Use for animations

https://www.canva.com/design/DAFnyGXWVWQ/ceQOI4EYoM1BixCjz_G0Mg/view?utm_content=DAFnyGXWVWQ&utm_campaign=designshare&utm_medium=link&utm_source=publishsharelink

ADSM

(Almost) Detecting Spam Messages

CHAPTER I

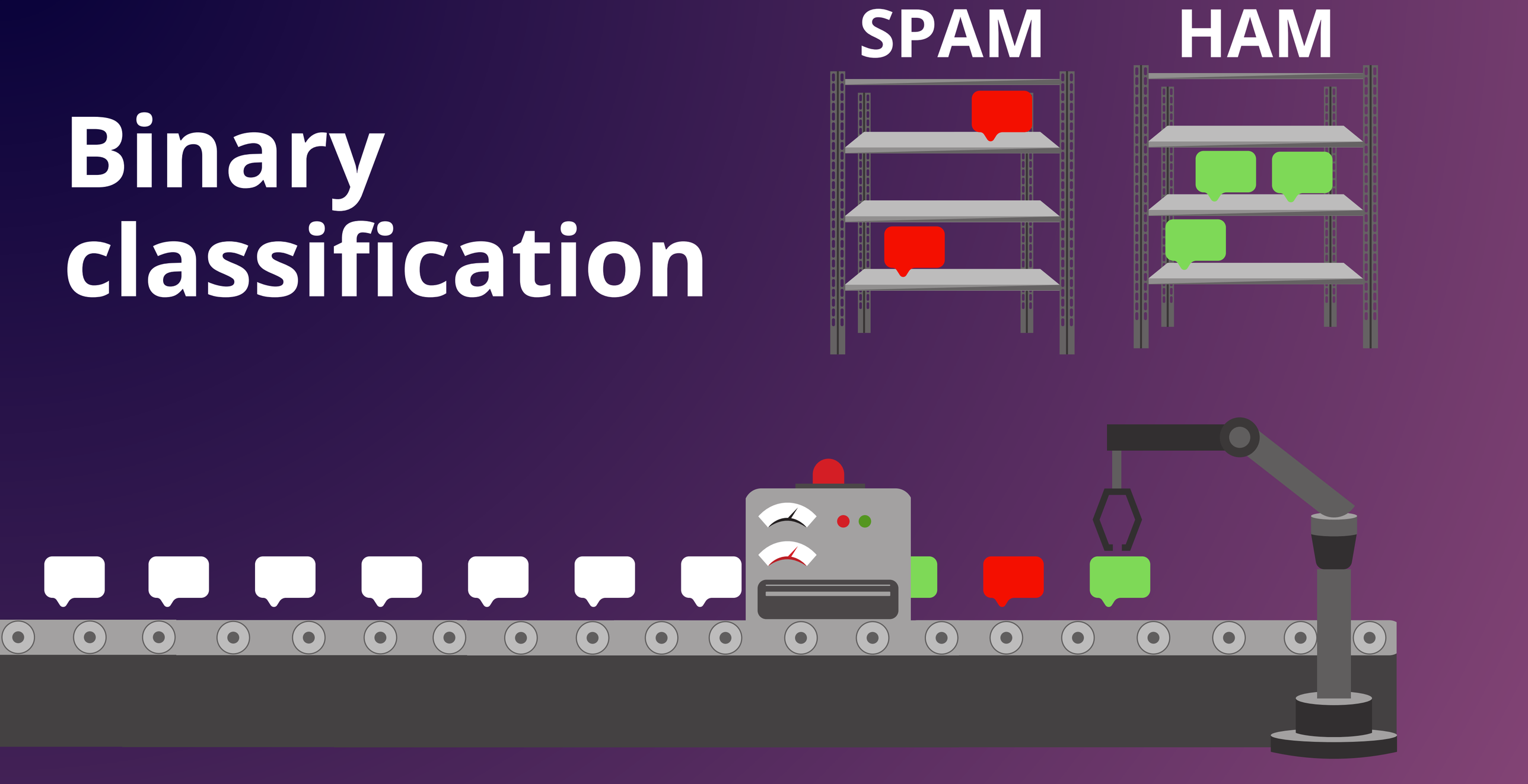
OUR GOAL

(ALMOST) DETECTING SPAM MESSAGES

01

Text Mining and NLP

Binary classification



CHAPTER II

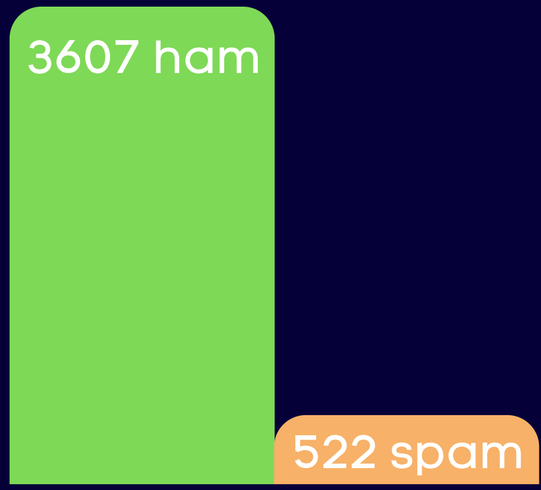
OUR DATASET

(ALMOST) DETECTING SPAM MESSAGES

02

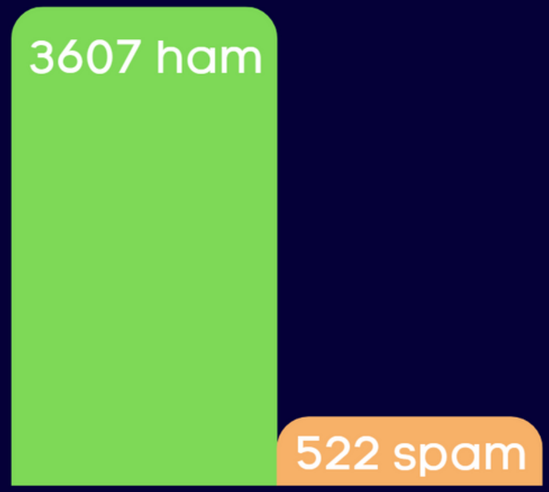
Text Mining and NLP

UNBALANCED CLASSES



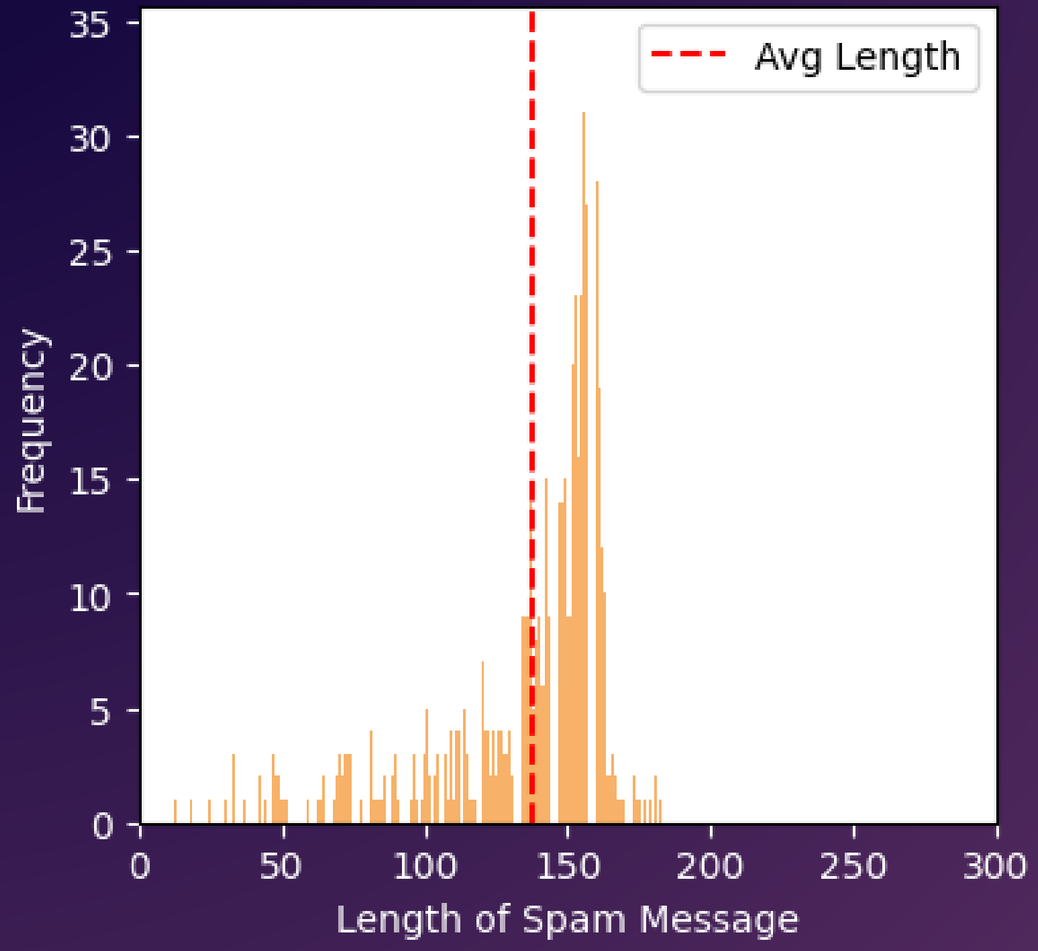
UNBALANCED CLASSES

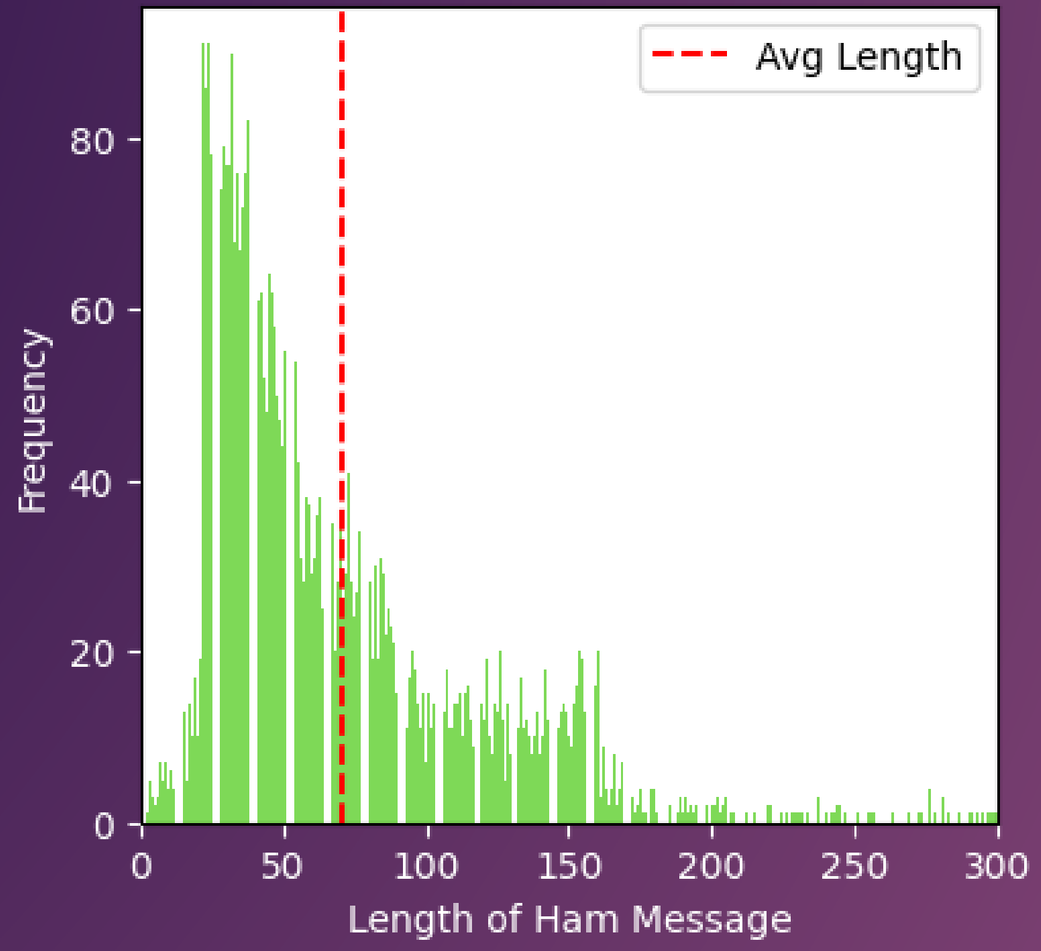
RandomOverSampler



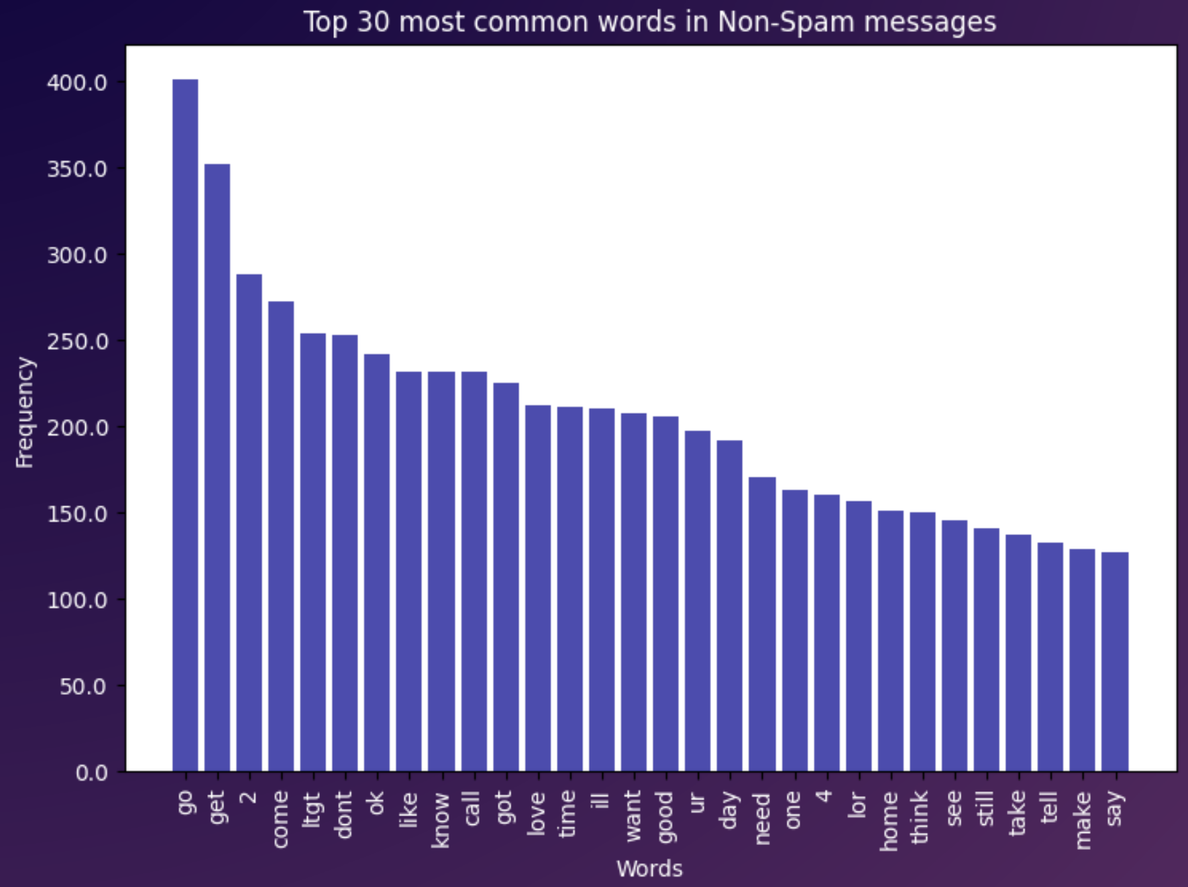

```
ros = RandomOverSampler(sampling_strategy=0.8,
    random_state=42)

x_train_resampled, y_train_resampled = ros
    .fit_resample(x_train.to_frame(), y_train)
```

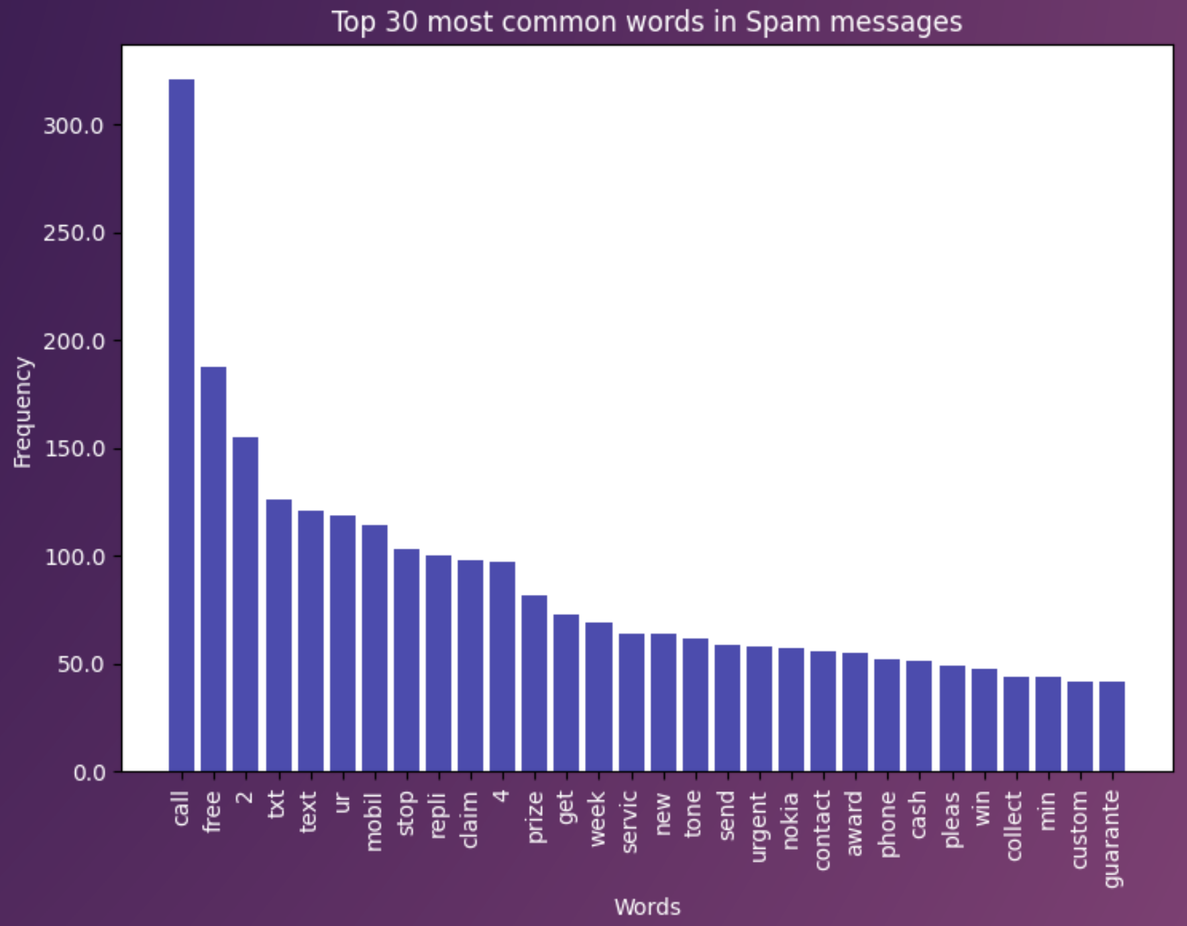
LENGTH OF MESSAGES



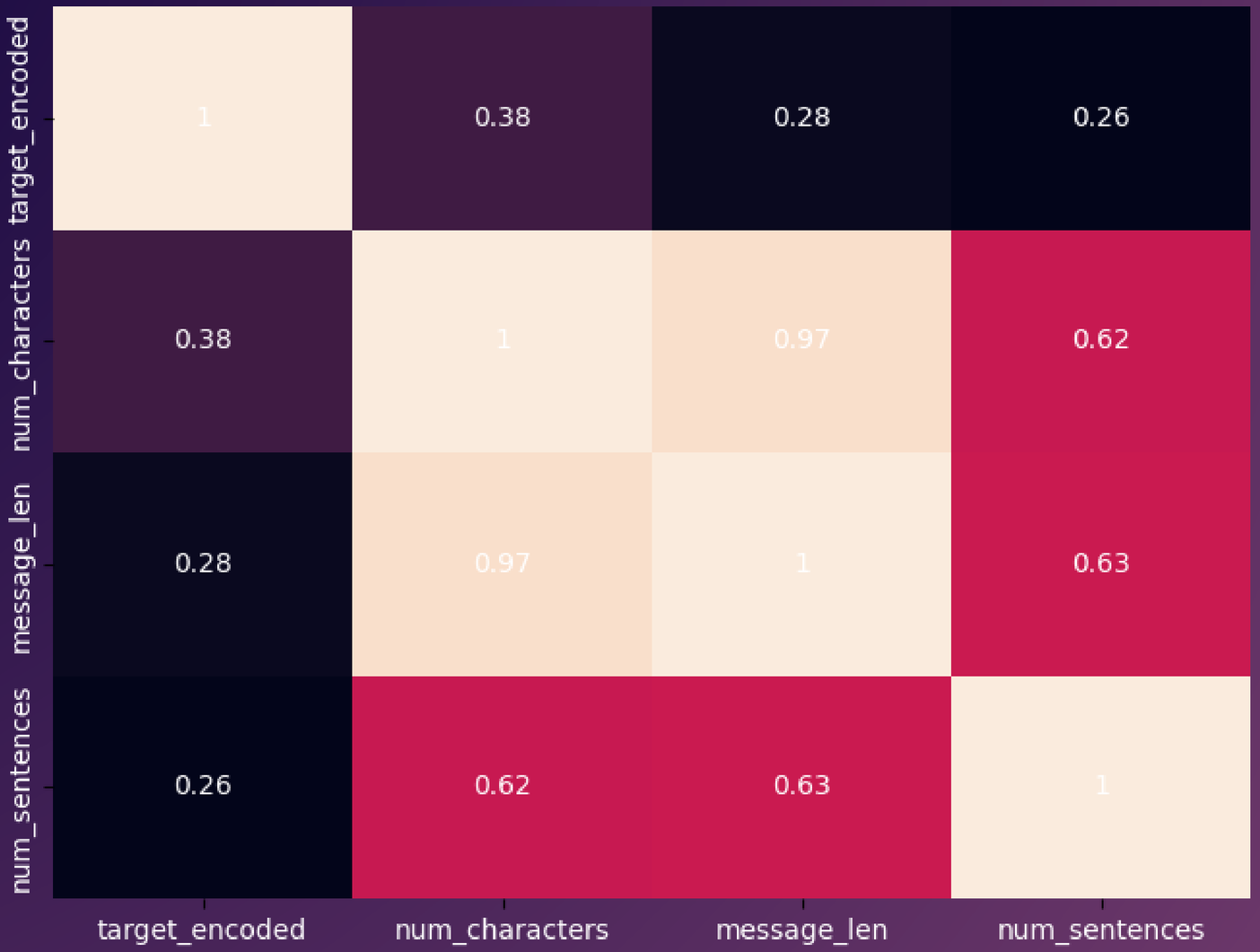


COMMON WORDS





CORRELATION MAP



CHAPTER III

PREPROCESSING

(ALMOST) DETECTING SPAM MESSAGES

03

Text Mining and NLP

Text clean  ing

```
1    def clean_text(text):
```

Text cleaning 

```
1    def clean_text(text):
2        text = str(text).lower()
```

Text cleaning 

```
1    def clean_text(text):
2        text = str(text).lower()
3        text = re.sub(r'[^\x00-\x7F]+', '', text) # remove non ascii chracter
```

Text cleaning 

```
1    def clean_text(text):
2        text = str(text).lower()
3        text = re.sub(r'[^\x00-\x7F]+', '', text) # remove non ascii chracter
4        text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
```

Text cleaning 

```
1    def clean_text(text):
2        text = str(text).lower()
3        text = re.sub(r'[^\x00-\x7F]+', '', text) # remove non ascii chracter
4        text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
5        #text = re.sub('\[.*?\]', '', text)
6        #text = re.sub('https?://\S+|www\.\S+', '', text)
7        #text = re.sub('<.*?>+', '', text)
8        #text = re.sub('\n', '', text)
9        #text = re.sub('\w*\d\w*', '', text)
10
11       return text
```

```
1  df.head()
```

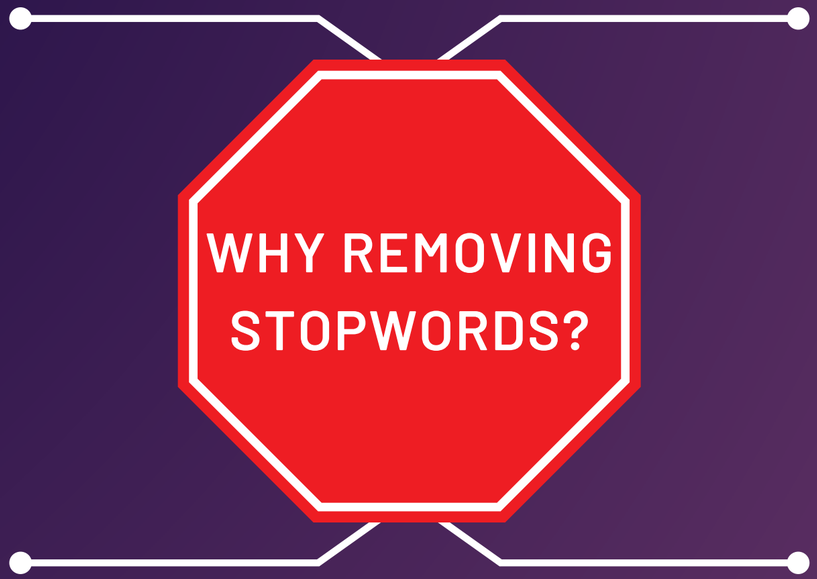
target	message	target_encoded	message_len	num_characters	num_sentences	message_clean
ham	Go until jurong point, crazy.. Available only ...	0	20	111	2	go until jurong point crazy available only in ...
ham	Ok lar... Joking wif u oni...	0	6	29	2	ok lar joking wif u oni...
spam	Free entry in 2 a wkly comp to win FA Cup fina...	1	28	155	2	free entry in 2 a wkly comp to win fa cup fina...
ham	U dun say so early hor... U c already then say...	0	11	49	1	u dun say so early hor u c already then say
ham	Nah I don't think he goes to usf, he lives aro...	0	13	61	1	nah i dont think he goes to usf he lives aroun...


'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're",
"you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them
theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll"
those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had',
do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as
while'
during
off'
why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some
no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'wi
don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain
'aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',
hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't",
'mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't"
STOPWORDS

Stopwords

Far too common

Thanks to Zipf's law and Luhn's analysis, we know these words are irrelevant.



Dimensionality reduction

Reduce the number of features and simplify the representation of the text data.

Focus on other words

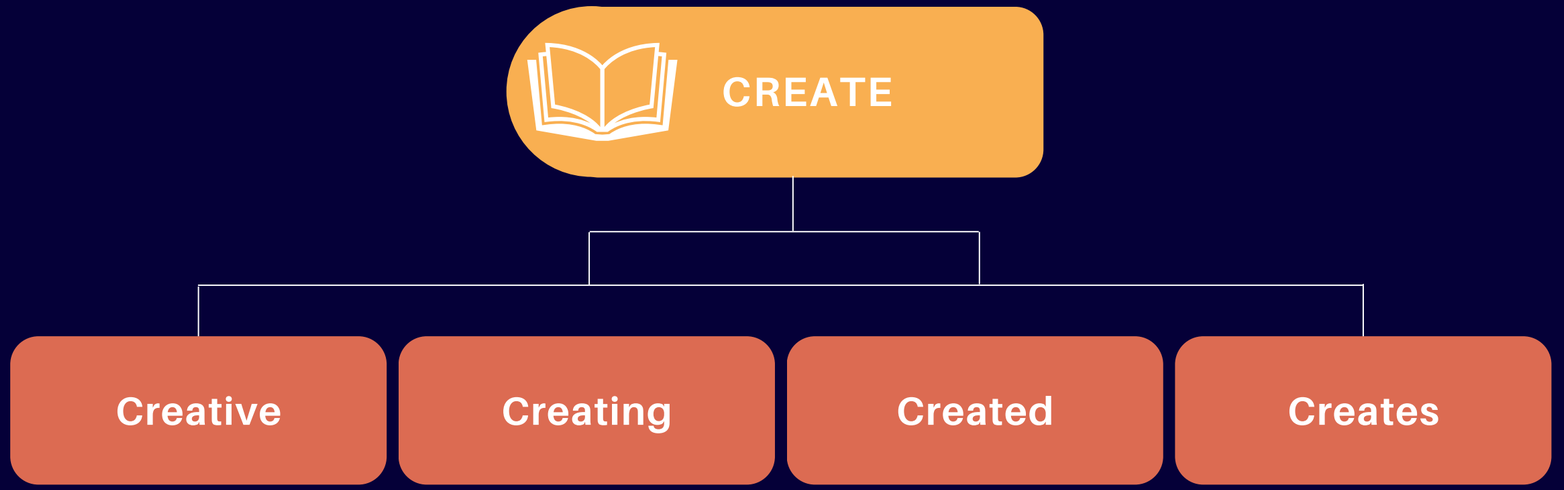
Other words are much more important for our task, we should focus on those.

Useless abbreviations

Additionally remove useless abbreviations too common in sms such as *k* (ok) or *u* (you).

Stemmer

```
1    stemmer = nltk.SnowballStemmer("english")
2
3    def stemm_text(text):
4        text = ' '.join(stemmer.stem(word) for word in text.split(' '))
5        return text
```

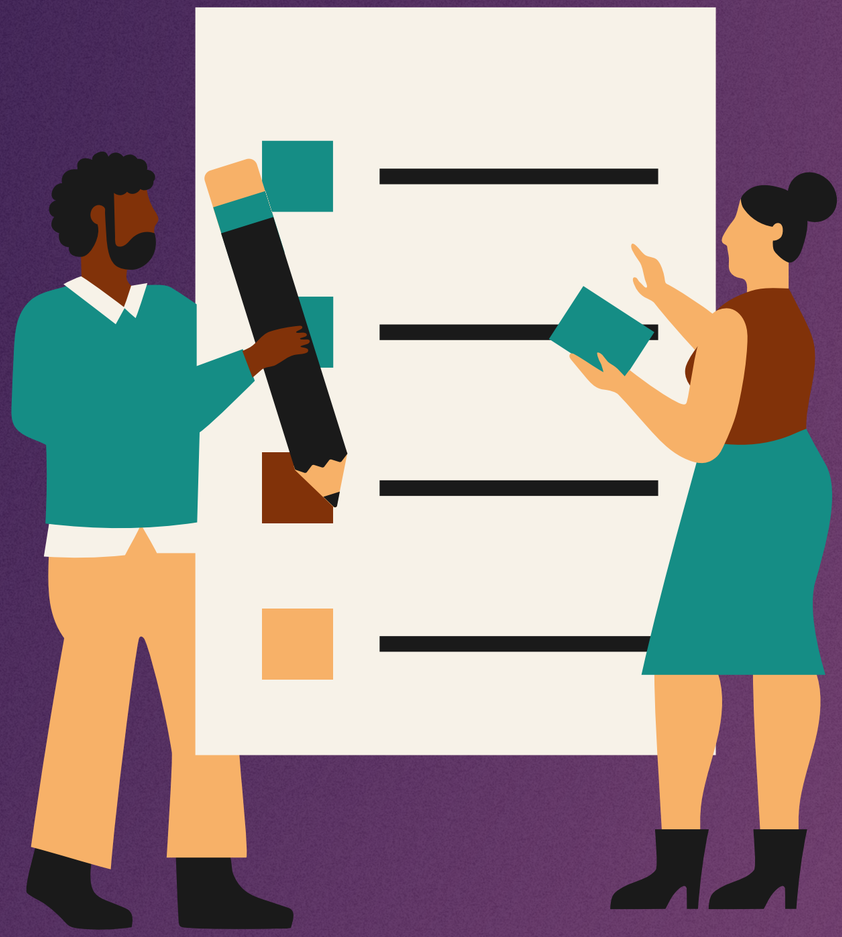


Results of stopwords and stemmer

BEFORE	AFTER
go until jurong point crazy available only in ...	go jurong point crazy available bugis great wo...
ok lar joking wif u oni	ok lar joking wif oni
free entry in 2 a wkly comp to win fa cup fina...	free entry 2 wkly comp win fa cup final tkts 2...
u dun say so early hor u c already then say	dun say early hor already say
nah i dont think he goes to usf he lives aroun...	nah dont think goes usf lives around though

CHAPTER IV

VECTORIZATION



(ALMOST) DETECTING SPAM MESSAGES

04

Text Mining and NLP

CountVectorizer

```
#vect = CountVectorizer()
vect = CountVectorizer(lowercase=True)
```

CountVectorizer

```
#vect = CountVectorizer()
vect = CountVectorizer(lowercase=True, stop_words=stop_words)
```

CountVectorizer

```
#vect = CountVectorizer()
vect = CountVectorizer(lowercase=True, stop_words=stop_words, ngram_range=(1,2))
```

CountVectorizer

```
#vect = CountVectorizer()
vect = CountVectorizer(lowercase=True, stop_words=stop_words, ngram_range=(1,2), min_df=0.1, max_df=0.7)
```

CountVectorizer

```
#vect = CountVectorizer()
vect = CountVectorizer(lowercase=True, stop_words=stop_words, ngram_range=(1,2), min_df=0.1, max_df=0.7, max_features=100)
```

CountVectorizer

```
#vect = CountVectorizer()
vect = CountVectorizer(lowercase=True, stop_words=stop_words, ngram_range=(1,2), min_df=0.1, max_df=0.7, max_features=100)
vect.fit(x_train)

x_train_dtm = vect.transform(x_train)
x_test_dtm = vect.transform(x_test)
```

TF-IDF

$$w_{i,j} = tf_{i,j} \cdot \log\left(\frac{N}{df_i}\right)$$

```
tfidf_transformer = TfidfTransformer()

tfidf_transformer.fit(x_train_dtm)
x_train_tfidf = tfidf_transformer.transform(x_train_dtm)

x_train_tfidf
```

How relevant is a word in a message

Count the number of occurrences in a document (x_train_dtm).

Inverse Document Frequency

Offset by the number of documents in the corpus that contain the word

Why use it?

Is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus

TF-IDF

$$w_{i,j} = tf_{i,j} \cdot \log\left(\frac{N}{df_i}\right)$$

```
tfidf_transformer = TfidfTransformer()

tfidf_transformer.fit(x_train_dtm)
x_train_tfidf = tfidf_transformer.transform(x_train_dtm)

x_train_tfidf
```

How relevant is a word in a message

Count the number of occurrences in a document (x_train_dtm).

Inverse Document Frequency

Offset by the number of documents in the corpus that contain the word

Why use it?

Is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus

TF-IDF

$$w_{i,j} = tf_{i,j} \cdot \log\left(\frac{N}{df_i}\right)$$

```
tfidf_transformer = TfidfTransformer()

tfidf_transformer.fit(x_train_dtm)
x_train_tfidf = tfidf_transformer.transform(x_train_dtm)

x_train_tfidf
```

How relevant is a word in a message

Count the number of occurrences in a document (x_train_dtm).

Inverse Document Frequency

Offset by the number of documents in the corpus that contain the word

Why use it?

It's a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus

CHAPTER V

EMBEDDINGS

(ALMOST) DETECTING SPAM MESSAGES

05

Text Mining and NLP

CHOOSE YOUR FIGHTER

CHOOSE YOUR FIGHTER

BPE

CHOOSE YOUR FIGHTER

A BRIEF HISTORY OF OUR BPE

How we did it

DATASET

NUS SMS Corpus Tao Chen and Min-Yen Kan (2013).

CONVERT TO TXT

Convert to txt.

PREPROCESSING

Remove duplicates and inconsistent rows. Keep only the SMS.

CLEANING

Same strategy of spam.csv.

TRAINING

Create BPE of NUS SMS Corpus. Shahad Mahmud (2023).

EMBEDDING

Use torch.randn().

APPLY

Apply the embedding on spam.csv, and create a txt.

GLOVE

Global Vectors for Word Representation

glove.6B.100d.txt (347.12 MB)

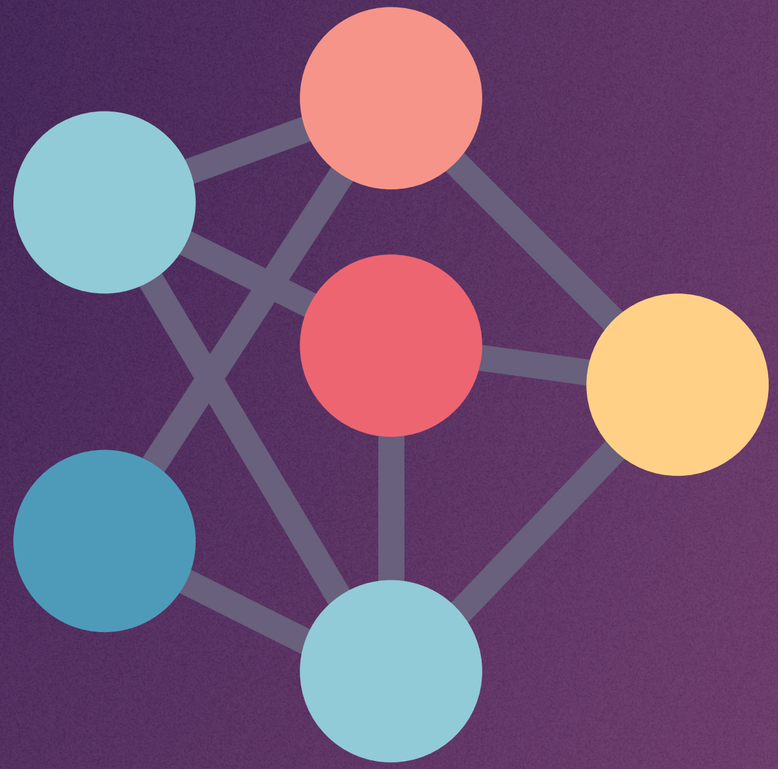
This preview is truncated due to the large file size. Create a Notebook or download this file to see the full content.

Download **Create Notebook**

```
the -0.038194 -0.24487 0.72812 -0.39961 0.083172 0.043953 -0.39141 0.3344 -0.57545 0.087459 0.28787 -0.06731 0.30906 -0.26384 -0.13231 -0.20757 0.33395 -0.33848 -0.31743 -0.48336 0.1464 -0.37304 0.34577 0.052041 0.44946 -0
, -0.10767 0.11053 0.59812 -0.54361 0.67396 0.10663 0.038867 0.35481 0.06351 -0.094189 0.15786 -0.81665 0.14172 0.21939 0.58505 -0.52158 0.22783 -0.16642 -0.68228 0.3587 0.42568 0.19021 0.91963 0.57555 0.46185 0.42363 -0.0
. -0.33979 0.20941 0.46348 -0.64792 -0.38377 0.038034 0.17127 0.15978 0.46619 -0.019169 0.41479 -0.34349 0.26872 0.04464 0.42131 -0.41032 0.15459 0.022239 -0.64653 0.25256 0.043136 -0.19445 0.46516 0.45651 0.68588 0.091295
of -0.1529 -0.24279 0.89837 0.16996 0.53516 0.48784 -0.58826 -0.17982 -1.3581 0.42541 0.15377 0.24215 0.13474 0.41193 0.67043 -0.56418 0.42985 -0.012183 -0.11677 0.31781 0.054177 -0.054273 0.35516 -0.30241 0.31434 -0.33846
to -0.1897 0.050024 0.19084 -0.049184 -0.089737 0.21006 -0.54952 0.098377 -0.20135 0.34241 -0.092677 0.161 -0.13268 -0.2816 0.18737 -0.42959 0.96039 0.13972 -1.0781 0.40518 0.50539 -0.55064 0.4844 0.38044 -0.0029055 -0.349
and -0.071953 0.23127 0.023731 -0.50638 0.33923 0.1959 -0.32943 0.18364 -0.18057 0.28963 0.20448 -0.5496 0.27399 0.58327 0.20468 -0.49228 0.19974 -0.070237 -0.88049 0.29485 0.14071 -0.1009 0.99449 0.36973 0.44554 0.28998 -
in 0.085703 -0.22201 0.16569 0.13373 0.38239 0.35401 0.01287 0.22461 -0.43817 0.50164 -0.35874 -0.34983 0.055156 0.69648 -0.17958 0.067926 0.39101 0.16039 -0.26635 -0.21138 0.53698 0.49379 0.9366 0.66902 0.21793 -0.46642 0
a -0.27086 0.044006 -0.02026 -0.17395 0.6444 0.71213 0.3551 0.47138 -0.29637 0.54427 -0.72294 -0.0047612 0.040611 0.043236 0.29729 0.10725 0.40156 -0.53662 0.033382 0.067396 0.64556 -0.085523 0.14103 0.094539 0.74947 -0.19
" -0.30457 -0.23645 0.17576 -0.72854 -0.28343 -0.2564 0.26587 0.025309 -0.074775 -0.3766 -0.057774 0.12159 0.34384 0.41928 -0.23236 -0.31547 0.60939 0.25117 -0.68667 0.70873 1.2162 -0.1824 -0.48442 -0.33445 0.30343 1.086 0
's 0.58854 -0.2025 0.73479 -0.68338 -0.19675 -0.1802 -0.39177 0.34172 -0.60561 0.63816 -0.26695 0.36486 -0.40379 -0.1134 -0.58718 0.2838 0.8025 -0.35303 0.30083 0.078935 0.44416 -0.45906 0.79294 0.50365 0.32805 0.28027 -0.
for -0.14401 0.32554 0.14257 -0.099227 0.72536 0.19321 -0.24188 0.20223 -0.89599 0.15215 0.035963 -0.59513 -0.051635 -0.014428 0.35475 -0.31859 0.76984 -0.087369 -0.24762 0.65059 -0.15138 -0.42703 0.18813 0.091562 0.15192
- -1.2557 0.61036 0.56793 -0.96596 -0.45249 -0.071696 0.57122 -0.31292 -0.43814 0.90622 0.06961 -0.053104 0.25029 0.27841 0.77724 0.26329 0.56874 -1.1171 -0.078268 -0.51317 0.8071 0.99214 0.22753 1.0847 0.88292 0.17221 -0.
that -0.093337 0.19043 0.68457 -0.41548 -0.22777 -0.11803 -0.095434 0.19613 0.17785 -0.020244 -0.055409 0.33867 0.79396 -0.047126 0.44281 -0.061266 0.20796 0.034094 -0.64751 0.35874 0.13936 -0.6831 0.25596 -0.12911 0.2608
on -0.21863 -0.42664 0.5196 0.0043103 0.58045 -0.10873 -0.37726 0.4566 -0.60627 -0.075773 0.11306 0.17703 0.1605 0.074514 0.63649 -0.078852 0.75268 -0.24962 -0.51628 -0.33348 0.66754 -0.34183 0.61316 0.31668 0.64846 -0.079
is -0.54264 0.41476 1.0322 -0.40244 0.46691 0.21816 -0.074864 0.47332 0.080996 -0.22079 -0.12808 -0.1144 0.50891 0.11568 0.028211 -0.3628 0.43823 0.047511 0.20282 0.49857 -0.10068 0.13269 0.16972 0.11653 0.31355 0.25713 0.
was 0.13717 -0.54287 0.19419 -0.29953 0.17545 0.084672 0.67752 0.098295 -0.035611 0.21334 0.51663 0.20687 0.44082 -0.33655 0.56025 -0.6879 0.51957 -0.21258 -0.52708 -0.12249 0.33099 0.026448 0.59007 0.0065469 0.45405 -0.33
said -0.13128 -0.452 0.043399 -0.99798 -0.21053 -0.95868 -0.24609 0.48413 0.18178 0.475 -0.22305 0.30064 0.43496 -0.3605 0.20245 -0.52594 -0.34708 0.0075873 -1.0497 0.18673 0.57369 0.43814 0.098659 0.3877 -0.2258 0.41911 0
with -0.43608 0.39104 0.51657 -0.13861 0.2029 0.50723 -0.012544 0.22948 -0.6316 0.21199 -0.018043 -0.39364 0.74164 0.30221 0.51792 -0.25191 0.25373 -0.65184 -0.42963 0.0093622 0.023334 -0.39245 0.34948 0.21217 0.7346 -0.21
he 0.1225 -0.058833 0.23658 -0.28877 -0.028181 0.31524 0.070229 0.16447 -0.027623 0.25214 0.21174 -0.059674 0.36133 0.13607 0.18755 -0.1487 0.31315 0.13368 -0.59703 -0.030161 0.080656 0.26162 -0.055924 -0.35351 0.34722 -0.
as -0.32721 0.096446 0.34244 -0.44327 0.30535 -0.042016 -0.071235 -0.31036 -0.22557 -0.181 -0.29088 -0.61542 0.29751 0.030491 0.41504 -0.51489 0.68628 -0.020302 -0.18486 0.31605 0.59472 -0.2147 0.29256 0.43262 0.35466 -0.2
it -0.30664 0.16821 0.98511 -0.33606 -0.2416 0.16186 -0.053496 0.4301 0.57342 -0.071569 0.36101 0.26729 0.27789 -0.072268 0.13838 -0.26714 0.12999 0.22949 -0.18311 0.50163 0.44921 -0.020821 0.42642 -0.068762 0.40337 0.0951
by -0.20875 -0.1174 0.26478 -0.28339 0.19584 0.7446 -0.03887 0.028499 -0.44252 -0.30426 0.27133 -0.51907 0.52183 -0.76648 0.28043 -0.48344 -0.15626 -0.49705 -0.51024 -0.03652 0.20579 -0.6136 0.46388 0.73497 0.66813 -0.4443
at 0.1766 0.093851 0.24351 0.44313 -0.39037 0.12524 -0.19918 0.59855 -0.82035 0.28006 0.54231 0.023079 0.12837 -0.044489 0.3837 -0.75659 0.40254 -0.4462 -0.81599 -0.0091513 0.65219 -0.043656 0.54919 -0.16696 0.73028 -0.207
( 0.19247 0.36617 0.52301 -0.79857 -0.2592 0.18267 0.19564 0.83148 -0.67636 -0.84648 1.4429 -0.84978 -0.023986 1.328 0.74061 0.039546 0.61659 -0.075604 -0.59537 0.69163 0.71303 0.016798 0.57518 0.94396 0.38447 -0.095771 0.
) -0.13797 0.27084 0.84036 -0.45668 -0.49429 0.35777 0.077772 0.42481 0.0076481 -0.50942 1.4008 -0.79993 0.053011 1.2054 0.3783 0.0842 0.91317 -0.35173 -0.30452 0.65606 0.50428 0.074796 0.31149 0.81957 0.40216 0.10982 0.3
from 0.30731 0.24737 0.68231 -0.52367 0.44053 0.42044 0.0002514 0.15265 -0.61363 0.22631 0.083071 0.070425 0.017683 0.56807 1.0067 -0.46206 0.44524 -0.50984 -0.42985 0.19935 0.22729 0.51662 0.56282 0.41282 0.17742 -0.1569
his 0.12883 -0.82209 0.27438 -0.069014 0.17989 0.72605 -0.15112 0.0085541 -0.95122 0.77243 -0.28375 0.28329 0.14825 -0.01223 -0.019267 -0.03446 0.31506 -0.16639 -0.013435 -0.0020459 0.064905 -0.20989 0.12524 0.3523 0.6404
'' 0.16478 0.17071 0.62111 -1.2101 -0.84063 0.21893 0.48123 -0.15044 0.36701 -0.20857 -0.23385 0.019356 -0.045098 0.18001 0.11995 -0.25622 -0.026299 0.28473 -0.91322 0.59811 0.30248 0.27973 0.11444 -0.073628 0.88137 1.0633
`` 0.092672 0.20241 0.69394 -0.50775 -0.097297 0.045522 -0.14156 0.30736 -0.35448 -0.20612 -0.21092 -0.0026685 -0.11537 0.052913 0.02908 0.0067036 0.47268 0.44669 -0.35419 0.70959 0.6984 0.42713 -0.40276 -0.37443 0.7434 0.
an -0.4214 -0.18797 0.46241 -0.17605 0.36212 0.36701 0.27924 0.14634 -0.054227 0.45834 0.065416 -0.33725 0.067505 -0.36316 0.50302 -0.010361 0.72826 -0.17564 -0.33996 0.072864 0.64481 -0.23908 0.38383 0.13858 1.0994 -0.248
be -0.46953 0.38432 0.54833 -0.63401 0.010133 0.11364 0.10612 0.58529 0.032302 -0.12274 0.030265 0.52662 1.0398 -0.082143 0.19118 -0.83784 0.50763 0.44488 -0.72604 0.036893 0.24211 -0.28878 0.33657 0.13656 0.14579 -0.13221
has 0.093736 0.56152 0.48364 -0.45987 0.56067 -0.1694 0.018687 0.45529 0.065615 0.25181 -0.14251 0.10532 0.77865 0.1428 -0.08114 -0.069555 0.32433 0.019611 -0.15608 0.22235 0.35559 0.14713 0.19156 0.2803 0.27691 -0.2067 -0
are -0.51533 0.83186 0.22457 -0.73865 0.18718 0.26021 -0.42564 0.67121 -0.31084 -0.61275 0.089526 -0.24011 1.1878 0.67609 -0.022885 -0.92533 0.071174 0.38837 -0.42924 0.37144 0.32671 0.43141 0.87495 0.34009 -0.23189 -0.411
have 0.15711 0.65606 0.0021149 -0.65144 -0.28427 -0.20369 -0.077596 0.40798 -0.03447 -0.1639 -0.21597 0.34178 1.196 0.33639 -0.21076 -0.56015 0.1507 0.34912 -0.97128 0.18152 0.74408 0.20029 0.66986 0.16085 -0.0013507 -0.55
but -0.057078 0.39874 0.68861 -0.68151 -0.45583 0.2008 0.17974 0.053648 0.43762 -0.026725 0.13383 -0.0078137 0.42207 -0.31801 0.18065 -0.35387 -0.30929 0.04066 -0.48854 0.3791 0.47955 -0.041942 0.40894 0.12419 0.40096 0.19
```

CHAPTER VI

MODELS



(ALMOST) DETECTING SPAM MESSAGES

06

Text Mining and NLP

MODELS

01 - XGBoost

Training a number of decision trees. Each tree is trained on a subset of the data.

02 - LSTM

Subtype of Recurrent Neural Networks (RNN). Used to recognize patterns in data sequences.

03 - BERT

Use self-attention mechanisms to capture long-range dependencies and contextual relationships between words.

Next Page

01 - XGBoost

```
pipe = Pipeline([
    ('bow', CountVectorizer()),
    #('bow', CountVectorizer(lowercase=True, stop_words=stop_words,
        ngram_range=(1,9), min_df=0.05, max_df=0.95, max_features=750)),
    ('tfid', TfidfTransformer()),
    ('model', xgb.XGBClassifier(
        learning_rate=0.1,
        max_depth=11,
        n_estimators=80,
        use_label_encoder=False,
        eval_metric='auc',
        # colsample_bytree=0.8,
        # subsample=0.7,
        # min_child_weight=5,
    ))
])

pipe.fit(x_train, y_train)

y_pred_class_xgb = pipe.predict(x_test)
y_pred_train_xgb = pipe.predict(x_train)
```

Next Page

MODELS

01 - XGBoost

Training a number of decision trees. Each tree is trained on a subset of the data.

02 - LSTM

Subtype of Recurrent Neural Networks (RNN). Used to recognize patterns in data sequences.

03 - BERT

Use self-attention mechanisms to capture long-range dependencies and contextual relationships between words.

Next Page

02 - LSTM

```python
def glove_lstm():
    model = Sequential()

    model.add(Embedding(
        input_dim=embedding_matrix.shape[0],
        output_dim=embedding_matrix.shape[1],
        weights = [embedding_matrix],
        input_length=length_long_sentence
    ))

    model.add(Bidirectional(LSTM(
        length_long_sentence,
        return_sequences = True,
        recurrent_dropout=0.2
    )))

    model.add(GlobalMaxPool1D())
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(length_long_sentence, activation = "relu"
        ))
    model.add(Dropout(0.5))
    model.add(Dense(length_long_sentence, activation = "relu"
        ))
    model.add(Dropout(0.5))
    model.add(Dense(1, activation = 'sigmoid'))
    model.compile(optimizer='rmsprop', loss
        ='binary_crossentropy', metrics=['accuracy'])

    return model

model = glove_lstm()
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 80, 5)             40045

bidirectional (Bidirectional (None, 80, 160)           55040

global_max_pooling1d (Globa  (None, 160)               0
lMaxPooling1D)

batch_normalization (BatchN  (None, 160)               640
ormalization)

dropout (Dropout)            (None, 160)               0

dense (Dense)                (None, 80)                12880

dropout_1 (Dropout)          (None, 80)                0

dense_1 (Dense)              (None, 80)                6480

dropout_2 (Dropout)          (None, 80)                0

dense_2 (Dense)              (None, 1)                 81

=================================================================
Total params: 115,166
Trainable params: 114,846
Non-trainable params: 320
_________________________________________________________________
```

Next Page

MODELS

01 - XGBoost

Training a number of decision trees. Each tree is trained on a subset of the data.

02 - LSTM

Subtype of Recurrent Neural Networks (RNN). Used to recognize patterns in data sequences.

03 - BERT

Use self-attention mechanisms to capture long-range dependencies and contextual relationships between words.

Next Page

03 - BERT

```python
tokenizer = BertTokenizer.from_pretrained('bert-large-uncased')
#tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

def bert_encode(data, maximum_length) :
    input_ids = []
    attention_masks = []

    for text in data:
        encoded = tokenizer.encode_plus(
            text,
            add_special_tokens=True,
            max_length=maximum_length,
            pad_to_max_length=True,

            return_attention_mask=True,
        )
        input_ids.append(encoded['input_ids'])
        attention_masks.append(encoded['attention_mask'])

    return np.array(input_ids),np.array(attention_masks)

texts_train_bert, texts_test_bert, target_train_bert, target_test_bert = train_test_split
    (df['message_clean'], df['target_encoded'], test_size=0.2, stratify = y, random_state=42)
```

Next Page

CHAPTER VII

RESULTS

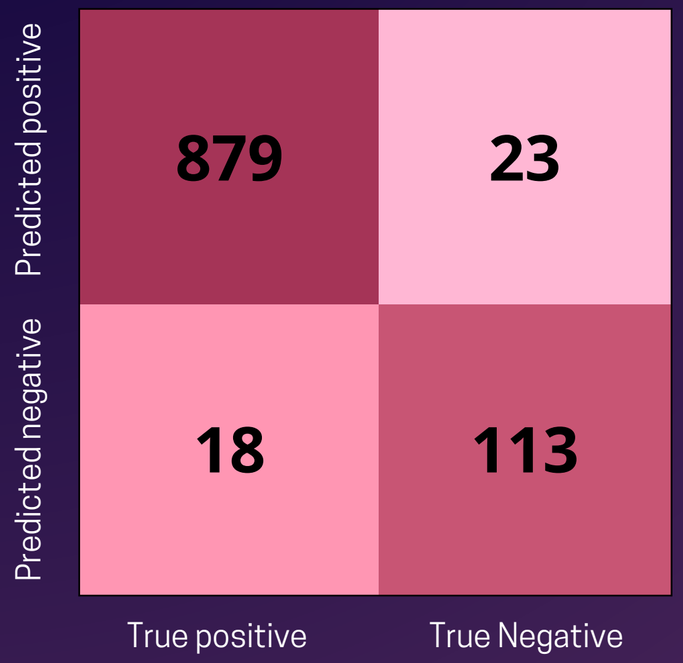
(ALMOST) DETECTING SPAM MESSAGES

07

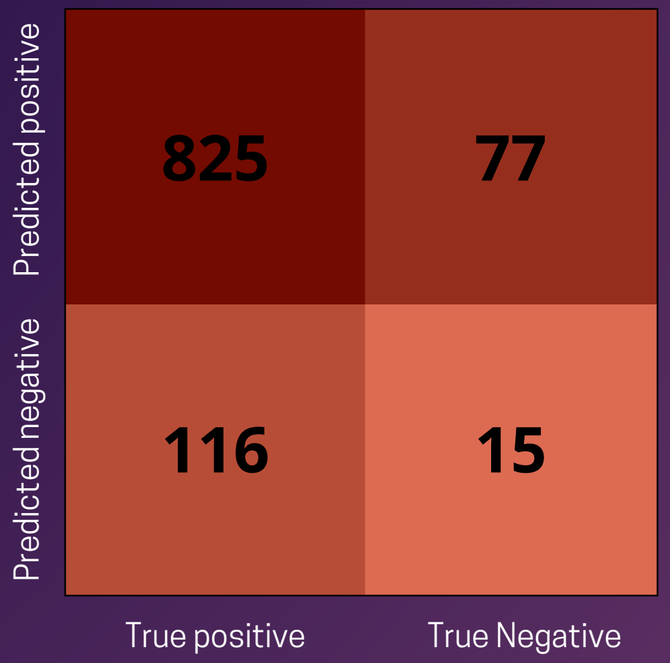
Text Mining and NLP

WITH BPE

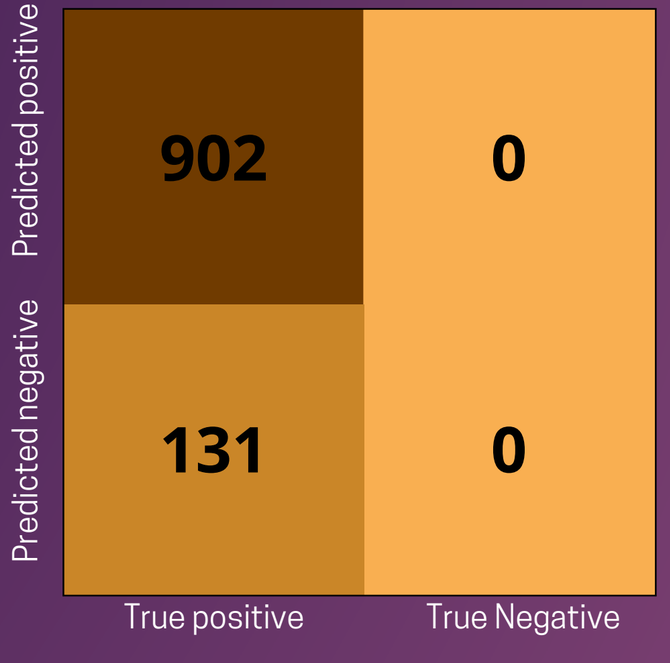
01 - XGBoost



02 - LSTM

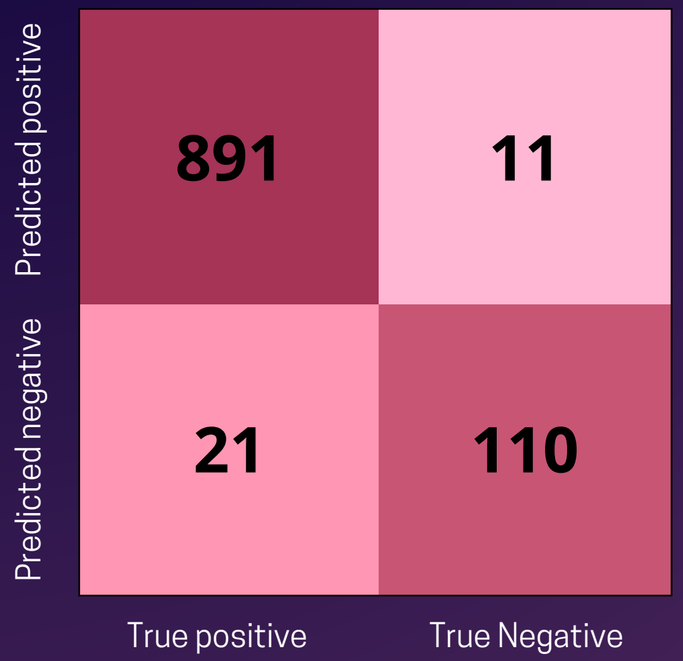


03 - BERT

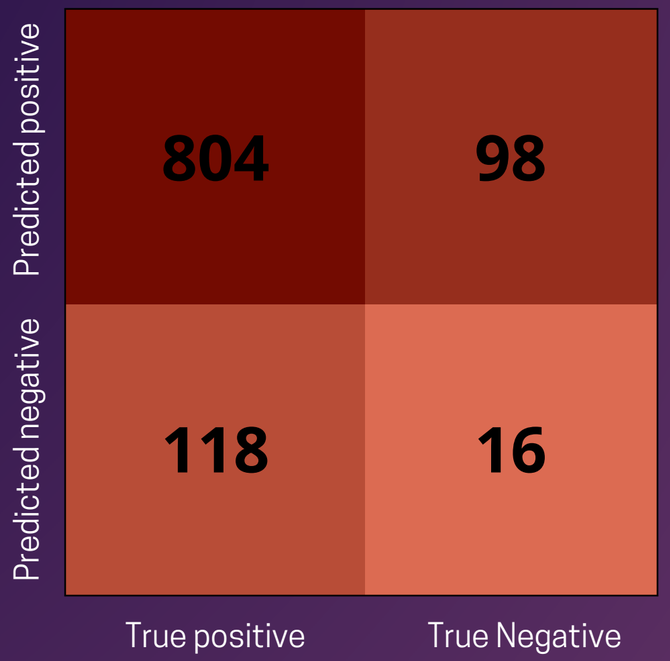


WITH GLOVE

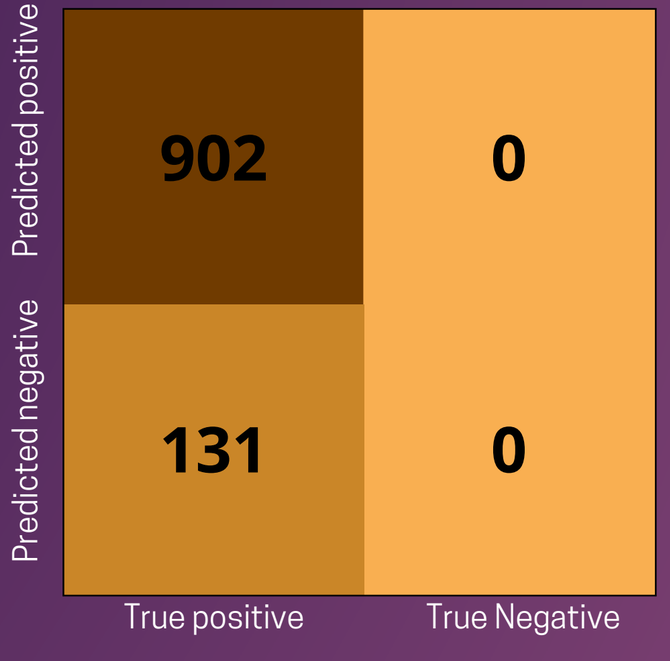
01 - XGBoost



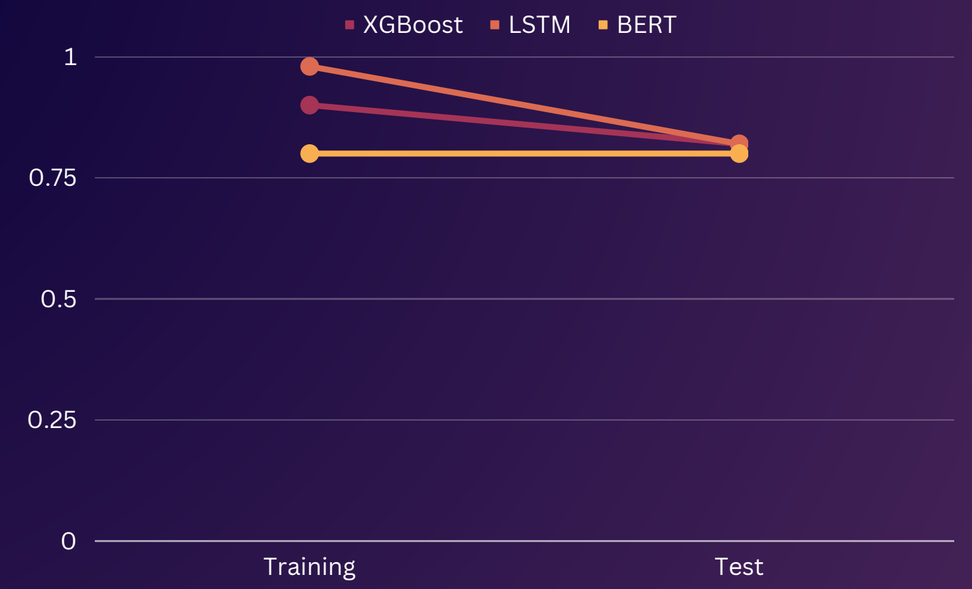
02 - LSTM



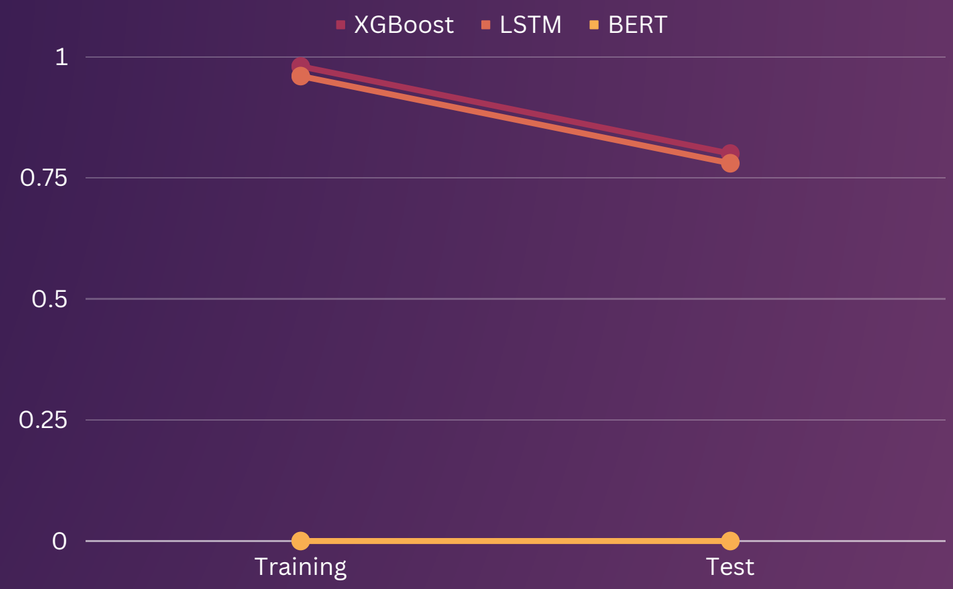
03 - BERT



ACCURACY

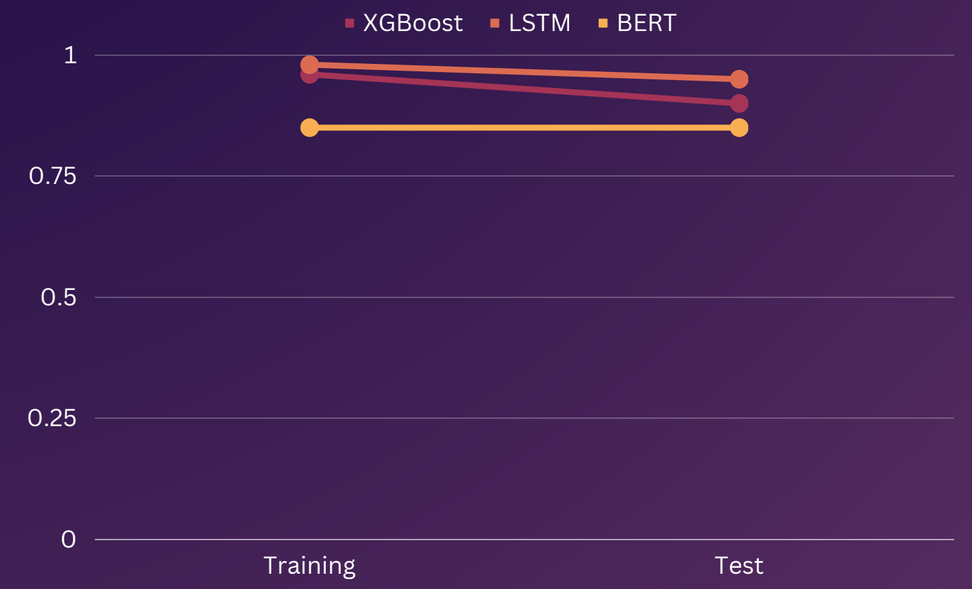


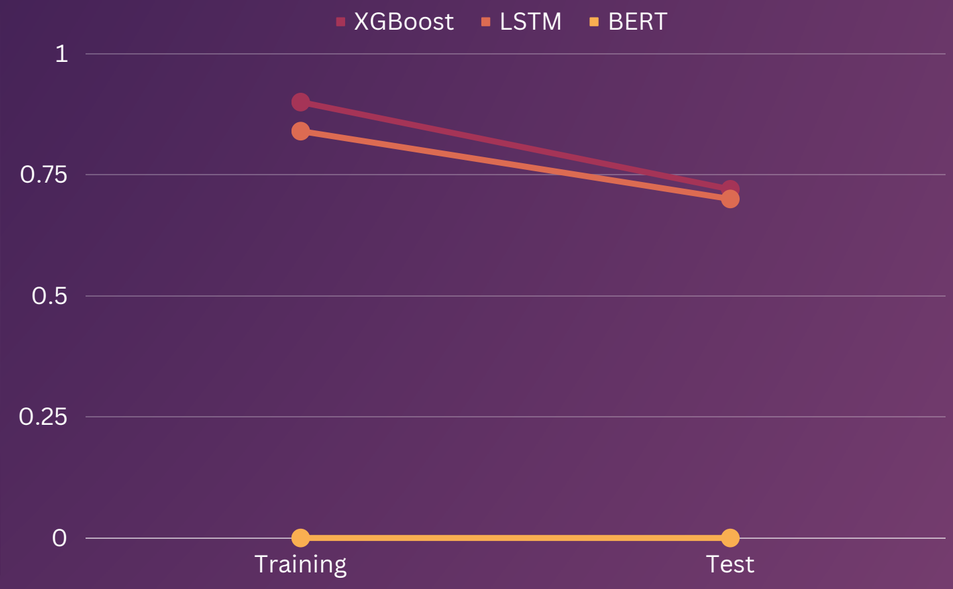
F1-SCORE



BPE

GLOVE





CHAPTER VIII

IMPROVEMENTS

(ALMOST) DETECTING SPAM MESSAGES

08

Text Mining and NLP

What we could have done better

There is always room for improvement

Better embedding	BPE + torch.randn might not be the best solution
RandomOverSampler	Consider other alternatives too
Hyperparamters	Perform a gridsearch

CHAPTER IX

THE END

```
print("The end")
```

(ALMOST) DETECTING SPAM MESSAGES

09

Sources

Kaggle SMS spam dataset

https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

NUS SMS Corpus

https://github.com/kite1988/nus-sms-corpus

BPE

https://github.com/shahad-mahmud/bpe

Kaggle

https://www.kaggle.com/code/mariapushkareva/nlp-disaster-tweets-with-glove-and-lstm/input

THANKS FOR THE ATTENTION

THANKS FOR THE ATTENTION

PLS CLAP