Dive Into Anything

AI-Powered Moral Judgement Dilemmas of Reddit

Group 1

Aafrin Shehnaz, Aiswarya Raghavadesikan,
Shreenithi Sivakumar, Shivram Sriramulu,
Yogavarshni Ramachandran

Problem Statement

- In the vibrant and interactive world of Reddit, the subreddit r/AmItheAsshole (AITA) provides a unique platform where users share personal stories and seek community judgments about their actions in various life scenarios.
- Participants in these discussions contribute by voting and commenting, declaring whether they believe the poster acted wrongly ("You're the Asshole") or not ("Not the Asshole").
- Currently, the process relies entirely on human interaction, which can delay responses and feedback due to the volume of posts and varying reader availability.

r/AITA

3.9M subscribers

121 posts per day

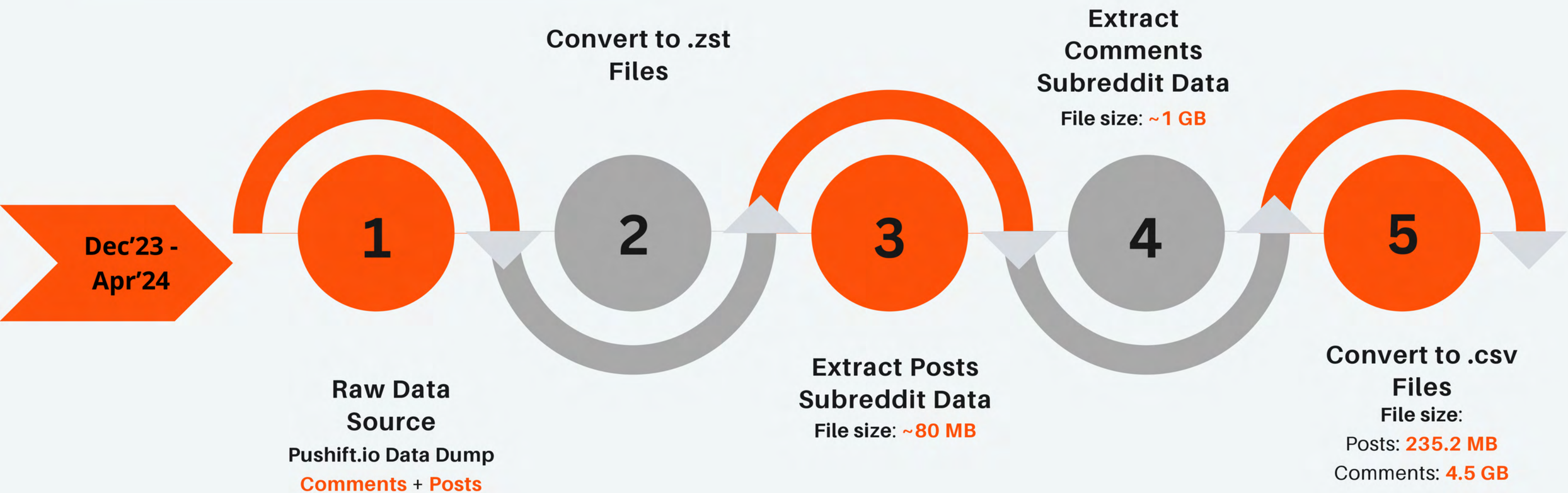
2800 comments per day

3 million pageviews per month

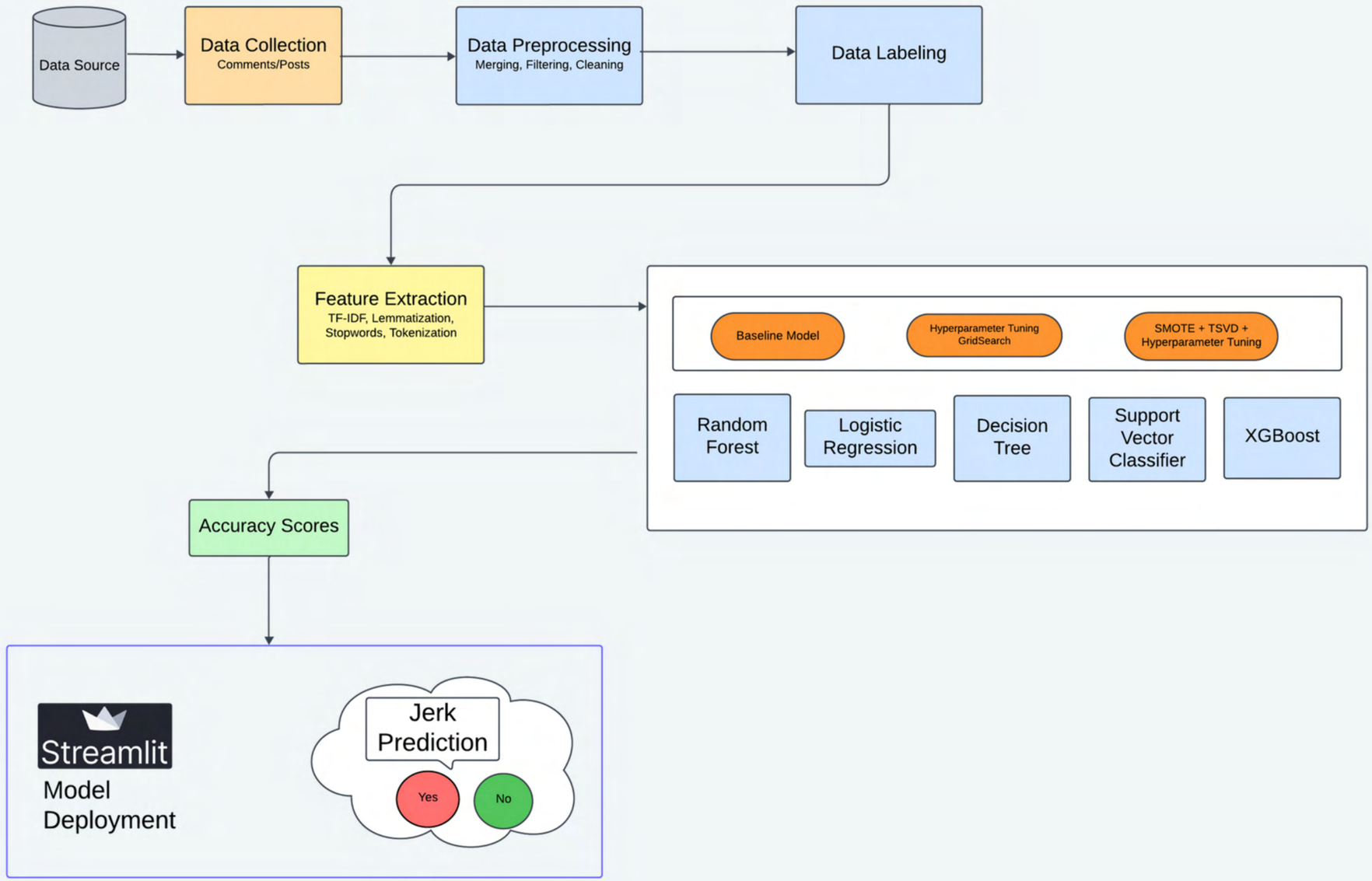
Goal

- Leveraging natural language processing and Machine learning techniques, to analyze the textual content of posts and deliver an immediate judgment of "Yes" or "No" to the poster, simulating a virtual community response.
- This automation aims to provide instant feedback to users, enriching their interactive experience on the platform and reducing their dependency on the availability and engagement of other readers.

Data Extraction & Conversion



System Architecture



Exploratory Data Analysis

Comments Dataset: Filtered : link_id=parent_id

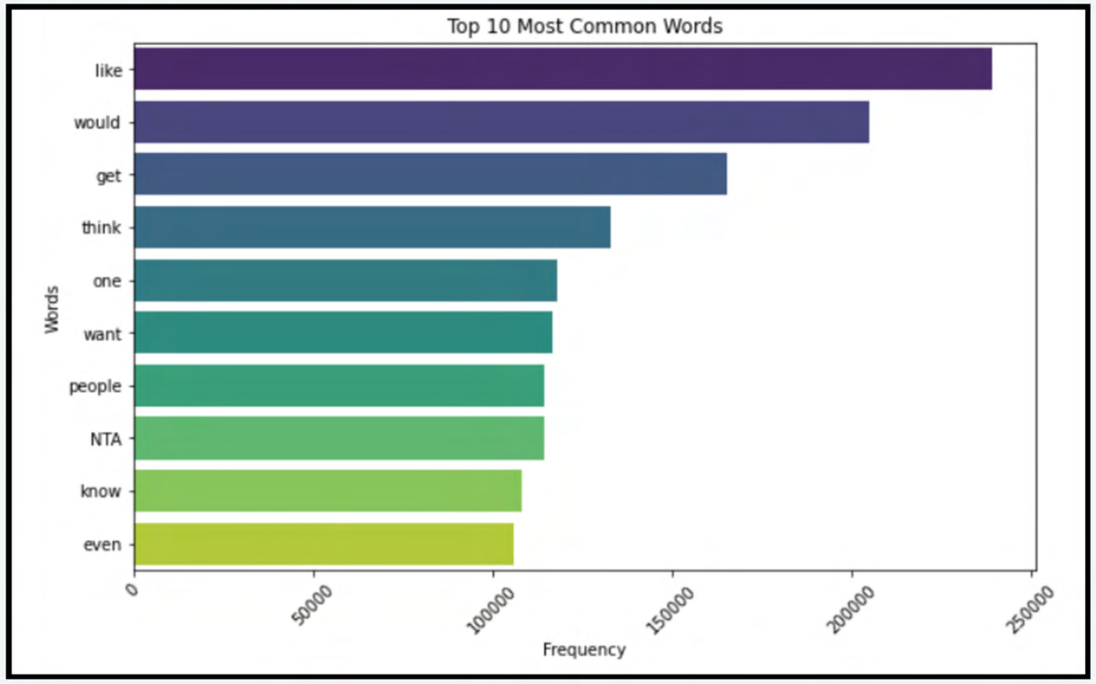
```
df.shape
```

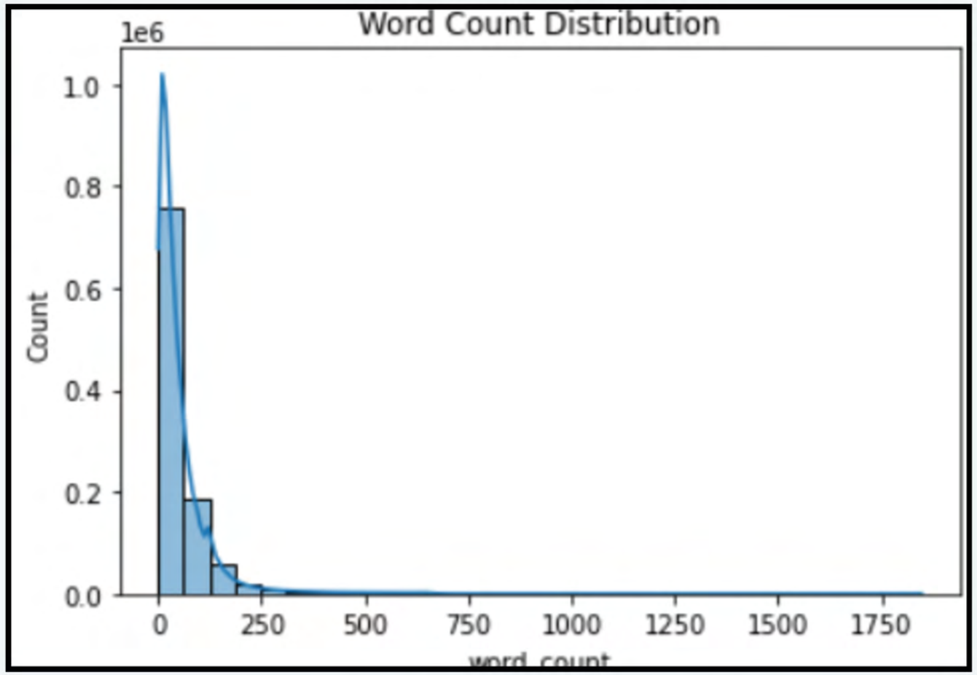
```
(1048573, 5)
```

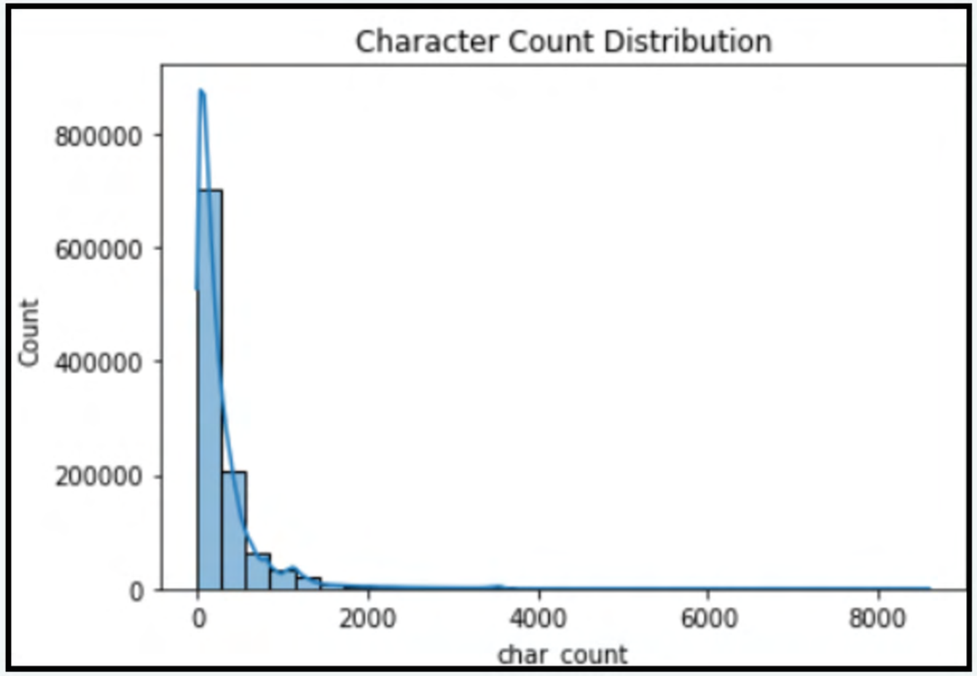
```
filtered_df.shape
```

```
(575247, 5)
```

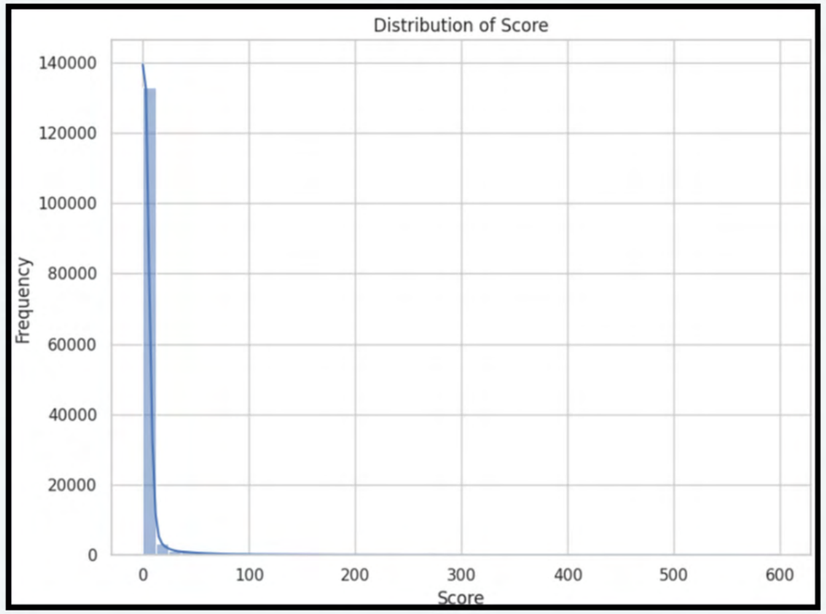
“body” - used for labeling





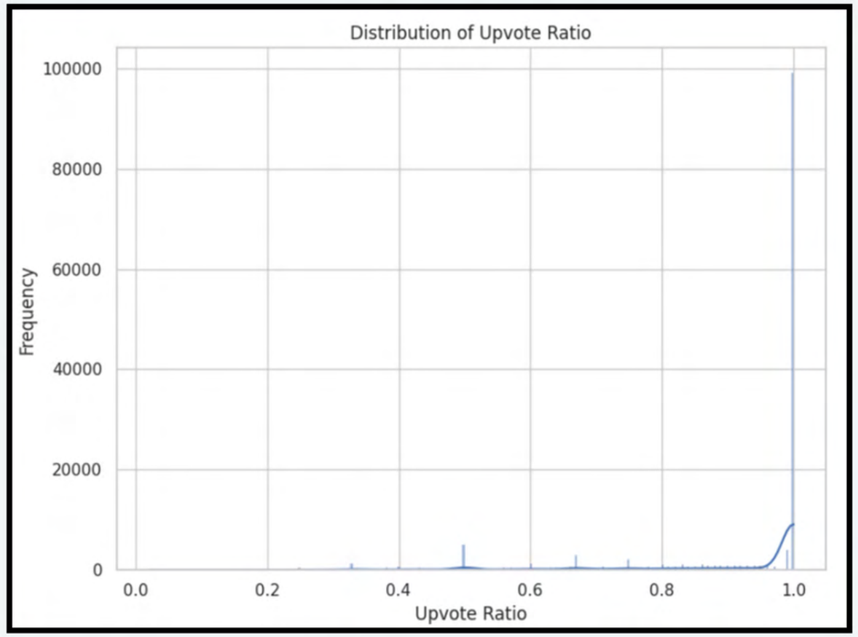


Exploratory Data Analysis

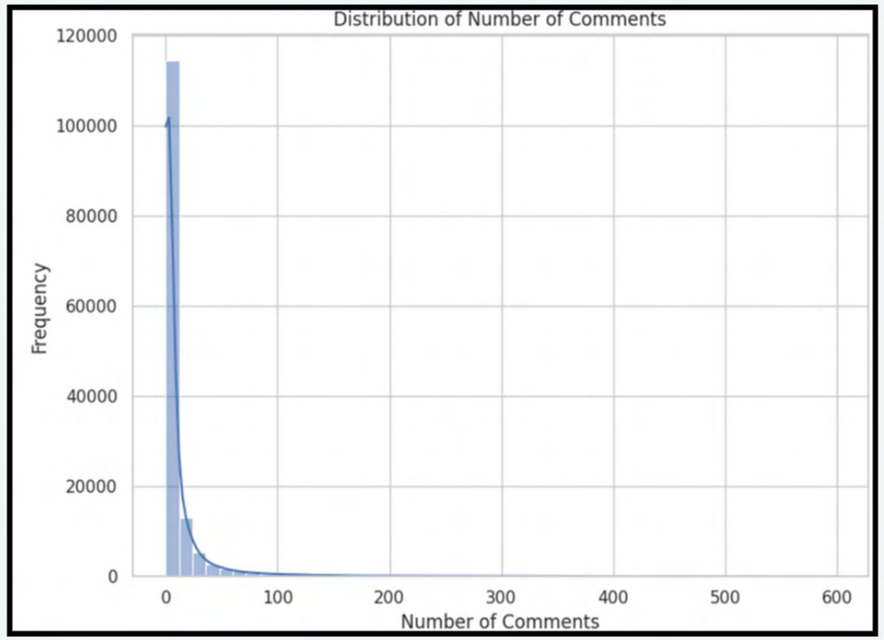


Submission Dataset:

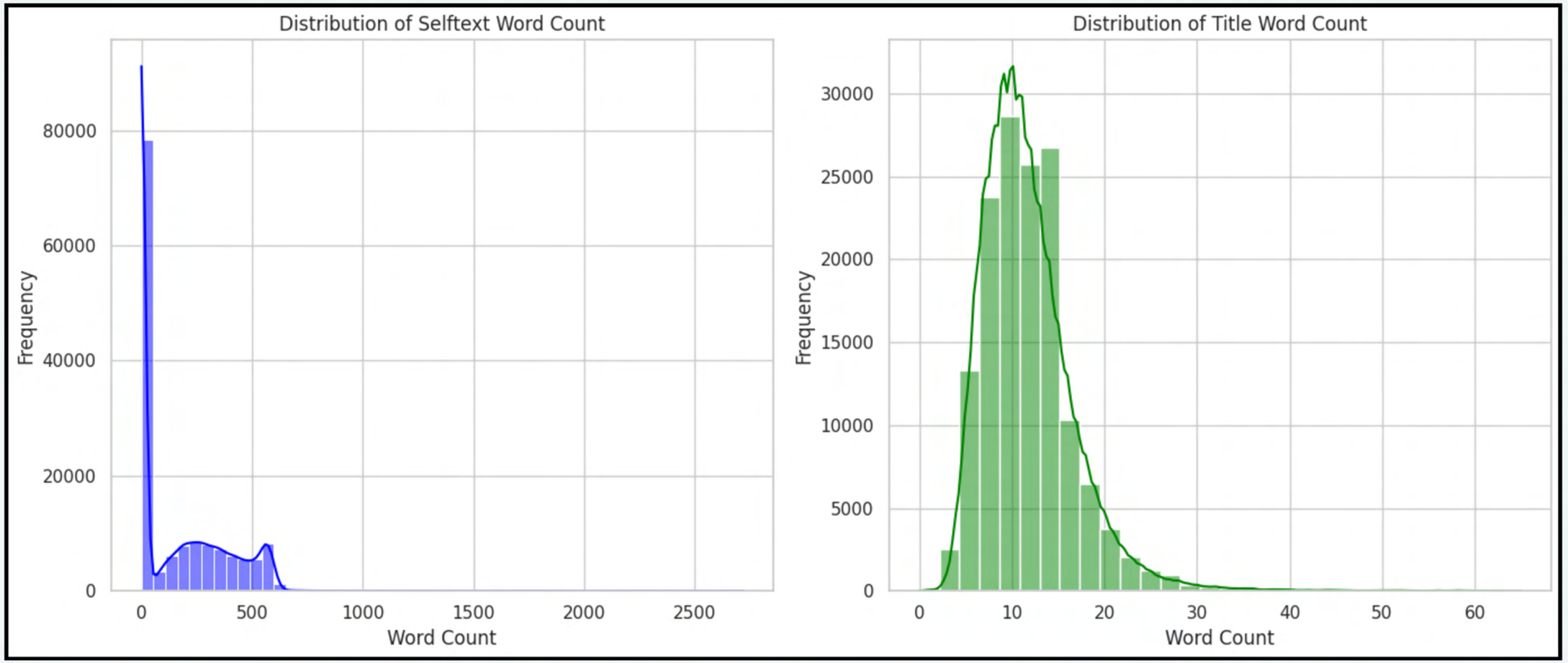
Mostly it is ups, very few downs



right-skewed distribution



Exploratory Data Analysis



Most posts are brief, yet some contain a substantial amount of text, suggesting they offer detailed information.

Exploratory Data Analysis

WordCloud of selftext

Focus: ‘family’, ‘mom’, ‘feel’, ‘asked’.
Frequency: ‘feel’, ‘want’

WordCloud of Title

Focus: ‘friend’,’girlfriend’,’boyfriend’,’husband’,wife’, ‘sister’.
Frequency: ‘refusing’,’accept’,’calling’,’wanting’.

WordCloud of selftext

WordCloud of Title

Data Cleaning

Comments

- Drop all columns except link_id, parent_id and body
- Filter out only the top level comments
- Remove rows where 'body' equals '[removed]' or [deleted] or is NaN
- Data Labelling

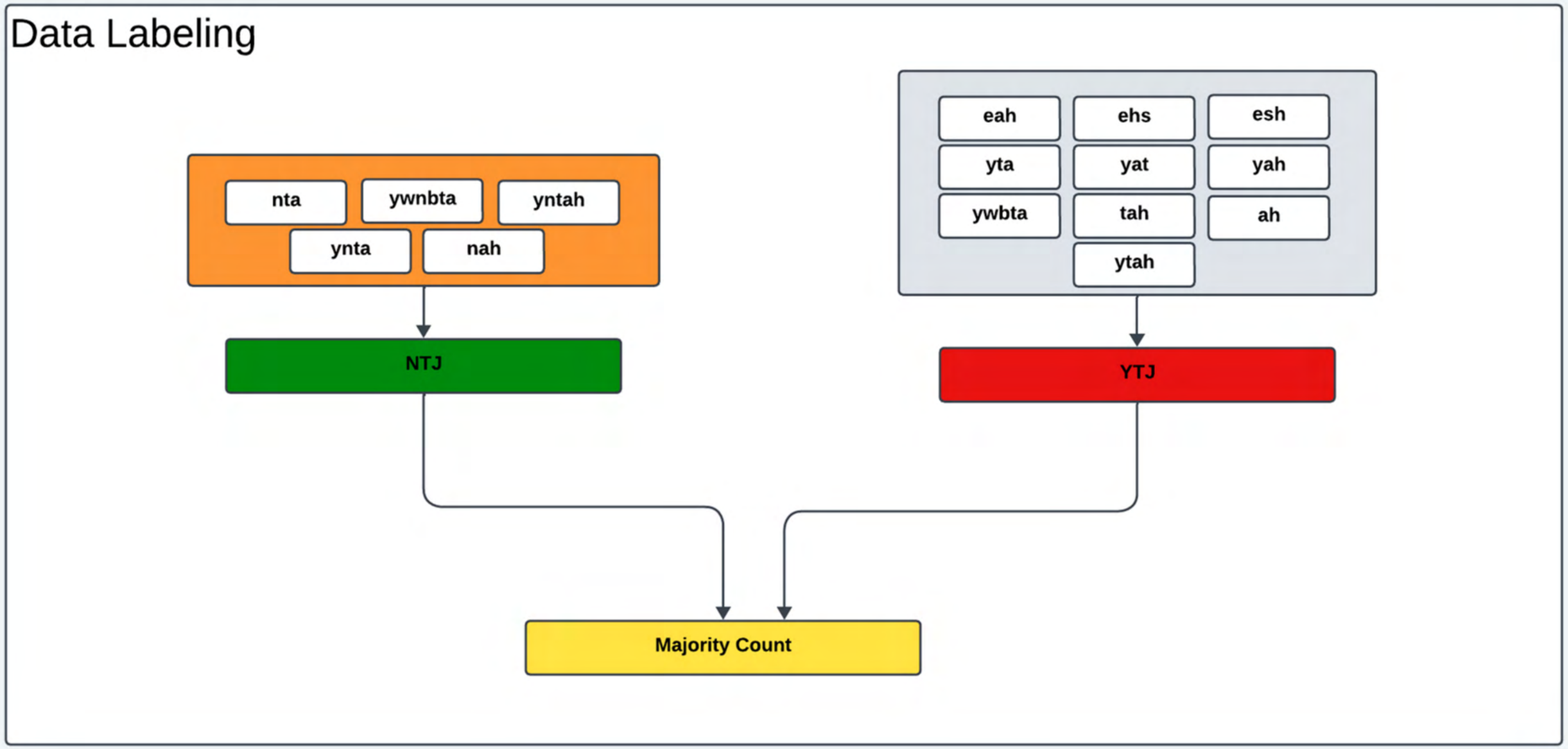
Posts

- Drop all columns except selftext and name (id column)

	link_id	final_label
0	t3_142jic8	ntj
2	t3_143yx8q	ntj
3	t3_144u0s6	ntj
4	t3_149bfca	ntj
5	t3_149ssj1	ytj

Comments

Data Labeling



Data Cleaning (Contd.)

Final Dataset Shape

```
df.shape
```

```
(45639, 4)
```

Final Dataset Sample

```
df.head()
```

	link_id	label	selftext	name
0	t3_187xeed	ytj	I have never been on here asking for help befo...	t3_187xeed
1	t3_187xeha	ytj	While mv girlfriend 29F was working in Montrea...	t3_187xeha
2	t3_187xh0d	ntj	I'm the most junior member of a small team (bo...	t3_187xh0d
3	t3_187xmox	ntj	We lived in 4 single bedroom dorm, my roommate...	t3_187xmox
4	t3_187xooh	ntj	So I'm a bit of a mess. Due to my mental healt...	t3_187xooh

TF-IDF

Lemmatization

Breakingdown a word to its root meaning.

Tokenization

Process of converting a sequence of text into smaller parts, tokens.

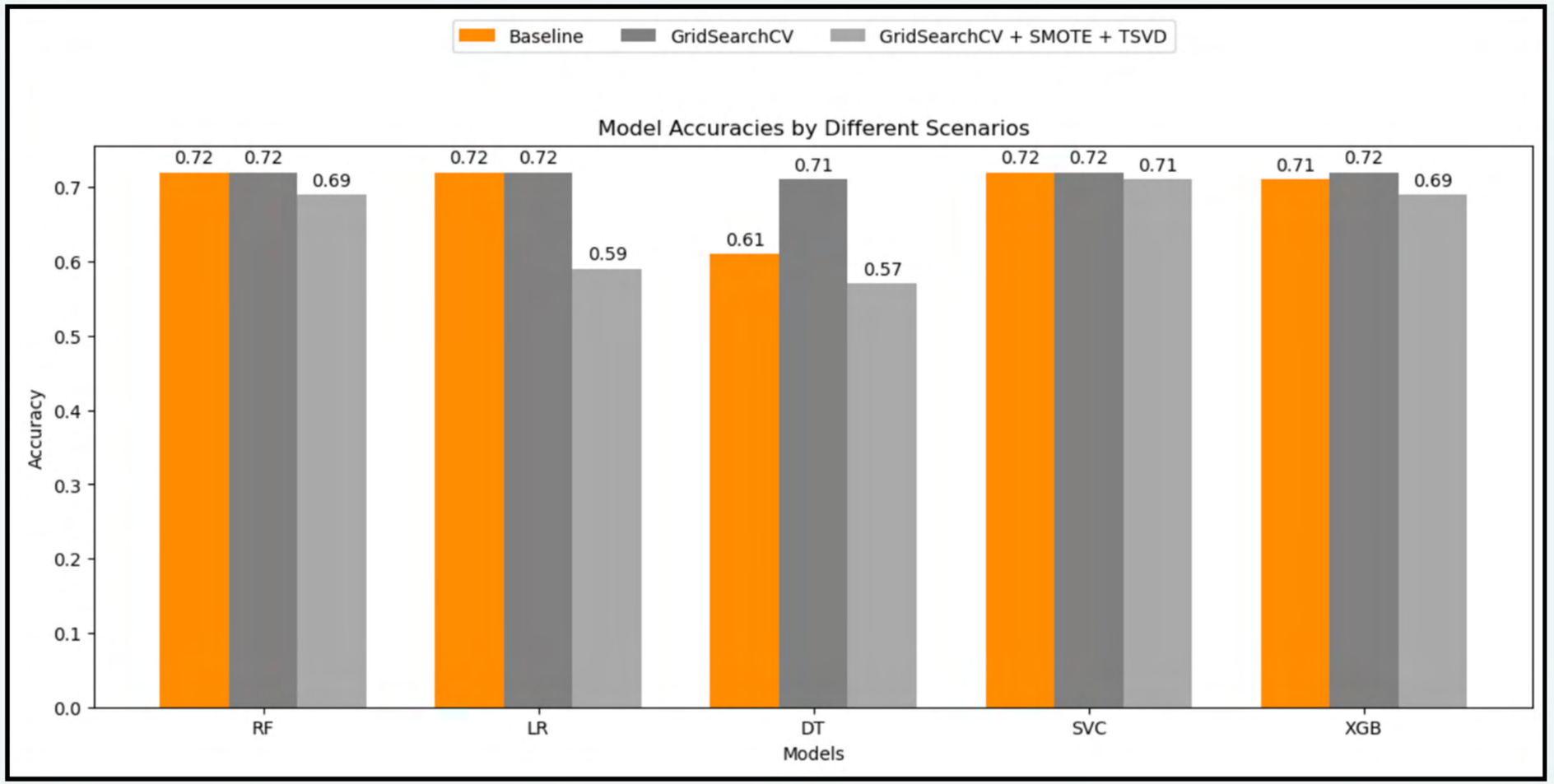
Stop Word Removal

Removal of words that occur commonly across all the documents in the corpus

```python
# Function to map NLTK's part-of-speech
# tags to wordnet tags
def get_wordnet_pos(treebank_tag):
    if treebank_tag.startswith('J'):
        return nltk.corpus.wordnet.ADJ
    elif treebank_tag.startswith('V'):
        return nltk.corpus.wordnet.VERB
    elif treebank_tag.startswith('N'):
        return nltk.corpus.wordnet.NOUN
    elif treebank_tag.startswith('R'):
        return nltk.corpus.wordnet.ADV
    else:
        return nltk.corpus.wordnet.NOUN

# Initialize the WordNet Lemmatizer
lemmatizer = WordNetLemmatizer()

# Set of English stop words
stop_words = set(stopwords.words('english'))

def nltk_lemmatizer(text):
    words = word_tokenize(text)
    pos_tags = pos_tag(words)
    lemmas = [
        lemmatizer.lemmatize(word, get_wordnet_pos(pos))
        for word, pos in pos_tags
        if word.isalpha() and word.lower() not in stop_words
    ]
    return lemmas
```

```python
text_transformer = TfidfVectorizer(tokenizer=nltk_lemmatizer, token_pattern=None)
```

Accuracy of Models

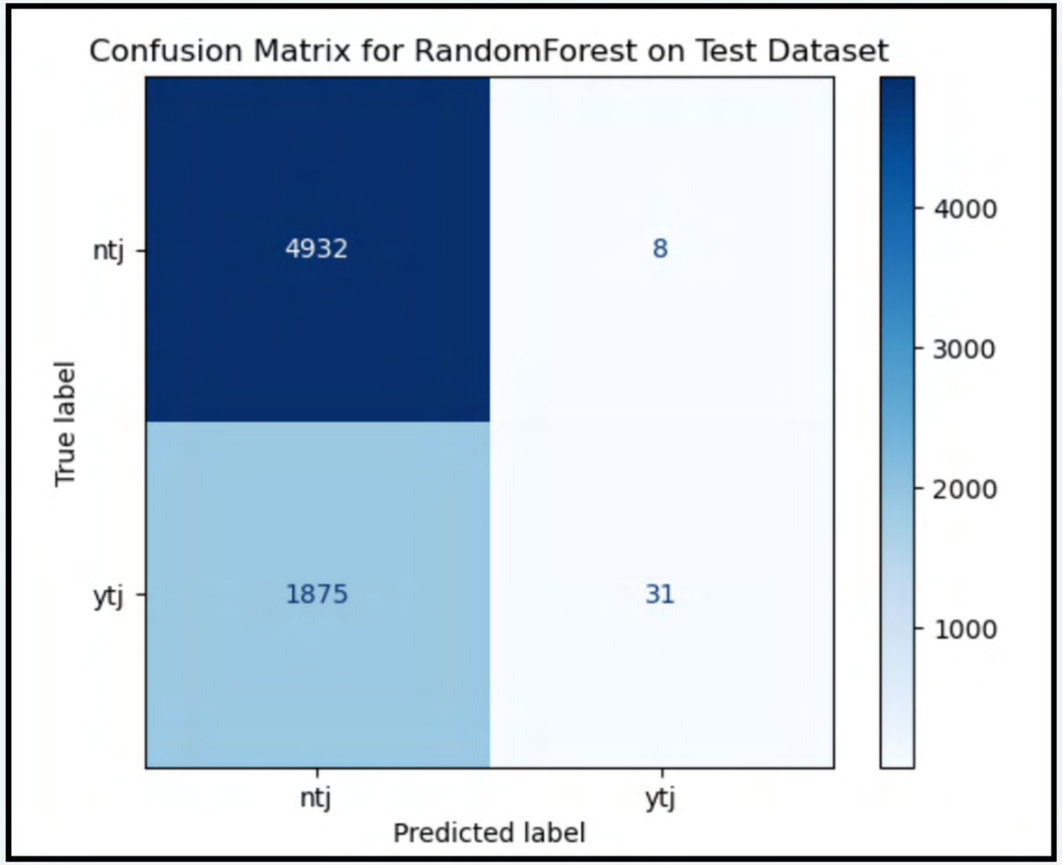


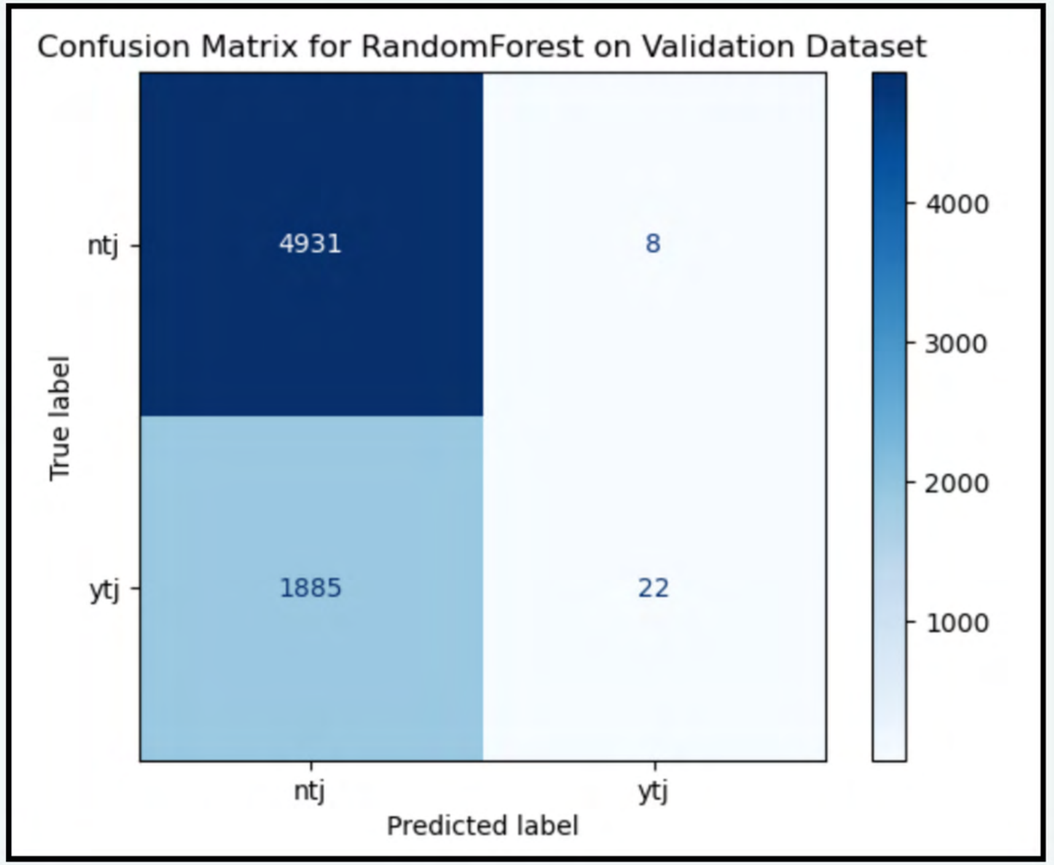
Random Forest

```
Fitting 3 folds for each of 108 candidates, totalling 324 fits
Best parameters for Random Forest: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimator
s': 100}
```

```
Classification Report for RandomForest on Test Dataset:
              precision    recall  f1-score   support

           0       0.72      1.00      0.84      4940
           1       0.79      0.02      0.03      1906

    accuracy                           0.72      6846
   macro avg       0.76      0.51      0.44      6846
weighted avg       0.74      0.72      0.61      6846

Classification Report for RandomForest on Validation Dataset:
              precision    recall  f1-score   support

           0       0.72      1.00      0.84      4939
           1       0.73      0.01      0.02      1907

    accuracy                           0.72      6846
   macro avg       0.73      0.50      0.43      6846
weighted avg       0.73      0.72      0.61      6846
```

Random Forest



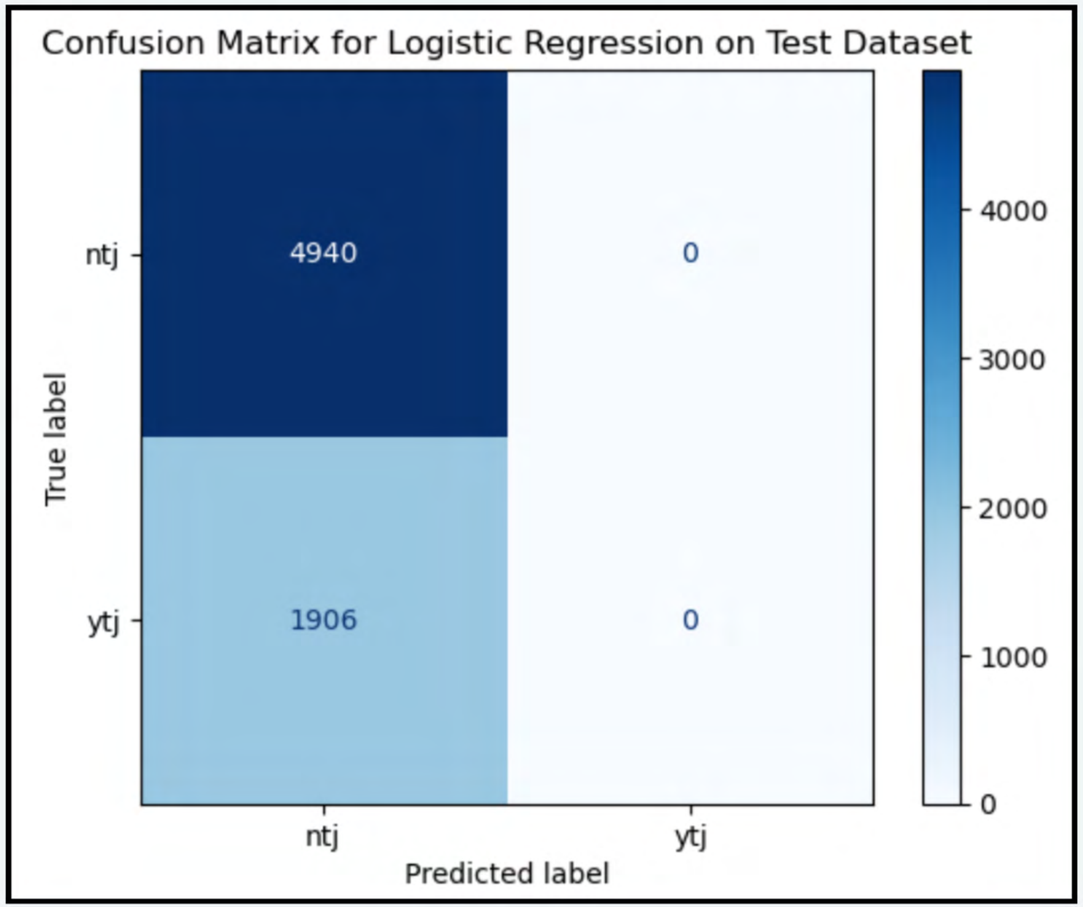


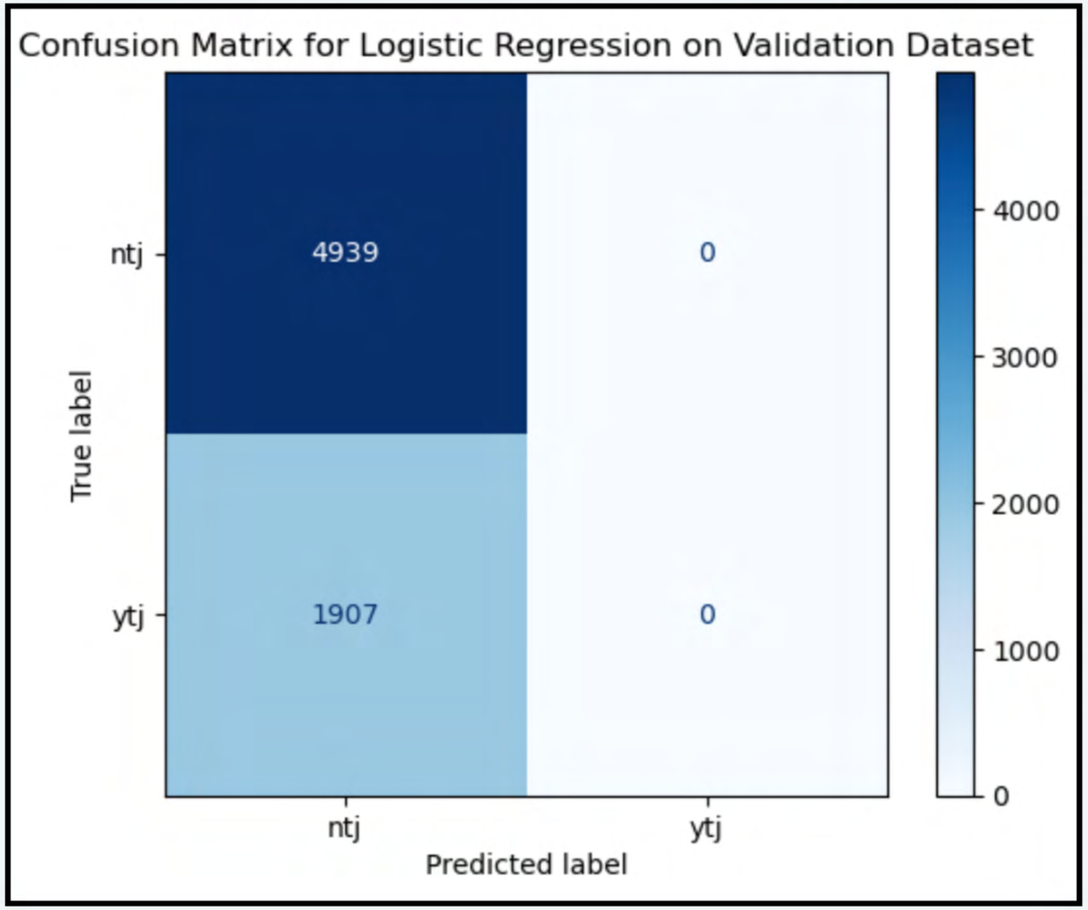
Logistic Regression

```
Fitting 3 folds for each of 32 candidates, totalling 96 fits
Best parameters for Logistic Regression: {'C': 0.01, 'class_weight': 'balanced', 'max_iter': 100, 'penalty': 'l1',
'solver': 'liblinear'}
```

```
Classification Report for Logistic Regression on Test Dataset:
              precision    recall  f1-score   support

           0       0.72      1.00      0.84      4940
           1       0.00      0.00      0.00      1906

    accuracy                           0.72      6846
   macro avg       0.36      0.50      0.42      6846
weighted avg       0.52      0.72      0.60      6846

Classification Report for Logistic Regression on Validation Dataset:
              precision    recall  f1-score   support

           0       0.72      1.00      0.84      4939
           1       0.00      0.00      0.00      1907

    accuracy                           0.72      6846
   macro avg       0.36      0.50      0.42      6846
weighted avg       0.52      0.72      0.60      6846
```

Logistic Regression





Decision trees

```
Fitting 3 folds for each of 36 candidates, totalling 108 fits
Best parameters for Decision Tree: {'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split': 2}
```

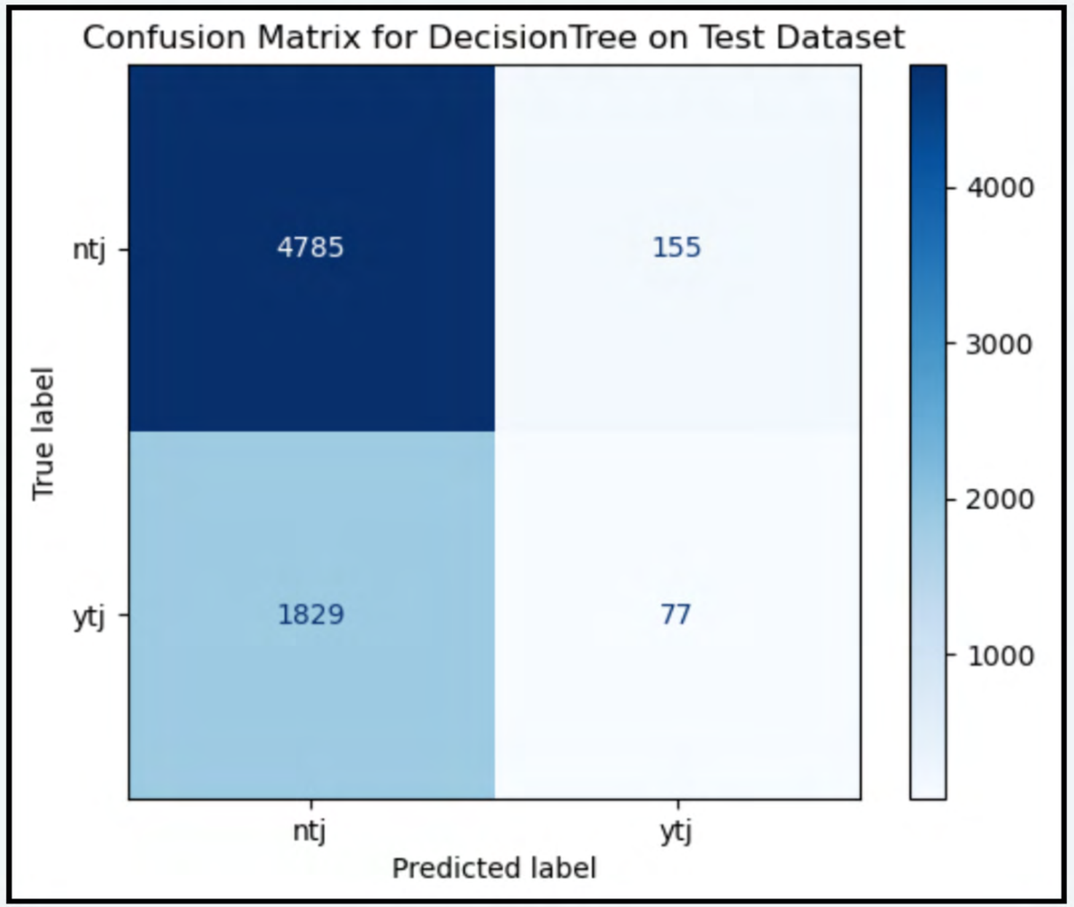
Classification Report for DecisionTree on Test Dataset:

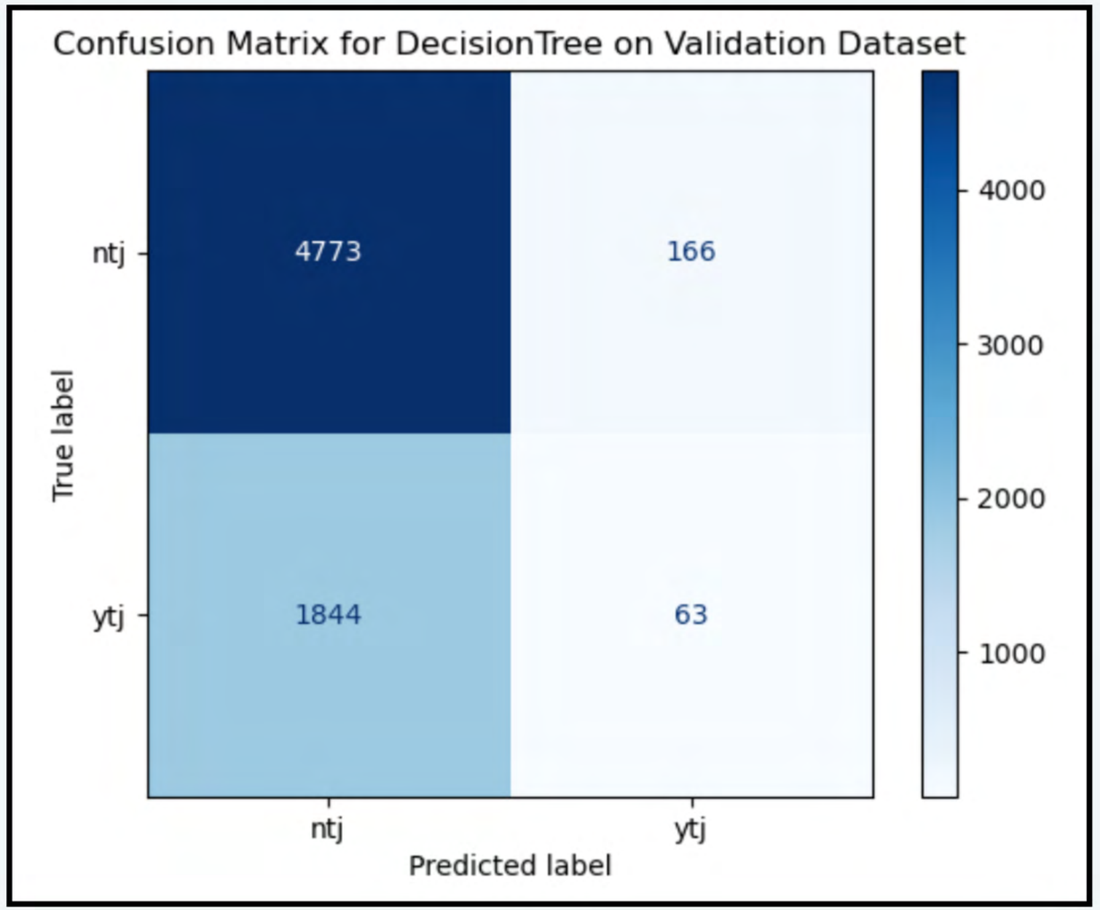
	precision	recall	f1-score	support
0	0.72	0.97	0.83	4940
1	0.33	0.04	0.07	1906
accuracy			0.71	6846
macro avg	0.53	0.50	0.45	6846
weighted avg	0.61	0.71	0.62	6846

Classification Report for DecisionTree on Validation Dataset:

	precision	recall	f1-score	support
0	0.72	0.97	0.83	4939
1	0.28	0.03	0.06	1907
accuracy			0.71	6846
macro avg	0.50	0.50	0.44	6846
weighted avg	0.60	0.71	0.61	6846

Decision trees





Support Vector Classifier

```
Fitting 3 folds for each of 6 candidates, totalling 18 fits
Best parameters for SVC: {'C': 1, 'kernel': 'rbf'}
```

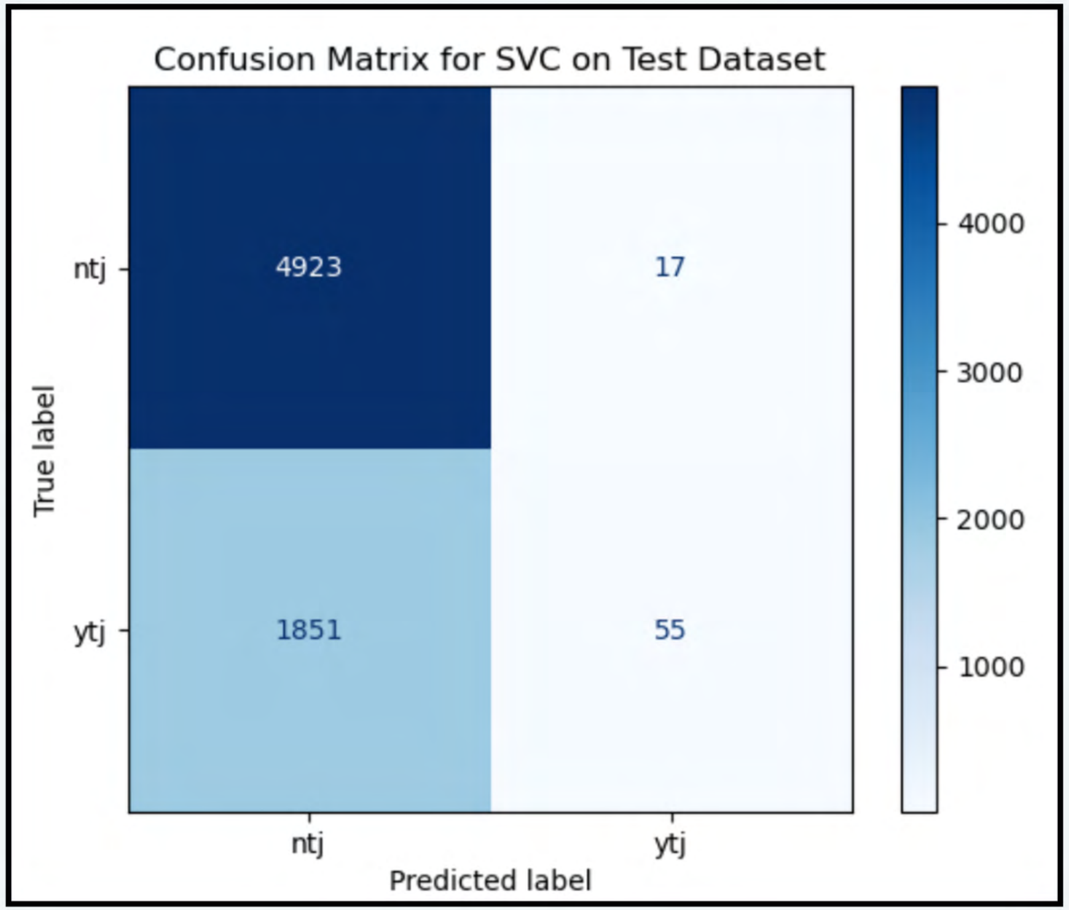
Classification Report for SVC on Test Dataset:

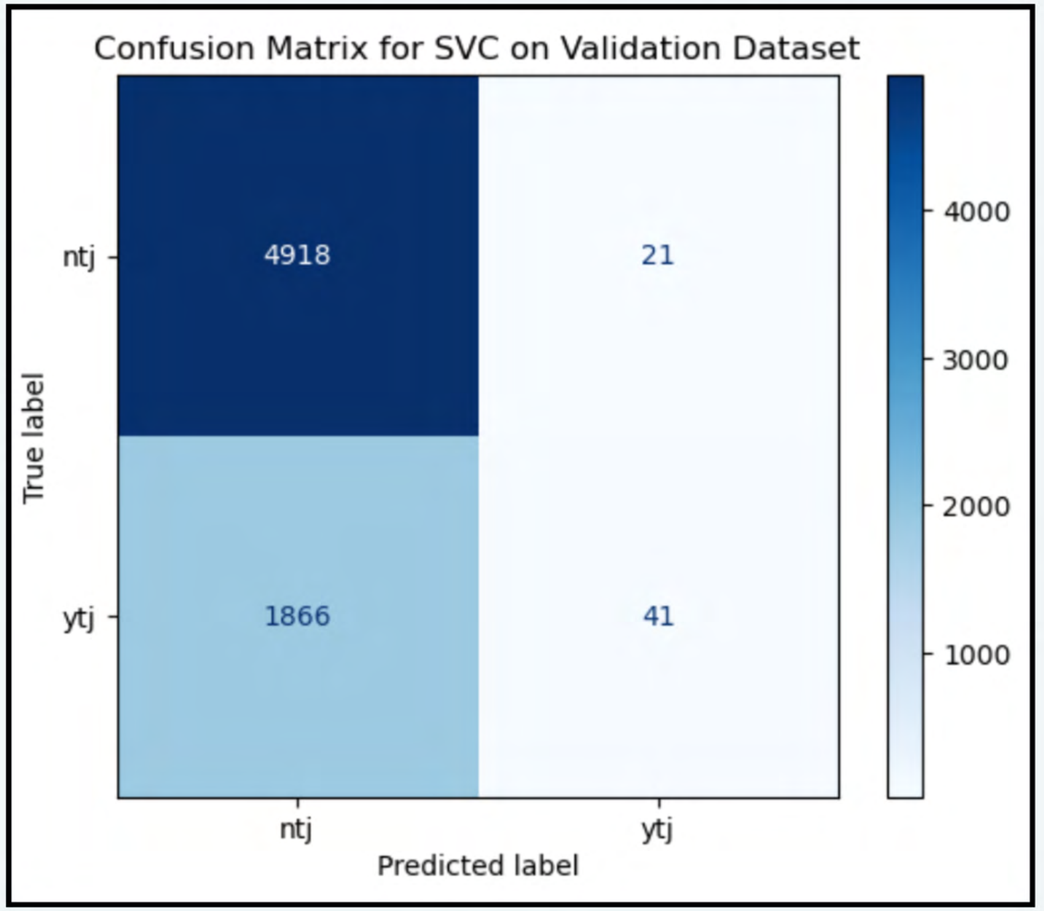
	precision	recall	f1-score	support
0	0.73	1.00	0.84	4940
1	0.76	0.03	0.06	1906
accuracy			0.73	6846
macro avg	0.75	0.51	0.45	6846
weighted avg	0.74	0.73	0.62	6846

Classification Report for SVC on Validation Dataset:

	precision	recall	f1-score	support
0	0.72	1.00	0.84	4939
1	0.66	0.02	0.04	1907
accuracy			0.72	6846
macro avg	0.69	0.51	0.44	6846
weighted avg	0.71	0.72	0.62	6846

Support Vector Classifier





XG Boost

```
Fitting 3 folds for each of 27 candidates, totalling 81 fits
Best parameters for XGBoost: {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 100}
```

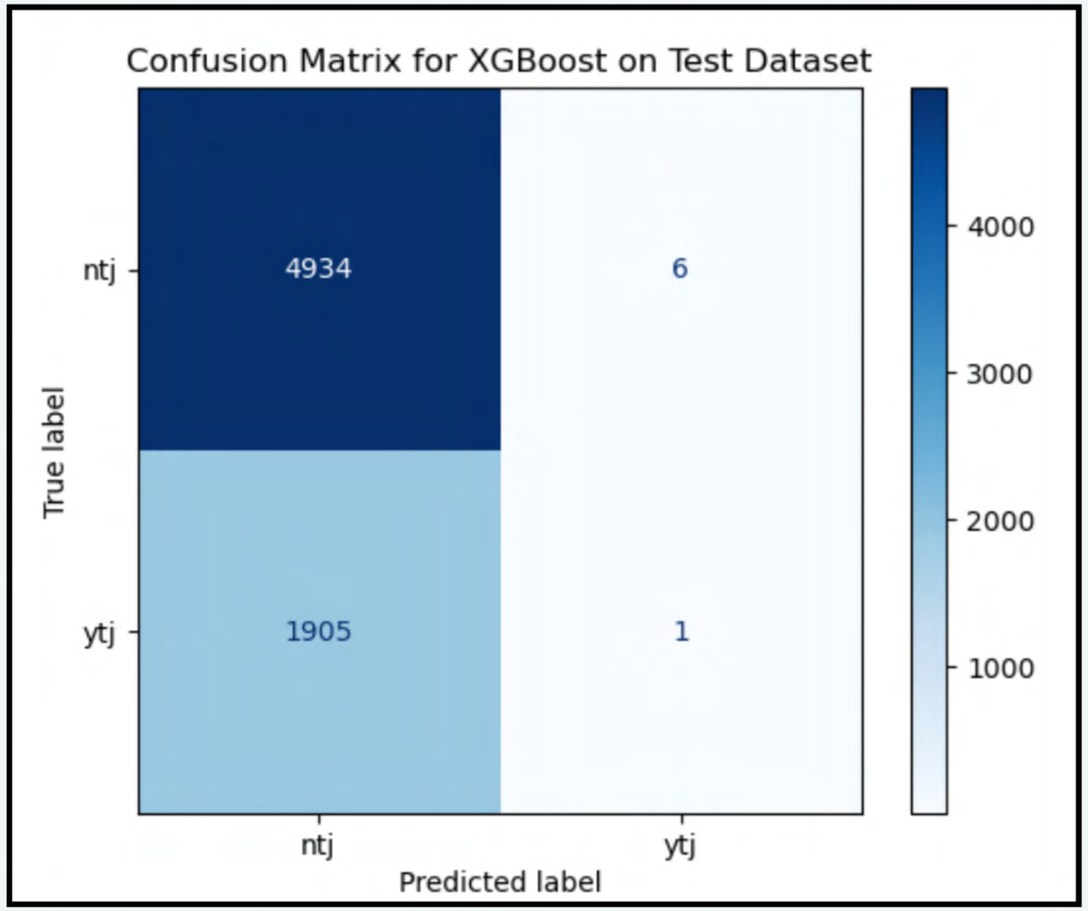
Classification Report for XGBoost on Test Dataset:

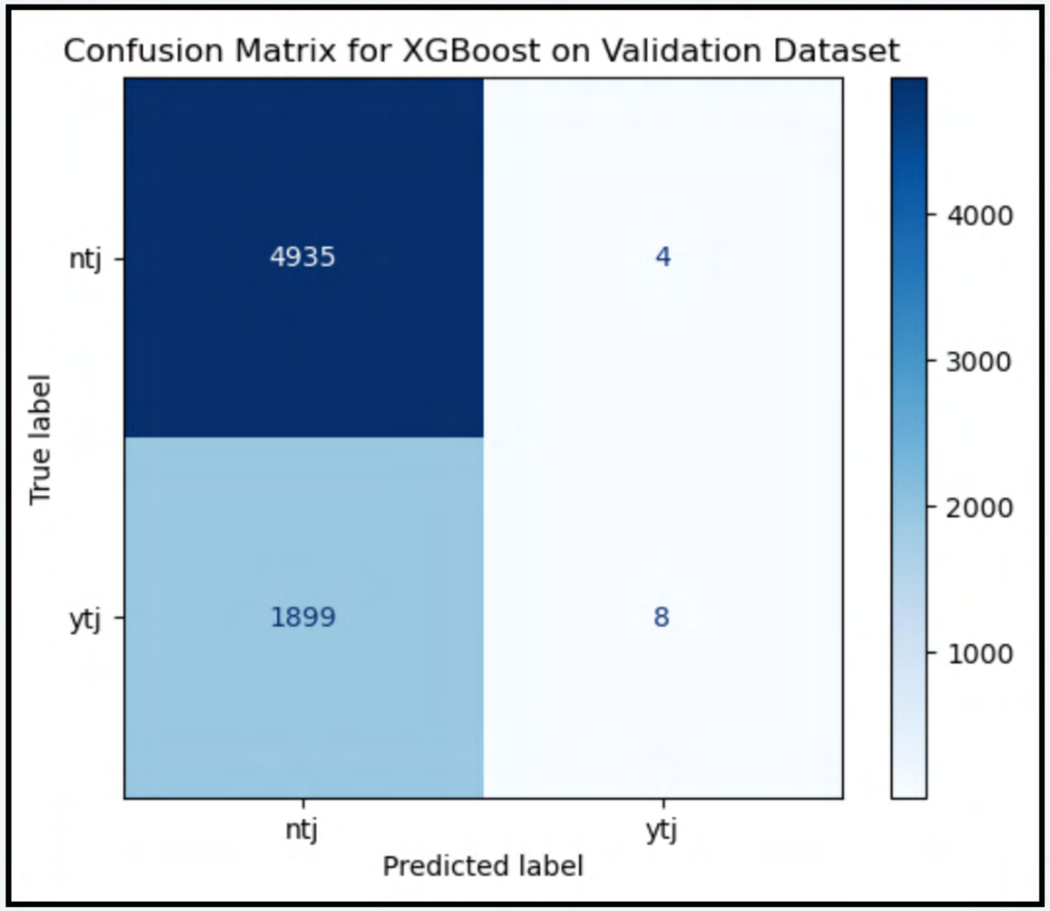
	precision	recall	f1-score	support
0	0.72	1.00	0.84	4940
1	0.14	0.00	0.00	1906
accuracy			0.72	6846
macro avg	0.43	0.50	0.42	6846
weighted avg	0.56	0.72	0.60	6846

Classification Report for XGBoost on Validation Dataset:

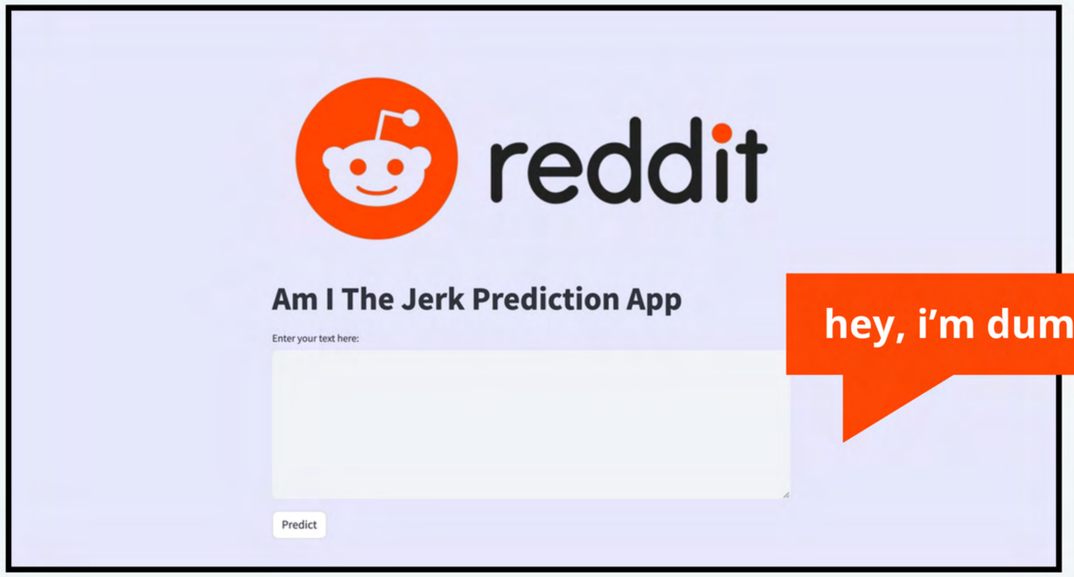
	precision	recall	f1-score	support
0	0.72	1.00	0.84	4939
1	0.67	0.00	0.01	1907
accuracy			0.72	6846
macro avg	0.69	0.50	0.42	6846
weighted avg	0.71	0.72	0.61	6846

XG Boost


Confusion Matrix for XGBoost on Test Dataset
ntj
ytj
4934
6
1905
1
True label
ntj
ytj
Predicted label
4000
3000
2000
1000


Confusion Matrix for XGBoost on Validation Dataset
ntj
ytj
4935
4
1899
8
True label
ntj
ytj
Predicted label
4000
3000
2000
1000

Model Deployment



hey, i'm dumb!

Random Forest	Yes 👍	0.61
Logistic Regression	Yes 👍	0.44
Decision Tree	Yes 👍	0.41
Support Vector Classifier	No 👶	0.61
Extreme Gradient Boost	Yes 👍	0.54

Result

Conclusion

- By leveraging machine learning models to automate analysis on Reddit's posts achieved significant insights into user interactions.
- By implementing ML models effectively classified user’s dilemmas as 'Yes' or 'No'

Future Works

- Advanced NLP techniques, such as BERT or GPT, could improve the understanding of context and nuances in the text.
- Exploring other ensemble methods and deep learning architectures may further enhance the accuracy and reliability of the classifications on real time basis.

Thank You!

Any Questions?