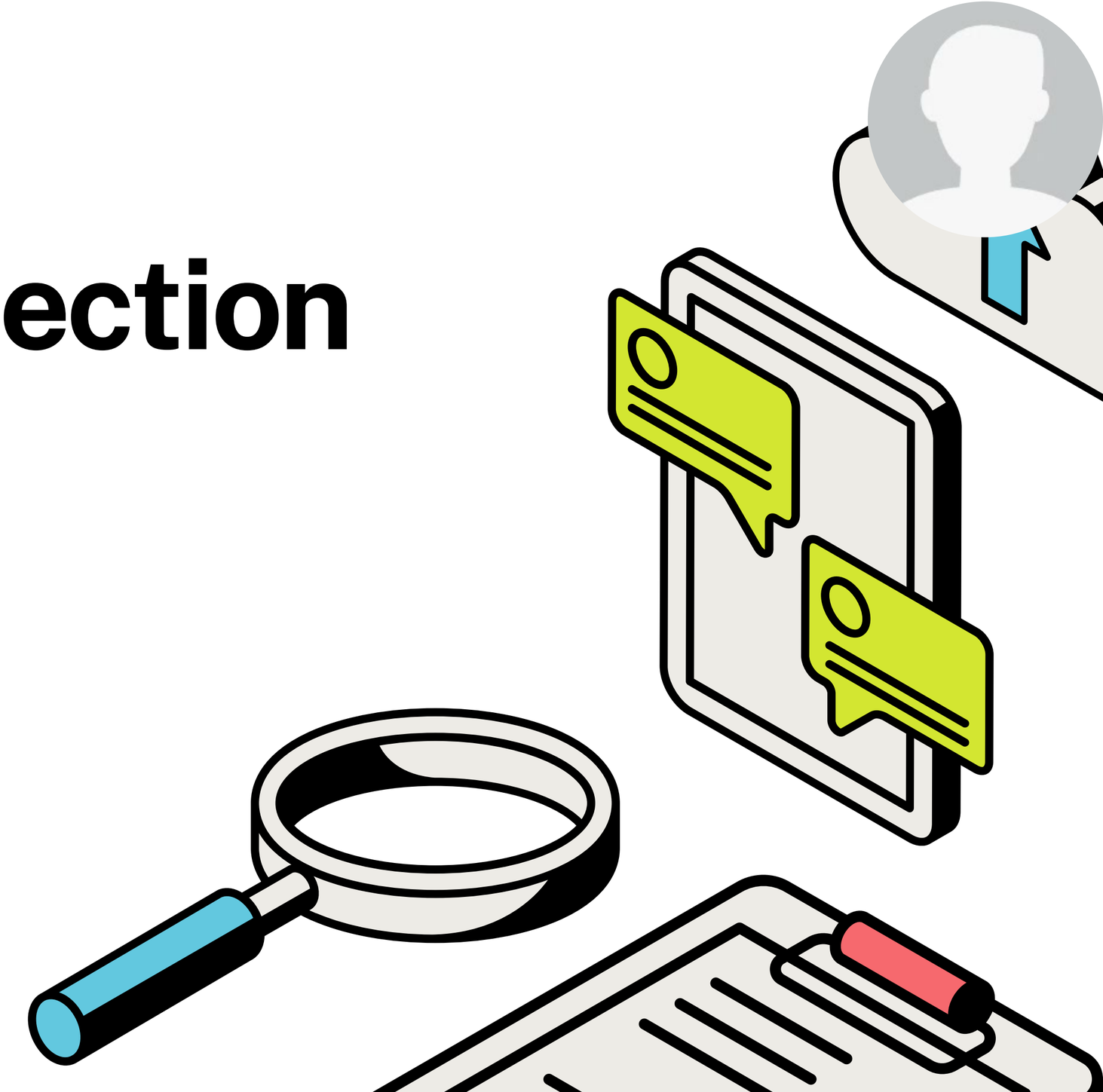
CS3244 MACHINE LEARNING

Sarcasm Detection



PG 16

Nguyen Huy Dat, A0258929H

Jeong Youngkyu, A0252154M

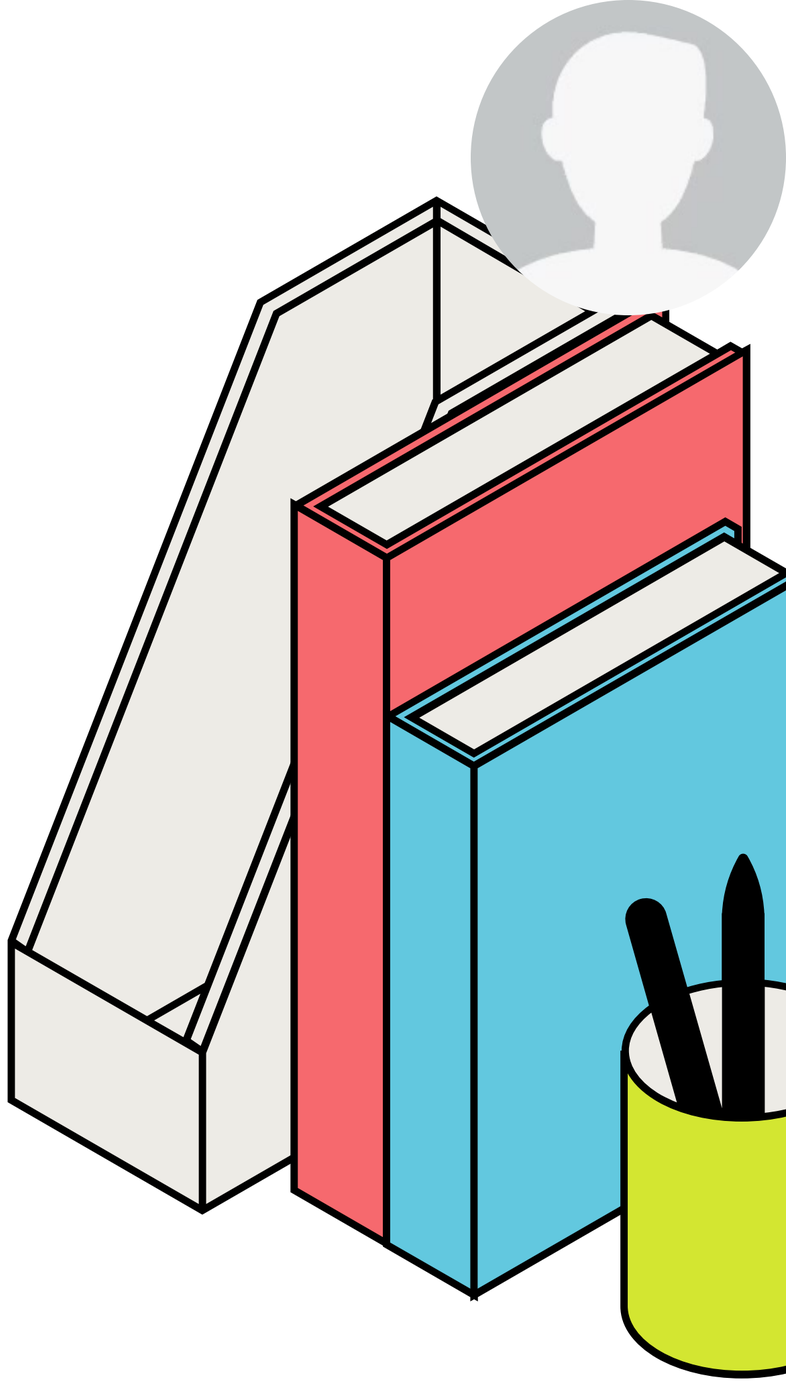
Odele Pang Kun Ting, A0245935W

Ethan Yeo Alsagoff, A0240231B

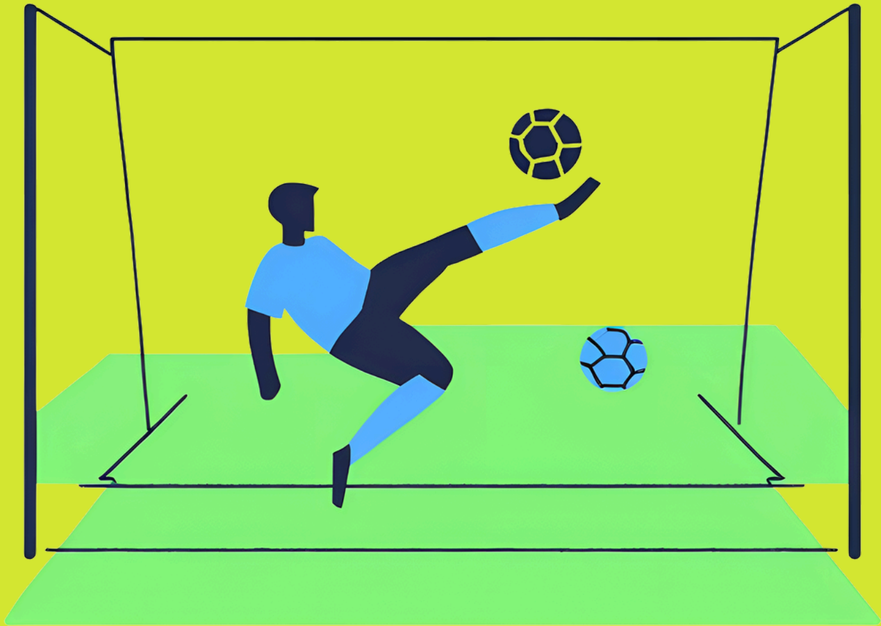
Low Mei Lin, A0240908E

Yap Yi Pin, A0258069R

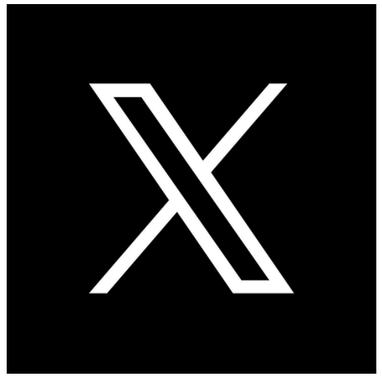
Contents



Project Goal



Sarcasm is everywhere



Poe's law is an adage of Internet culture which says that, without a clear indicator of the author's intent, any parodic or sarcastic expression of extreme views can be mistaken by some readers for a sincere expression of those views.[1][2][3]

Oops! China's Communist Party paper hails NKorea's Kim being named The Onion's 'sexiest man'

The Associated Press

Tue, November 27, 2012 at 10:00 PM GMT+8

Misunderstanding sarcasm can cause major issues!

Ex-Fifa vice president Jack Warner swallows Onion spoof

Football executive uses story from satirical website as basis for defending Fifa against US

Project Goal

Detect the use of sarcasm online

Focused on online discourse

Centered around world news and politics

Gauge public sentiment accurately

Dataset

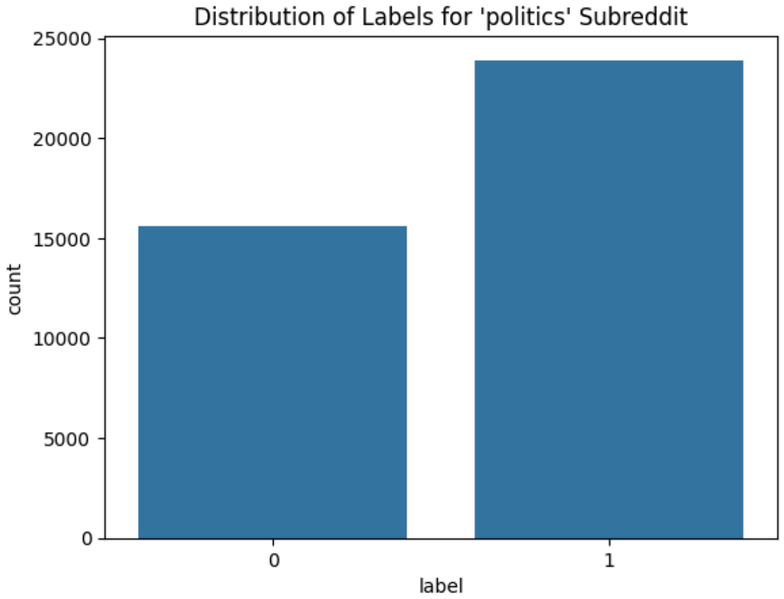
Sarcasm on reddit

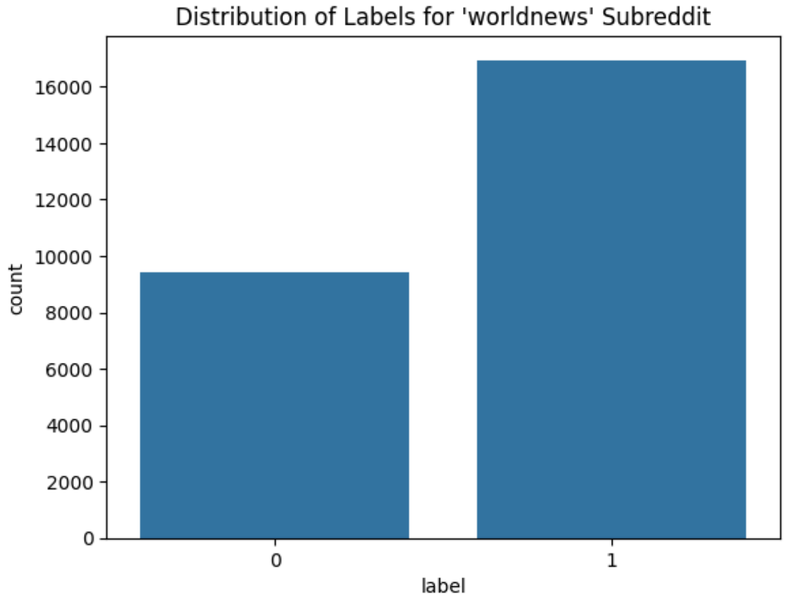
The dataset contains 1.3 million comments from Reddit, a forum social network.

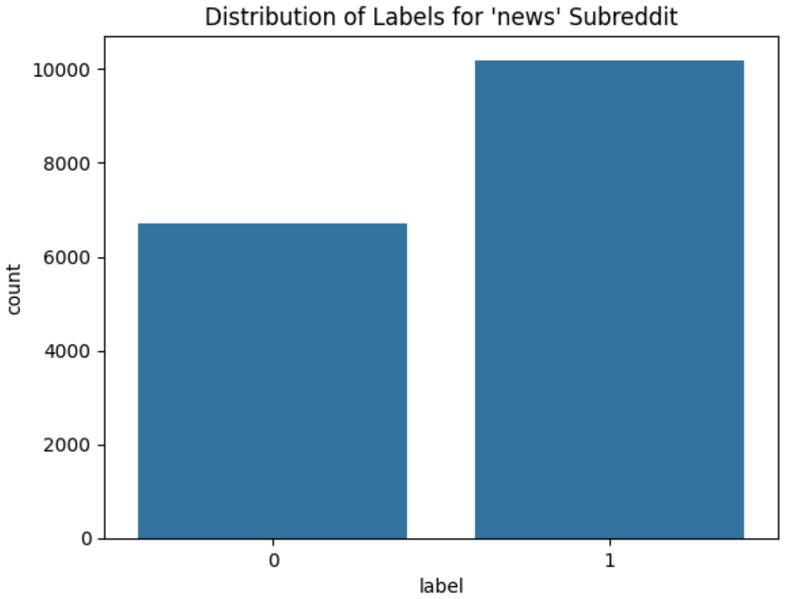
LABEL	Comments containing the “\s” tag are labelled as sarcastic
COMMENT	The comment being referred to
AUTHOR	The username of the commenter
SUBREDDIT	The subcommunity the comment was collected from
SCORE	The net value of upvotes minusdownvotes

UPS	The number of upvotes the comment received
DOWNS	The number of downvotes the comment received
DATE	The date the comment was posted
CREATED_UTC	The date and time the comment was posted, recorded in Coordinated Universal Time (UTC) format
PARENT_COMMENT	Another comment that the comment is responding to

More Sarcastic comments than non-sarcastic







Data Preparation

Feature Engineering

Word Count

Calculation of the number of words in each comment as the length of the comments may show the different sarcasm pattern.

e.g.
"Oh, great, because waiting in traffic for an hour is my favourite way to relax." ⟶ 15 words

Capital Count

Count of the number of characters in uppercase as sarcastic comments may have many uppercases for emphasis.

e.g.
“OF COURSE, BECAUSE THIS WAS UNDOUBTEDLY THE BEST USE OF MY TIME.”
⟶ 51 characters

Punctuation Count

Count of the number of punctuations as excessive use of punctuations may suggest sarcasm.

e.g.
"Sure!!! Because passing yet another committee to review the committee's review will surely solve everything...!!!"
⟶ 9 punctuations

Feature Engineering

Polarity of comments and parent comments

Measurement of the sentiment in the comment, ranging from -1 (negative) to 1 (positive).

e.g.
"Wow, what a brilliant idea to raise taxes even higher!"
→ “brilliant” has a high polarity score (near 1)

Subjectivity of comments and parent comments

Measurement of how subjective or objective a text is, ranging from 0 (objective) to 1 (subjective).

e.g.
“I just love how politicians always have our best interests at heart”
→ “love” has a high subjectivity score (near 1)

Text Preprocessing

Lowercasing → **Expansion of Contractions** → **Removal of stopwords**

Conversion of all the words to lowercase tox remove case sensitivity

e.g.
"Hello" ⟶ "hello"

"NO WAY!" ⟶ "no way!"

Expansion of the contractions to standardise the text

e.g.
"who's this?" ⟶ "who is this?"

"i can't believe this
⟶ "i cannot believe this"

Removal of stopwords using a predefined stopword list as they do not have any significant meaning.

e.g.
"he is a horrible president"
⟶ "horrible president"

Text Preprocessing

Removal of special characters & punctuatons

Elimination of characters like ‘@’, ‘!’, ‘%’ , etc., to focus on words rather than symbols

e.g.
“@users, thanks!”
⟶ “users thanks”

Removal of common words and rare words

Removal of these words as they do not contribute significantly to the meaning of the comments.

e.g.
top_words = { people, … , yes}
rare_words = { snack, etc.}

“yes people love eating snack”
⟶ “love eating”

Lemmatisation

Conversion of words into their dictionary form to reduce variability in words.

e.g.
“runs faster”
⟶ “run fast”

Bigram

Tokenisation of the text into bigrams to capture context and relationships between words better than single words.

e.g.
“i love trump”
⟶ “i love” , “love trump”

Embeddings

TF - IDF
(Term Frequency-Inverse Document Frequency)

It is a statistical method that transforms text into sparse numerical representations based on the importance of words relative to the corpus.

- **TF** : Measures how frequently a term occurs in a document.
- **IDF** : Measures how important a term is in the entire corpus
- Calculation: **TF-IDF** = **TF × IDF**

- **TF-IDF** can help identify sarcastic patterns by detecting rare but important words that appear more often in sarcastic contexts.
- **High TF-IDF words:** Sarcasm often relies on specific, emotionally loaded words that stand out in certain contexts.

e.g.

“Oh, great job on the presentation...”

- TF-IDF for "great" is usually high and the context suggests it’s negative sarcasm. The high TF-IDF score of "great" helps the model identify it as sarcastic.

Embeddings

BERT
(Bidirectional Encoder Representations from Transformers)

It is a pre-trained transformer model that provides dense, contextualised representations of text that capture the semantic and syntactic meaning of words in context.

- **Tokenisation:** BERT splits the text into tokens
- **Transformer Architecture:** It uses self-attention mechanisms to process the relationships between words in a sentence.

- **BERT**'s bidirectional context is useful for sarcasm detection because it can understand that words like "great" might have opposite meanings depending on surrounding words.

e.g.

“Oh, great job on the presentation...”

- **BERT** understands that the phrase "great job" has a negative sentiment based on the surrounding context as “presentation” could be associated with bad meanings.

Evaluation Metrics

Classification metrics

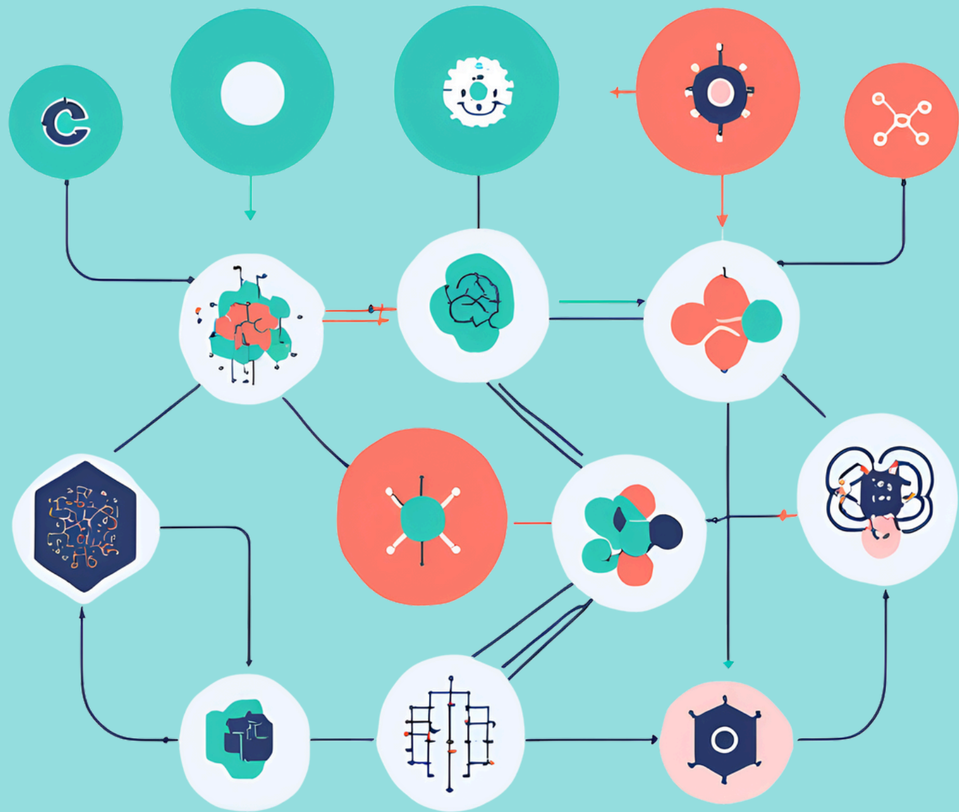
To assess how well each model classifies a comment as sarcastic/not sarcastic, we considered the following metrics:

ACCURACY	PRECISION	RECALL	F1
How often the model assigns the correct label	How many of the sarcastic comments identified are truly sarcastic	How many of the truly sarcastic comments are identified as sarcastic	Balance between precision and recall
$\frac{TP+TN}{TP+TN+FP+FN}$	$\frac{TP}{TP+FP}$	$\frac{TP}{TP+FN}$	$\frac{2TP}{2TP+FP+FN}$

Since our sarcasm detector was to be used in chatbots, we chose to **optimise for precision to minimise false positives**.

- Users may be frustrated if chatbot misinterpret words as sarcastic

Baseline Models



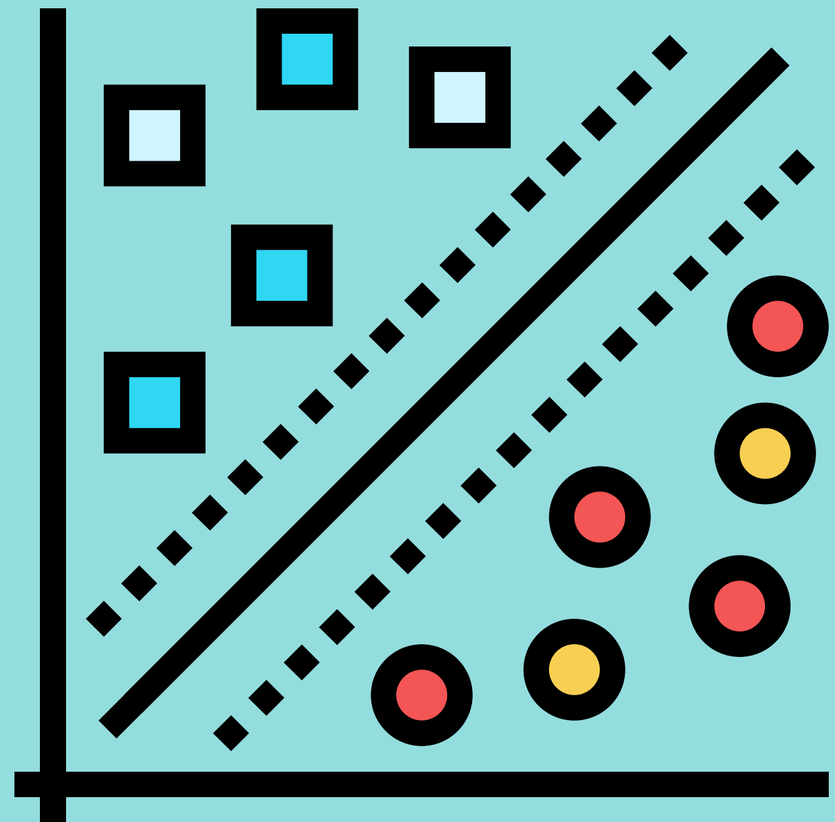
Summary of results

Our initial exploration included 6 models. 5-fold cross validation (CV) was applied to each model and their corresponding CV precision/accuracy/recall/F1 score were taken:

Model	CV Precision	CV Accuracy	CV Recall	CV F1
Support Vector Machine (SVM)	0.77	0.62	0.55	0.64
Recurrent Neural Network (RNN)	0.70	0.66	0.77	0.74
Random Forest	0.68	0.67	0.88	0.77
Logistic Regression	0.70	0.67	0.81	0.75
Decision Tree	0.65	0.59	0.79	0.72
K Nearest Neighbours	0.67	0.61	0.71	0.69

Improved Model 1

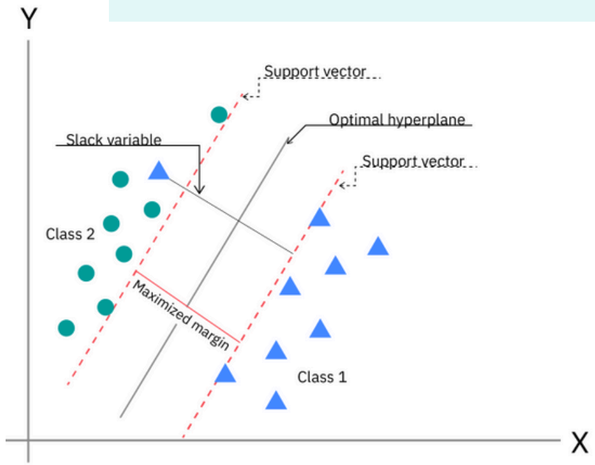
SVM



Why We Chose SVM

The Support Vector Machine (SVM) algorithm handles **linear and non-linear classification problems**. When the data is not linearly separable, SVM uses different kernel functions (**linear, polynomial, radial basis function (RBF), and sigmoid functions**) to map data into a higher-dimensional space, which makes linear separation possible. This approach is also known as "**kernel trick**", which enables the SVM to choose the correct. Choosing the optimal kernel function allows the SVM to resolve complex classification problems (IBM, 2023).

In the context of sarcasm detection, SVM could be helpful as the kernel trick allows it to capture subtle and non-linear relationships in the data, such as the ones present in **sarcasm's linguistic and contextual features**. By transforming the data into a **higher-dimensional space**, the algorithm helps identify **patterns in a text, such as cues, tone, word usage, and context shifts**, which could help **distinguish sarcastic expressions from non-sarcastic** statements.



Experimenting with Different Kernels

Features: Sparse Matrix consisting of TFIDF Matrix + Numeric Features
Model Parameters:

- C = 1.0 (the larger the cost, the slower it takes for the SVM model to converge)
- class_weight = ‘balanced’
- Kernel types varied as shown below

Summary of Results for SVM with Different Kernels (k=2 folds)

Kernel Types	CV Precision	CV Accuracy	CV Recall	CV F1
Linear	0.76	0.62	0.56	0.64
RBF	0.68	0.59	0.66	0.67
Sigmoid	0.66	0.54	0.54	0.59
Poly	0.64	0.63	0.96	0.77

The SVM model with linear kernel performs better than the other kernels in terms of precision.

Experimenting with Removal of Numeric Features

Model Parameters:

- C = 1.0
- class_weight = ‘balanced’

Summary of Results for Linear SVM (k=2 folds)

Features	CV Precision	CV Accuracy	CV Recall	CV F1
A. TFIDF Matrix Only	0.763	0.61	0.55	0.64
B. Numeric Features + TFIDF Matrix (Previous Slide)	0.764	0.62	0.56	0.64

By using only the TFIDF matrix as the input features, there is not much of a difference when compared to using both TFIDF and numeric features. **B** performs slightly better than **A** when comparing their mean precision score.

Final Linear SVM after Hyperparameters Tuning

Features: TFIDF Matrix + Numeric Features

Model Parameters:

- Kernel: Linear
- C: 0.1
- Gamma: 1
- Class weight: Balanced

	Predicted 0	Predicted 1
True 0	23497	7506
True 1	24025	26495

*Aggregated counts across 5 folds

TN	FP
FN	TP

Results for Linear SVM after Hyperparameter Tuning (k=5 folds)

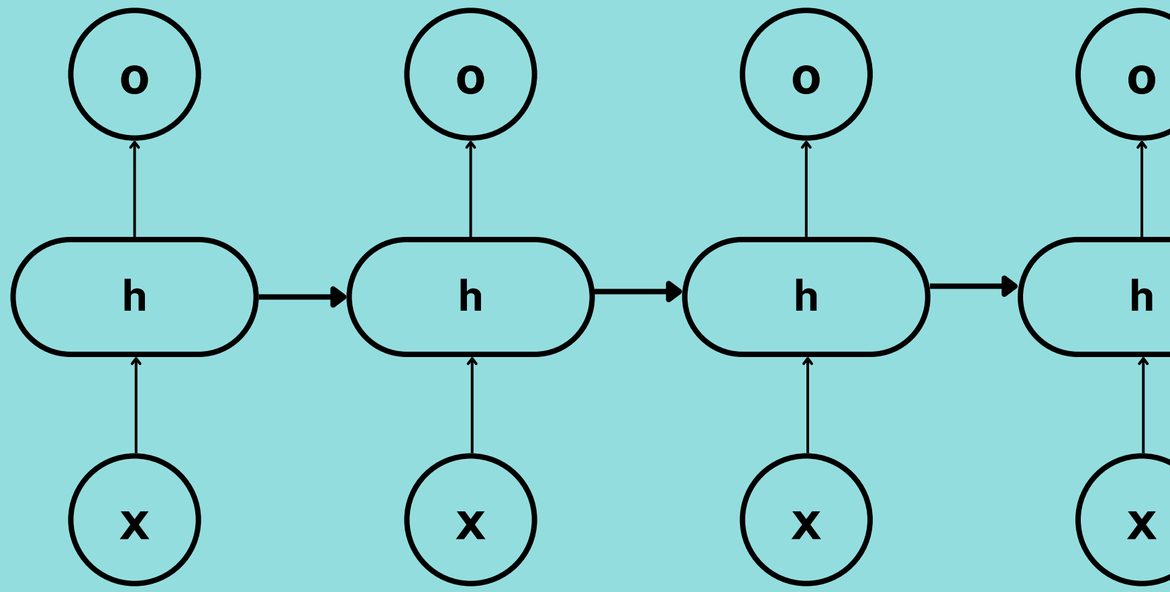
CV Precision	CV Accuracy	CV Recall	CV F1
0.78	0.61	0.52	0.63

*Mean score from 5-fold cross-validation

By tuning the hyperparameters using Grid Search, we noticed a significant increase in precision score.

Improved Model 2

RNN



Why We Chose Recurrent Neural Network?

Sequential Information Processing: Because they are built to handle sequences, RNNs work well in jobs where the order in which the input data is entered is important

Feature Learning: From the input data, RNNs automatically extract pertinent features and representations.

Experiments with Embedding

TF-IDF: Each comment -> (1, 13804) vector
BERT: Each comment -> (15, 768) matrix with padding. 15 is the 75th percentile of the number of words

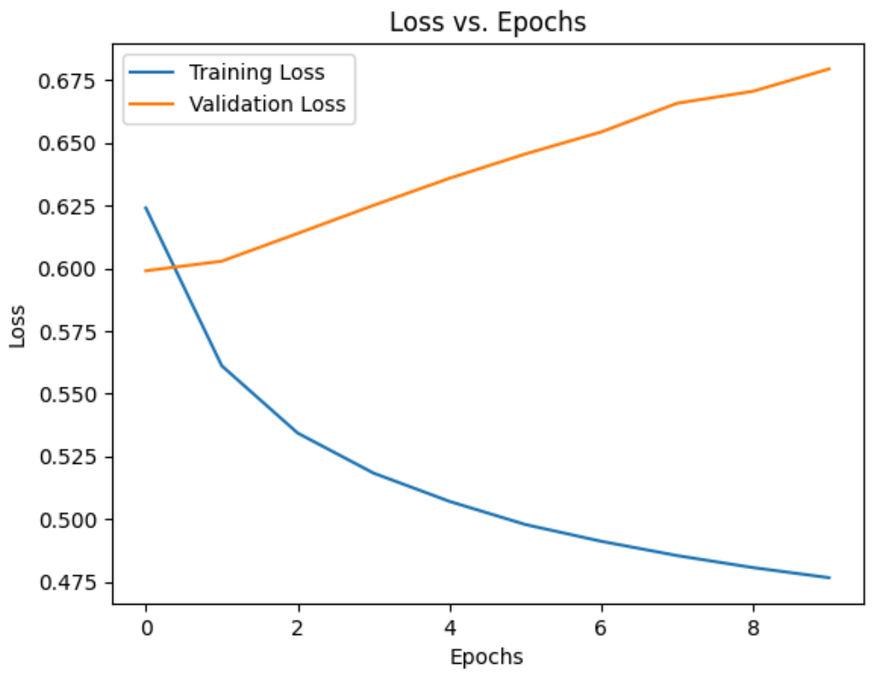
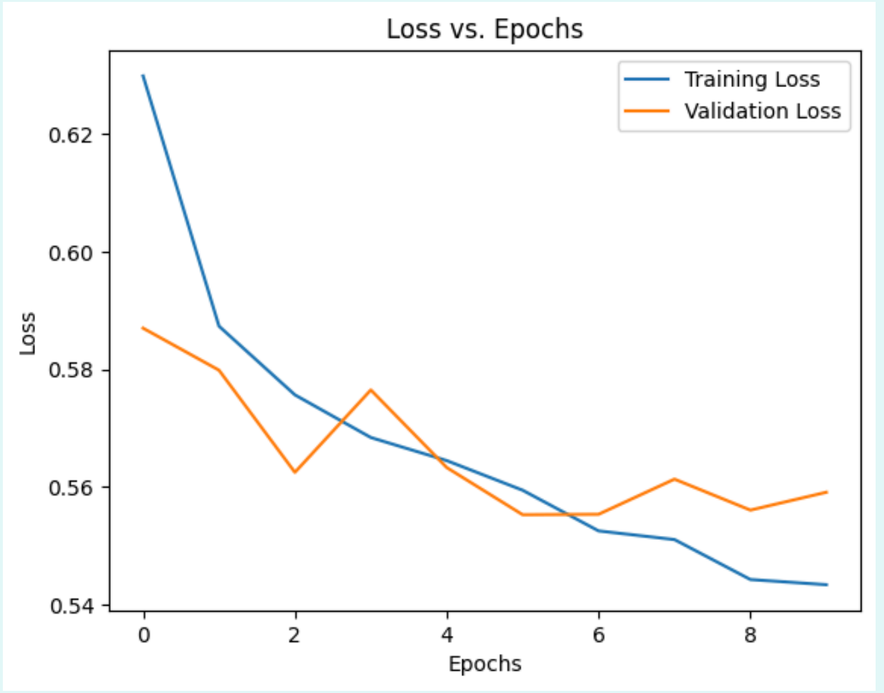
Model: Simple RNN (16 units), Dropout(0.2) Dense layer with sigmoid activation function
Learning Rate: 0.001
Batch Size: 32
Epoch: 10

Summary of Results for RNN with Different Embeddings

Embeddings	CV Precision	CV Accuracy	CV Recall	CV F1
TF-IDF (Baseline Model)	0.70	0.66	0.77	0.74
BERT Embedding	0.74	0.71	0.81	0.78

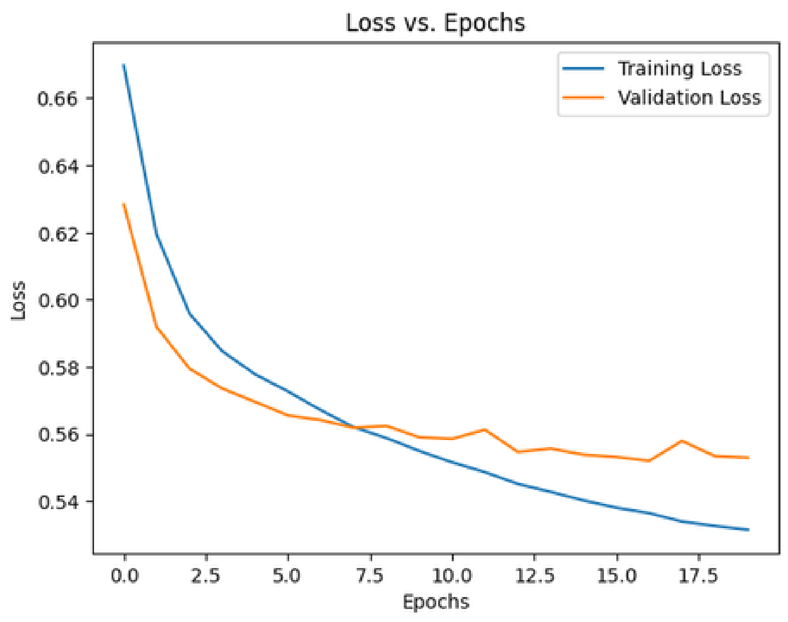
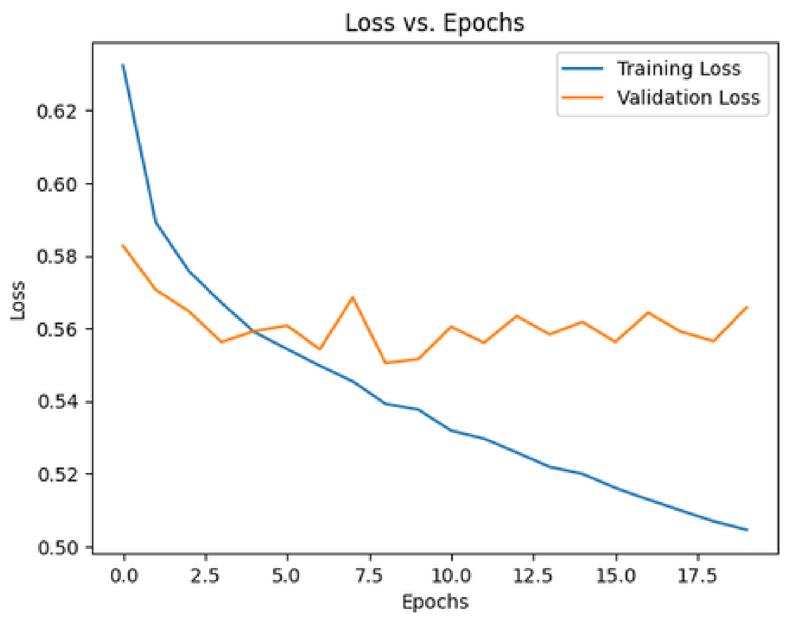
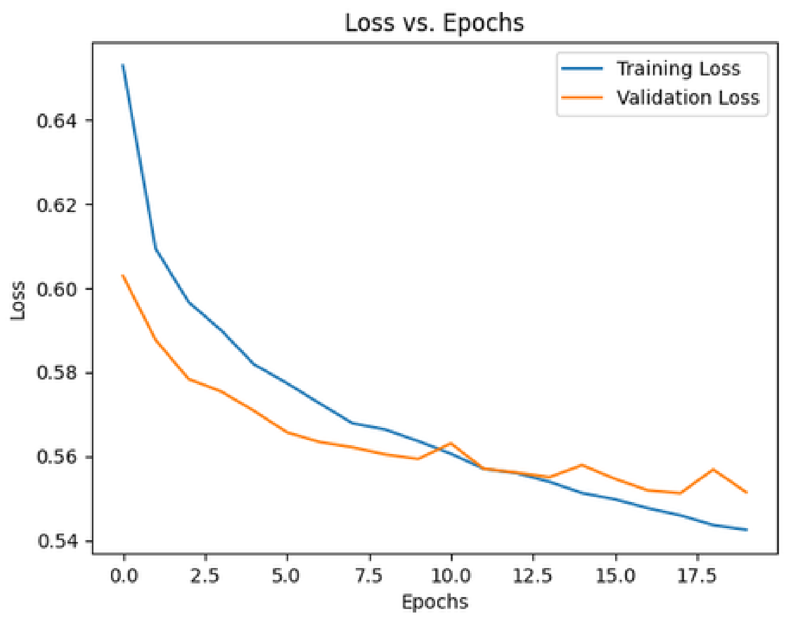
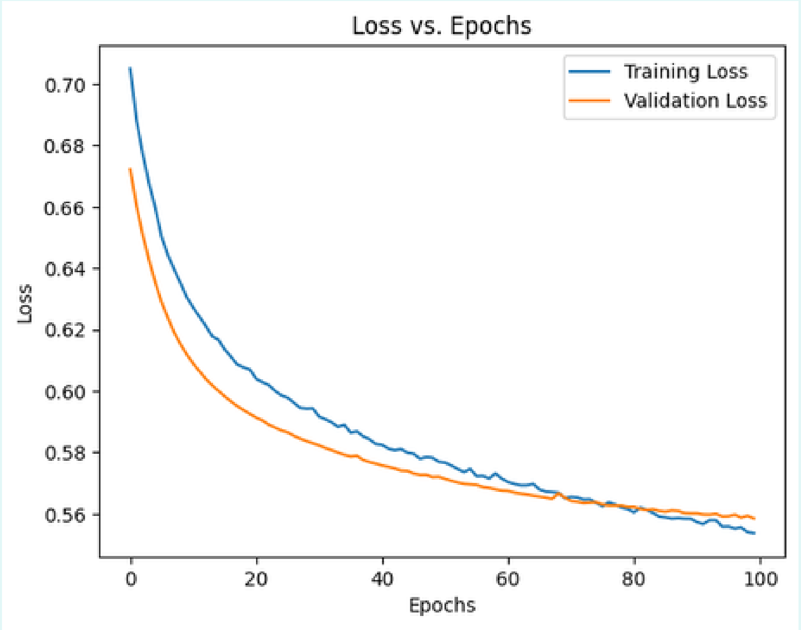
Experiments with Embedding

Loss-Epoch Graphs for RNN with Different Embeddings

	TFIDF	BERT
Loss - Epoch Graph		

Experiment with Batch Size (BS) and Learning Rate (LR)

Loss - Epoch Graphs for RNN with Different Batch Size and Learning Rate

<table>
<tr><th></th><th>BS = 32
LR = 0.0001</th><th>BS = 64
LR = 0.001</th><th>BS = 64
LR = 0.0001</th><th>BS = 64
LR = 0.00001</th></tr>
<tr><td>Loss - Epoch Graph</td>
<td> 
</td>
<td> 
</td>
<td> 
</td>
<td> 
</td></tr>
</table>

Experiment with Class Weight (CW)

Summary of Results for RNN with and without Class Weight and different BS, LR and Epochs

Embeddings	Precision	Accuracy	Recall	F1
No CW, BS = 64, LR = 0.0001	0.76	0.70	0.77	0.76
With CW, BS = 64, LR = 0.0001	0.77	0.70	0.73	0.75
With CW, BS = 64, LR = 0.00001	0.79	0.69	0.68	0.73
With CW, BS = 128, LR = 0.00001	0.78	0.69	0.69	0.73

Final Model

Model: Simple RNN (16 units)
Dropout(0.2)
Dense with Sigmoid Activation Function

Loss: 'binary_crossentropy'
Learning Rate: 0.0001
Batch Size: 64
Epoch: 100
With Class Weight

	Predicted 0	Predicted 1
True 0	22956	8757
True 1	16486	34565

TN	FP
FN	TP

CV Precision	CV Accuracy	CV Recall	CV F1
0.80	0.70	0.68	0.73

Improved Model 3

Random Forest

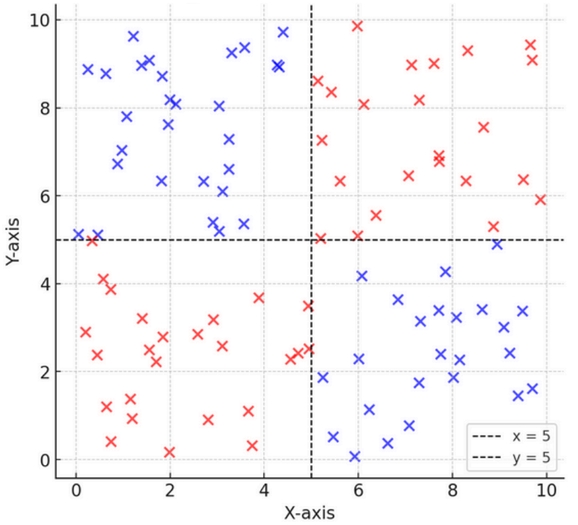
Why We Chose Random Forest

Random Forest operates by constructing multiple **decision trees** during training. Once all decision trees are constructed, they collectively make predictions. In our case, each tree votes for whether a data comment is sarcasm or not, and the class with the highest votes becomes the final prediction.

col 1	...	col n
	...	
	...	
	...	

Text embeddings or engineered features often result in high-dimensional datasets, where not all features are equally useful. Random Forest can automatically identify and prioritize the most relevant features, reducing the impact of noisy or irrelevant ones.

Random Forest can model complex, non-linear relationships between features because decision trees can split data into specific regions using thresholds. By averaging across many such trees, Random Forest captures intricate patterns that simpler models (e.g., linear models) might miss.



Summary of results for Random Forest

We tried random forest on 2 different embeddings, both with the numerical features. 5-fold cross validation (CV) was applied to each model and the average precision, accuracy, recall & F1 score of the folds were taken:

Embeddings	**CV Precision**	**CV Accuracy**	**CV Recall**	**CV F1**
TF-IDF (Baseline Model)	0.68	0.67	0.88	0.77
Bi-grams with TF-IDF	0.71	0.69	0.84	0.77

Building Random Forest with TF-IDF Embeddings after Bi-grams gave higher precision, so we will fine-tune the Random Forests built with Bi-grams with TF-IDF embeddings

We wanted to build a Random Forest with BERT Embeddings. However, the BERT embeddings was 3 dimensional and needed to be flattened before it can used for training Random Forest. That exceeded the RAM provided by Google Colab, so we had to give up on that.

Finetuning Random Forest

We used 5-fold cross validation (CV) to fine-tune the number of trees in the forest and the function to measure the quality of a split. The average precision, accuracy, recall, F1 score of the folds were taken:

Parameters	CV Precision	CV Accuracy	CV Recall	CV F1
100, gini	0.71	0.69	0.84	0.77
100, entropy	0.72	0.69	0.82	0.77
150, entropy	0.72	0.69	0.83	0.77
200, entropy	0.72	0.69	0.83	0.77
150, entropy, balanced class weight	0.74	0.70	0.79	0.76

Breakdown of Predicted and True Label

This is the cross-validated predictions of the random forest with the highest precision against the ground truth

Parameters	CV Precision	CV Accuracy	CV Recall	CV F1
150, entropy, balanced class weight	0.74	0.70	0.79	0.76

Recap:

PRECISION	RECALL
$\frac{TP}{TP+FP}$	$\frac{TP}{TP+FN}$

Confusion matrix of the best random forest:

	Predicted 0	Predicted 1
True 0	16757	13923
True 1	10760	39683

Since 0 is seen as not sarcasm, it will be the negative. And 1 is sarcasm, it will be the positive.

TN	FP
FN	TP

Evaluation of Finetuning

After fine-tuning, there wasn't significant improvement in the precision of Random Forest.

- Using entropy to measure the quality of split helps to increase the precision score slightly.

Parameters	CV Precision	CV Recall
100, gini	0.71	0.84
100, entropy	0.72	0.82

PRECISION

$$\frac{TP}{TP+FP}$$

RECALL

$$\frac{TP}{TP+FN}$$

	Predicted 0	Predicted 1
True 0	TN	FP ↓
True 1	FN ↑	TP

The logarithmic component in entropy allows it to better capture the presence of minority classes, leading to splits that consider minority classes more effectively. (Kaplan, 2024)

Since more focus is placed on the minority class 0, lesser label 0 instances are predicted wrongly, so precision increased slightly.

However, recall decrease slightly because less focus is placed on the majority label 1, so more label 1 instance is predicted wrongly.

Evaluation of Finetuning

After fine-tuning, there wasn't significant improvement in the precision of Random Forest.

- Using balanced class weights helps to increase the precision score slightly.

Parameters	CV Precision	CV Recall
150, entropy	0.72	0.83
150, entropy, balanced class weight	0.74	0.79

PRECISION	RECALL
$\frac{TP}{TP+FP}$	$\frac{TP}{TP+FN}$

	Predicted 0	Predicted 1
True 0	TN	FP ↓
True 1	FN ↑	TP

When the class_weight is set to 'balanced', it influences the training process by penalizing misclassifications of minority class more heavily (Kamaldeep, 2024).

The model can learn decision boundaries that are more sensitive to the minority classes so less label 0 instances are predicted wrongly.

However, recall decrease because less focus is placed on the majority label 1, so more label 1 instance is predicted wrongly.

Evaluation of Finetuning

After fine-tuning, there wasn't significant improvement in the precision of Random Forest.

- Adding more trees does not help to improve the scores.

Parameters	CV Precision	CV Accuracy	CV Recall	CV F1
100, entropy	0.72	0.69	0.82	0.77
150, entropy	0.72	0.69	0.83	0.77
200, entropy	0.72	0.69	0.83	0.77

PRECISION

$$\frac{TP}{TP+FP}$$

RECALL

$$\frac{TP}{TP+FN}$$

	Predicted 0	Predicted 1
True 0	TN	FP
True 1	FN	TP

The performance of the Random Forest model plateaus despite the increase in trees because that is the most pattern the Random Forest could capture from the dataset while other parameters remained the same.

The Random Forest model may have other suboptimal hyperparameter settings (e.g., too few features considered per split) which we did not finetune, increasing the number of trees cannot compensate for these limitations.

Evaluation

Evaluation Against Test Set: Final Linear SVM Model

Results for Evaluating the Model Performance Using K-Cross Validation where K=5

CV Precision	CV Accuracy	CV Recall	CV F1
0.78	0.61	0.52	0.63

Results for Evaluating the Model Performance Against Test Set

Test Precision	Test Accuracy	Test Recall	Test F1
0.79	0.62	0.53	0.64

Since the cross-validation metrics and test metrics are very close, with slight improvements in the test set scores, the model generalises well to data that are unseen, with no evidence of overfitting.

Insights: Final Linear SVM Model

Results for Evaluating the Model Performance Against Test Set

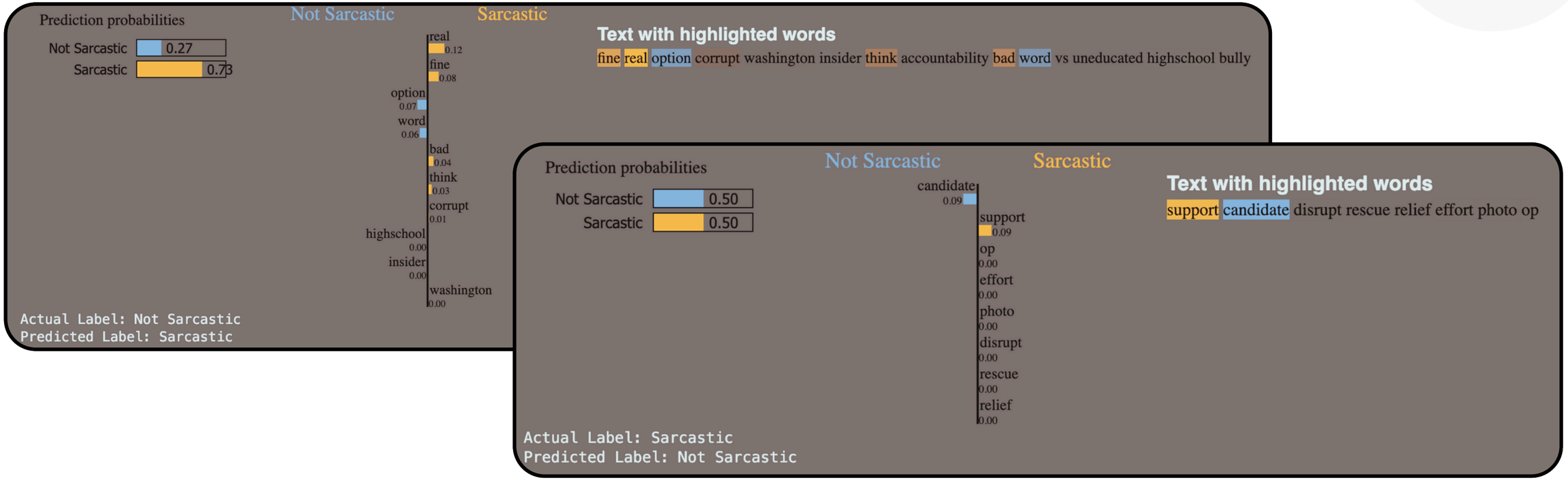
Test Precision	Test Accuracy	Test Recall	Test F1
0.79	0.62	0.53	0.64

	Predicted 0	Predicted 1
True 0	23968	7035
True 1	23663	26857

TN	FP
FN	TP

- Low recall suggests that the model struggles to identify positive samples
- The high precision however suggests that the model is very confident whenever it predicts a positive class

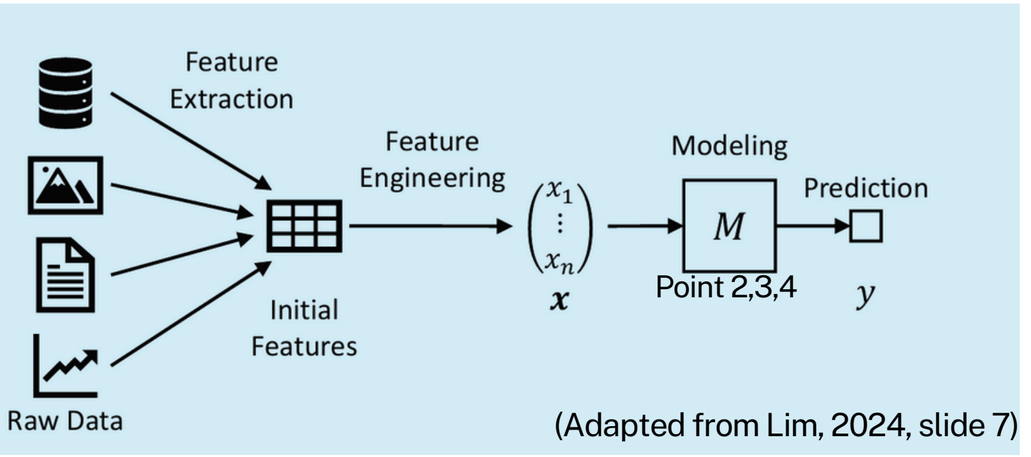
Model Explainability Using LIME



- LIME explains which words (or features) in the input text **contribute positively or negatively** to the predicted class
- **Degree of contribution** for each word
- Predictions on a **specific instance** rather than globally describing how the model works overall
- Words that make the model decide **"Sarcastic"** over **"Not Sarcastic"**
- It tells us whether the model focuses on **relevant words** (e.g., sarcasm-indicating words) or identifies **irrelevant features**

Next Steps

Improving Performance of each Model



1. Include more features for model training
   - month of comment
   - subreddit category
   - topic modelling to categorise into different topics
2. SVM used TFIDF embeddings as a feature, could consider using BERT embeddings that capture semantic relationships and meaning in texts (D'Sa, 2020)
3. Fine-tune other hyperparameters of Random Forest that we did not manage to fine-tune
4. Recurrent Neural Networks
   - Fine-tune the number of hidden layers
   - Try to use Long Short-Term Memory (LSTM) to take in the BERT Embeddings and make predictions (Pandey & Singh, 2023)

Thank you!