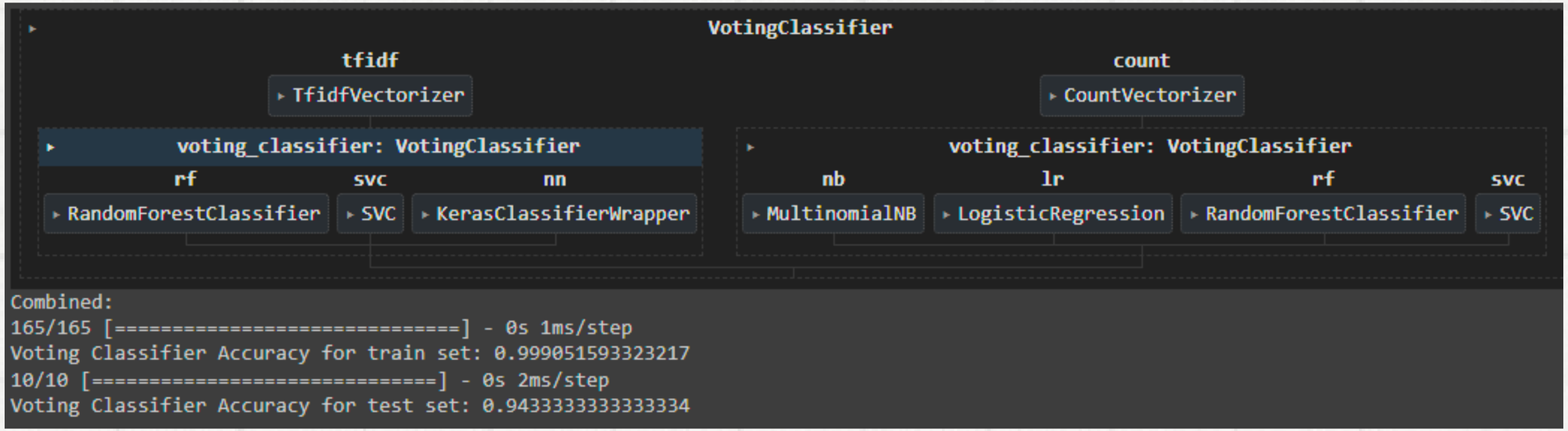

```
Combined:
165/165 [==============================] - 0s 2ms/step
Voting Classifier Accuracy for train set: 0.9977238239757208
10/10 [==============================] - 0s 2ms/step
Voting Classifier Accuracy for test set: 0.93
```

(soft voting)

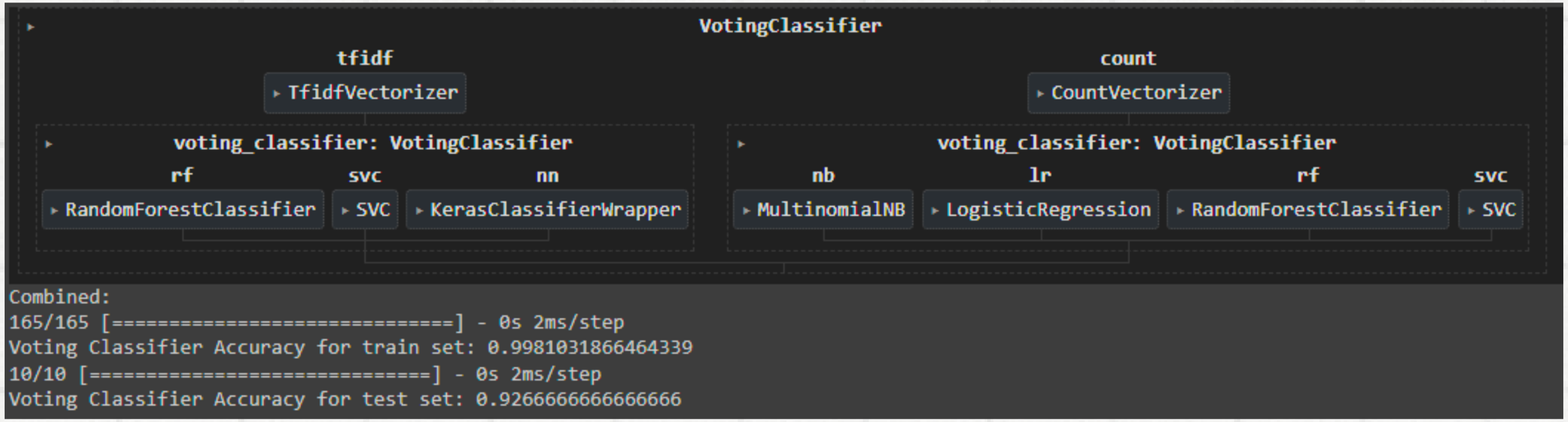
Machine Learning - improving combined model

- For TF-IDF, Naive Bayes and Logistic Regression had the lowest accuracies on the test set (0.85 and 0.87 respectively), so they were removed from the model

Soft voting:



Hard voting:



Data-Driven Insights

Data-Driven Insights

Count Vectorization

Support Vector Machine (SVM) Classifier Accuracy: **0.94**
Naive Bayes Classifier Accuracy: **0.94**
Neural Networks using Keras Accuracy with Droput Layer: **0.937**
Logistic Regression Accuracy: **0.92**
Random Forest Classifier Accuracy: **0.917**

TF-IDF

Support Vector Machine (SVM) Classifier Accuracy: **0.937**
Neural Networks using Keras Accuracy with DropOut Layer : **0.937**.
Random Forest Classifier Accuracy: **0.91**
Logistic Regression Accuracy: **0.87**
Naive Bayes Classifier Accuracy: **0.85**

Key Details

SVM Classifier showing the highest Accuracy for both Count Vectorization and TF-IDF

- In spam email detection, the feature space can be quite high-dimensional due to the large vocabulary size from TF-IDF or Count Vectorization. Hence, SVMs can effectively separate spam and non-spam emails in this high-dimensional space, making them well-suited for spam detection tasks.

Poor Accuracy for Naives Bayes Classifier and Logistic Regression with TF-IDF

- TF-IDF's weighted representation can violate the feature independence assumption of Naive Bayes. The weights assigned by TF-IDF capture the interactions between words, which can lead to dependencies between features that Naive Bayes fails to capture due to its independence assumption.

- The weighted representation can introduce non-linearity into the feature space, which might not be effectively captured by Logistic Regression's linear decision boundary assumption.

Data-Driven Insights

Why Count Vectorization > TF-IDF

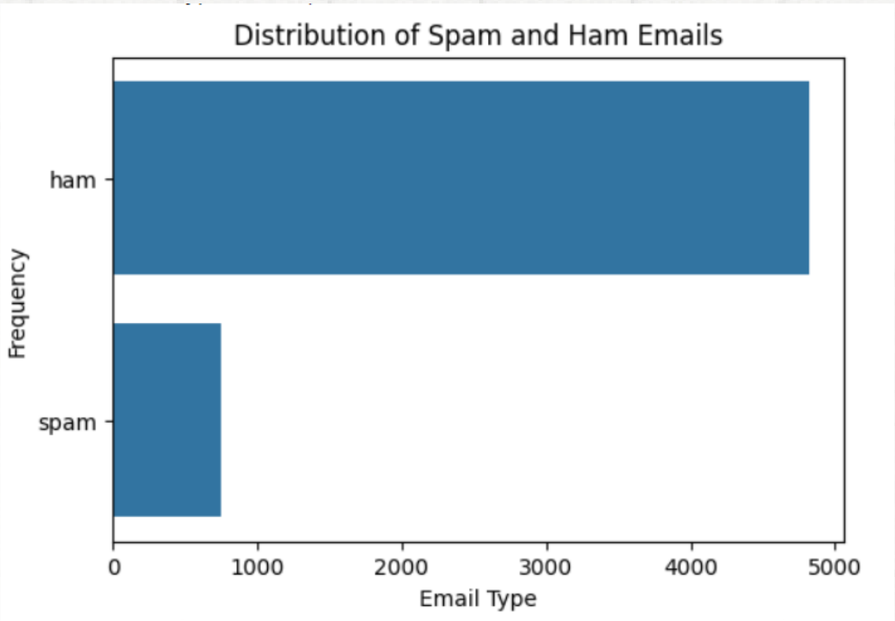
The low amounts of training data for our project favor Count Vectorization over TF-IDF.

Overfitting Concerns:

- With limited spam email data, the high dimensionality introduced by TF-IDF can increase the risk of overfitting. The model might learn noise from the limited training data, leading to poor generalization to new, unseen spam emails.

Count Vectorization's Simplicity:

- Count Vectorization offers a straightforward representation by counting the frequency of each word in the spam emails. This direct and simple representation can capture the essential spam-related patterns and relationships between words more effectively with limited data



Recommendations

Implementing a Combination of Classifiers with Soft Voting

- The combination of all classifiers excluding Naive Bayes and Logistic Regression with Soft Voting provided the Highest Classification Accuracy of 0.943, which is higher than any of the uncombined classifiers

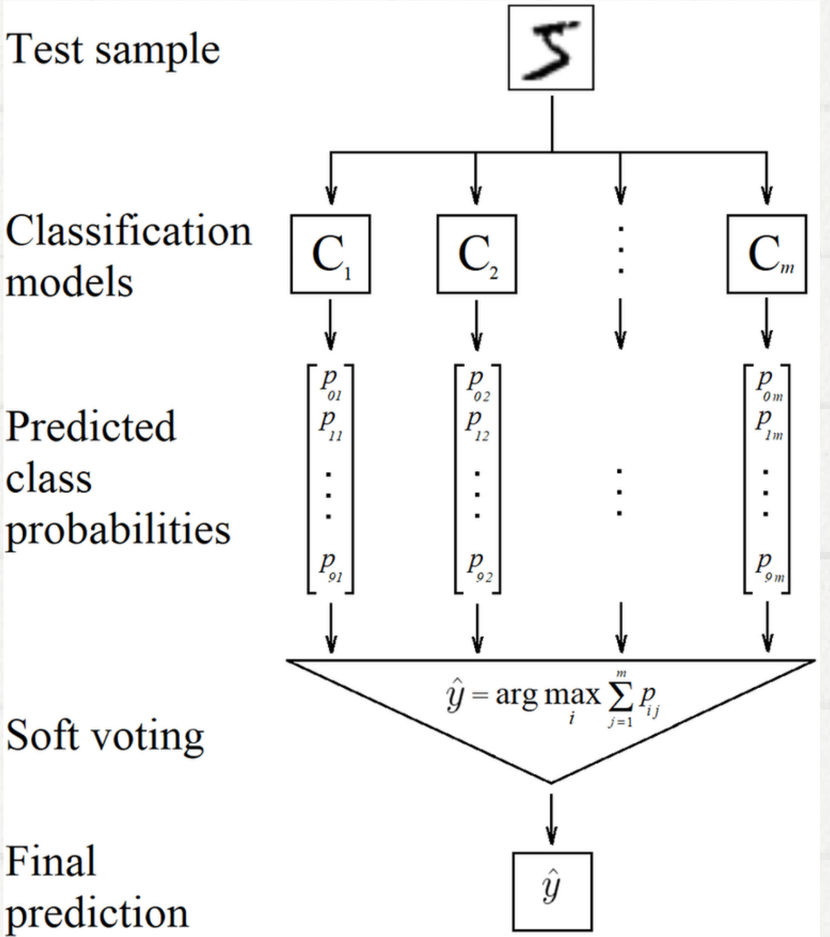
How Soft Voting improves classification accuracy of spam emails:

Leveraging Diverse Classifiers:

- Soft Voting combines predictions from diverse classifiers, each capturing unique aspects and patterns in spam emails. This diversity allows the ensemble to identify a broader range of spam characteristics, enhancing the classification's robustness and accuracy

Overfitting Mitigation:

- Soft Voting helps to counteract the tendencies of individual classifiers to overfit to noise or specific patterns in the training data by averaging out predictions across classifiers. This leads to better generalization and improved performance on new, unseen spam emails by focusing on genuine spam indicators.



Future Outcomes

Development of new Combinations of Models

- By including more models on a voting classifier with a larger data sample, we can enhance the model's generalization ability. This will allow us to fulfill our project outcome, which is to improve the accuracy of spam email classification.

Thank you very much!

Contributions

Ivan – Slides, code and script for 1. Problem Formulation, 2. Data Preparation and 3. Exploratory Data Analysis

Wen Rong – Code, slides, video for 4. Machine Learning, video editor for Ivan's part and overall video editor

Napatr – Slides and video for 5. Data Driven Insights and 6. Evaluation, and recorded video for Ivan's part

Citations

Gupta, S. (2023, July 13). Email spam filtering using naive Bayes classifier. Springboard Blog. https://www.springboard.com/blog/data-science/bayes-spam-filter/

Shrivas, A. K., Dewangan, A. K., & Ghosh, S. M. (2021). Robust Text Classifier for Classification of Spam E-Mail Documents with Feature Selection Technique. Ingénierie Des Systèmes D’information (2001), 26(5), 437–444. https://doi.org/10.18280/isi.260502