AI-Powered Moral Judgment Dilemmas of Reddit

1st Aafrin Shehnaz Mohamed Sulaiman
Master of Science in Data Analytics
San Jose State University
San Jose, United States
aafrinshehnaz.mohamedsulaiman@sjsu.edu
Student ID: (016801837)

2nd Aiswarya Raghavadesikan
Master of Science in Data Analytics
San Jose State University
San Jose, United States
aiswarya.raghavadesikan@sjsu.edu
Student ID: (014574599)

3rd Shivram Sriramulu
Master of Science in Data Analytics
San Jose State University
San Jose, United States
shivram.sriramulu@sjsu.edu
Student ID:(017439916)

4th Shreenithi Sivakumar
Master of Science in Data Analytics
San Jose State University
San Jose, United States
shreenithi.sivakumar@sjsu.edu
Student ID:(017473534)

5th Yogavarshni Ramachandran
Master of Science in Data Analytics
San Jose State University
San Jose, United States
yogavarshni.ramachandran@sjsu.edu
Student ID:(017404036)

***Abstract*—In the fast-paced world of Reddit, an online platform, r/AmItheJerk is a subreddit in which users share a story, and seek others to pass moral judgements. Commenters weigh in, asserting whether they think the poster was the jerk or not, using an acronym to indicate Yes The Jerk, or Not The Jerk: "YTJ", or "NTJ". This heavily depends on humans, which can cause delays depending on reader availability for replies and feedback. Employing Machine Learning and Natural Language Processing techniques, the text classification as "Yes" or "No" will not only help the Original Poster but also other readers to quickly understand their moral compass. The top-level comments from the post are taken into consideration for the project purpose. After labeling the data, further exploration of NLTK packages for lemmatization, TF-IDF feature extraction, and the Grid Search Cross-Validation-based hyperparameter tuned Random Forest classifier model rendered an overall accuracy of 72%, and an average weighted precision value of 74% on both the hyperparameter-tuned and baseline models, compared to other ML models - XGBoost, Support Vector Machine, Logistic Regression, and Decision Trees. The text classification outcome was deployed using the Streamlit app to demonstrate the efficiency of the models and ML algorithms.**

***Keywords*—machine learning, natural language processing, reddit, subreddit, text classification, lemmatization, hyperparameter, grid search cross validation, random forest, xg boost, support vector machine, logistic regression, decision tress.**

I. Introduction

Reddit is a dynamic platform where users share their ideas and discuss about their life. It also serves as a huge repository for data enthusiasts. [1] In the subreddit r/AmItheJerk, users post personal dilemmas and their life experiences to seek moral judgments from the community readers on whether their actions or scenarios is justified or not. Responses range from declaring someone as "Yes the Jerk" and "Not the Jerk." Our project aims to automate this decision making stage utilizing advanced machine learning (ML) and natural language processing (NLP) techniques. By analyzing the textual content from the submissions, we seek to deliver quick automated responses which shows the Reddit community's accord, overall providing instant feedback to the users. This study utilizes a comprehensive dataset extracted from Pushshift.io data dump of Reddit's submissions and comments, focusing on the application of various ML algorithms like Random Forest, Logistic Regression, and XG Boost to predict community judgments efficiently.

Disclaimer: Original subreddit r/aita is used throughout the project however it is concealed to maintain decorum and renamed as subreddit r/AmITheJerk.

II. Project Methodology

A. Data Collection

The dataset for this project is extracted from pushshift.io data dump, which is a vast collection of data available for many months for all subreddits and users. For this project, the focus was only on the Reddit submissions and comments from the subreddit AmITheJerk for the months of December 2023 - April 2024. The size of the raw data for the Subreddit's Post and Comments categories are approximately 125GB and 250 GB respectively. They are downloaded in the form of .zst_blocks. These are further converted to .zst files where the extracted post data size is now 80 MB and Comments data size is 1 GB. These .zst files were converted to get the final dataset in CSV format. After the files were merged and converted to CSV format, the final dataset size for Subreddit's Post is 235.2 MB and the Comments is 4.5 GB which was further considered for the pre-processing stage. Figure 1 shows the month wise file shape of the Subreddit's Submission (Post) category and Figure 2 shows the month wise file shape of the Subreddit's Comments category.

B. Architecture Diagram

Figure 3 is the architecture diagram that represents the flow of the project from the data collection to the model deployment. The chart was developed using lucid chart.

```
Shape of 2023_12 Submissions File: (32970, 111)
Shape of 2024_01 Submissions File: (31394, 111)
Shape of 2024_02 Submissions File: (28259, 111)
Shape of 2024_03 Submissions File: (27244, 111)
Shape of 2024_04 Submissions File: (26911, 112)
```

Fig. 1. File shape of Subreddit's Raw Submissions category.

```
Shape of 2023_12 Comments File: (1248172, 72)
Shape of 2024_01 Comments File: (1356231, 71)
Shape of 2024_02 Comments File: (1191114, 72)
Shape of 2024_03 Comments File: (1111939, 72)
Shape of 2024_04 Comments File: (1049400, 72)
```

Fig. 2. File shape of Subreddit's Raw Comments category.

C. Exploratory Data Analysis

In this section, exploratory data analysis was carried out using two different datasets: comments and submissions. The topic discussed here is about processing the comments dataset, with 72 columns. From a pioneering review of the metadata, the columns were reduced to the five most interesting ones. For labeling, the parent's comments are only considered. So for parent comments, we extracted the parent comments on the Reddit platform using a matching method through which the parent_id had to be equated with the link_id to be recognized as such. However, the primary point of our focus is the 'body' column of the comments dataset, because it carries the text. It is important since the column becomes the foundation for labeling, which is really crucial in other stages of this research.

The following two histograms depict the distribution of character and word counts for a corpus of textual data. Character Count Distribution is a histogram of the comment frequency per number of characters shown in Figure5. A right-skewed distribution with sharp kurtosis would entail most comments are quite short, but comments with a few unusually long comments are included. Word Count Distribution is to note also how the histogram graphs the frequency of comments against the number of words, shown in Figure 4. By the above two distributions, we will get to see the insight of the text entries verbosity in the dataset. Tokenization or padding might be necessary to handle such a wide range of comments when text processing algorithms are run over these data.

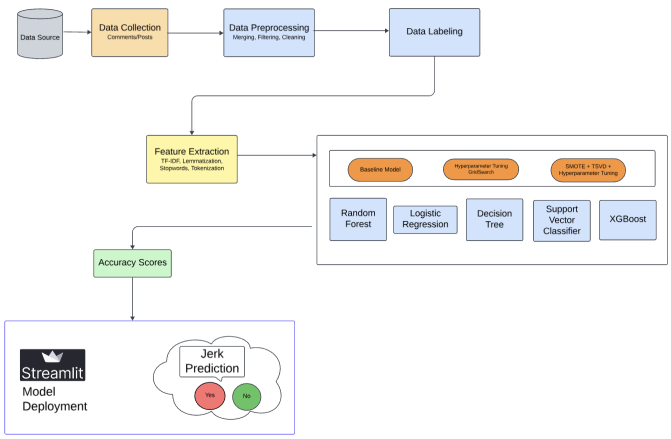

Fig. 3. Architecture Diagram

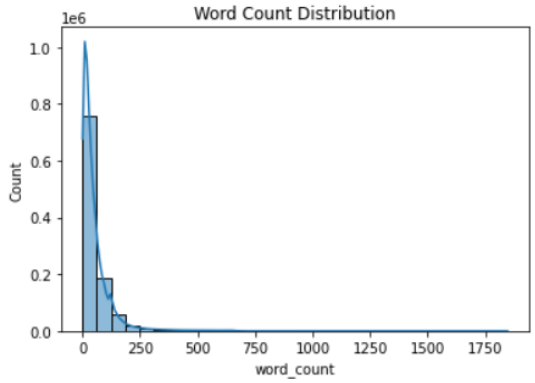

Fig. 4. Distribution of Word Count

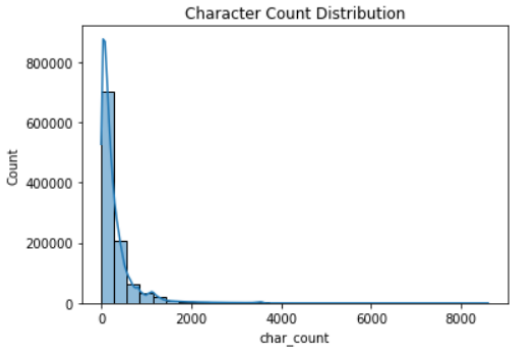

Fig. 5. Distribution of Character Count

After the removal of stop words shared among the datasets using the NLTK library, this cumulative analysis for the top 10 most frequently used substantive words reflects valuable information that could be used to interpret language constructions and present themes within the comments. The most frequently used word is "like," which, connected with other high-frequency verbs such as "would," and "get," denotes a dominance of characterized opinion in expressed language. Words such as "one," "want," "people," and "know" reflect the volume and general topics of personal experience or preference that are widely discussed by the users, shown in figure 6 This offers a more detailed analysis, free from the mundane stop words, to obtain a clearer understanding and insight for labeling.

The initial study was based on a dataset with 146,778 posts recorded and 112 attributes. After exhaustive exploratory analysis, six major columns, which were considered extremely important and thus required further research and feature selection, included 'author', 'num_comments', 'score', 'selftext', 'title', and 'upvote_ratio'.The two histograms shown represent the distributions of 'Number of Comments' and 'Score' of a

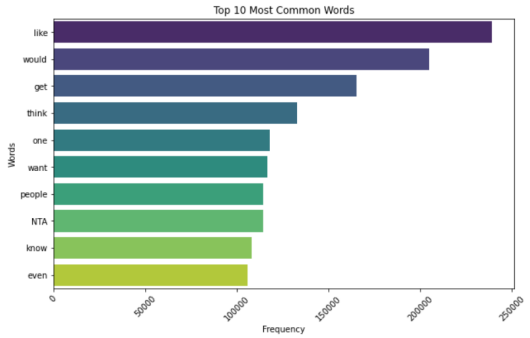

Fig. 6. Top 10 Most Common Words

collection of posts. Each very plainly shows a very strong skew to the right, which means most of the posts get a very small number of comments and very little scores, whereas a few of the posts get a lot more engagement or scores. For both 'Number of Comments' and 'Score', the histograms of most posts with values near zero comments or near-zero scores are shown with decreasing frequency as we get further away from these values. This indicated the post which has a high number of comments has a high score. Figure 7 and 8 shows the distribution of numerical features.

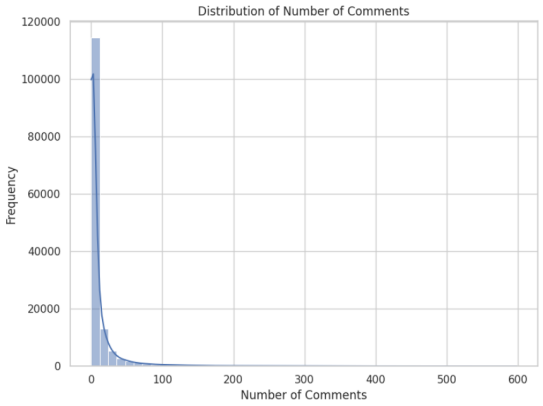

Fig. 7. Distribution of Number of Comments

As illustrated by the bar chart shown in figure 9, these are the top 10 authors in this dataset, with 'wdlthtBtheworstthing' being the most prolific, followed by 'Disastrous_Ship780' and 'Dizzy-Potato3557'. This therefore implies that in the spread, there is a high variance in content generation. Figure 10 illustrates the distribution of word counts for 'Selftext' and 'Title' within a dataset of posts: The distribution of 'Selftext Word Count' seems to indicate that most posts contain very little text, making the distribution tail across a long range of increasing word counts, with very few posts much longer than most. The 'Title Word Count' distribution is significantly more concentrated around the average length, meaning most titles are typically between five and 15 words, and the chance of longer lengths tails off very quickly. Such distributions focus on the typical brevity of titles relative to the high variability

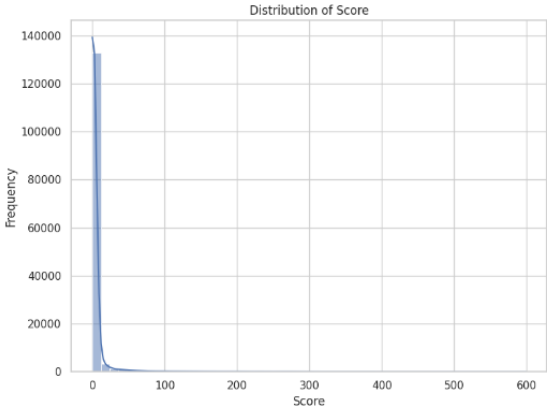

Fig. 8. Distribution of Scores

in body text length for posts, so they may function as useful intelligence for how to approach text processing and feature extraction for a model or analysis.

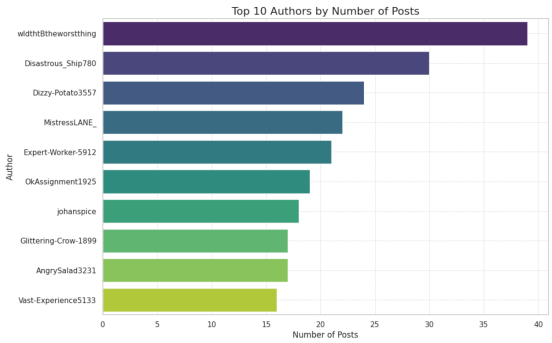

Fig. 9. Top 10 Authors by Post

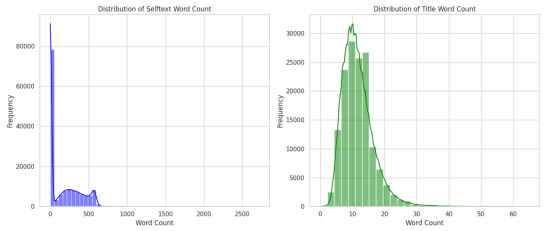

Fig. 10. Word Count Distribution of Title and Selftext

The bar chart is a pictorial representation of the word frequency of the top 20 words in the 'Selftext' of posts, and words such as "like," "said," and "would" appear most in that order. This is so because it has been the most important characteristic , shown in figure 11. Word clouds in this section provide a visual summary of the most common words found in the 'Selftext' and 'Title' fields of posts from a dataset. For example, in the word cloud of 'Selftext', words such as "need," "mom," or "work" reach out to personal-relational issues with almost tingling detail. In contrast, the word cloud of 'Title' discloses such words as "telling," "girlfriend," and "brother," showing that mostly, titles are about mentions related to rela-

tionships and important social interactions, shown in figure 12. By all numerical and categorical analyses of both comments and post dataset, labeling can proceed with the ”body” feature in the comments dataset, and for the descriptive feature, the ”selftext” feature gives rich contextual information which will be sufficient for the classification task of the subreddit.

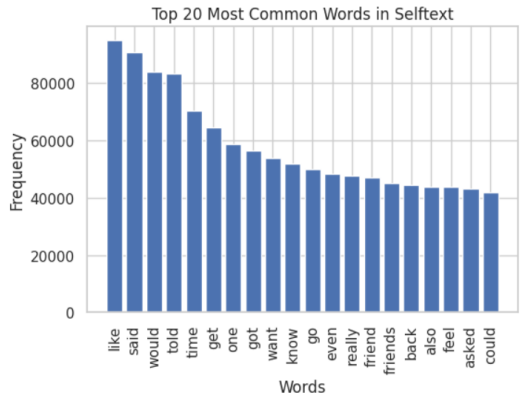

Fig. 11. Top 20 Most Common Words in Selftext

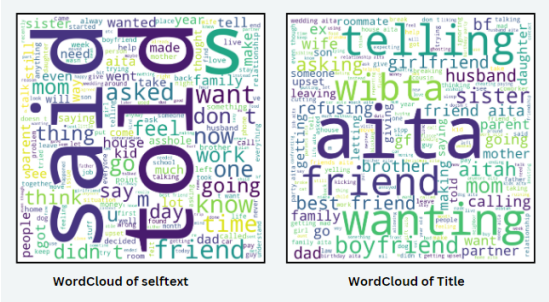

Fig. 12. WordCloud of Selftext and Title

D. Data Pre-processing

In the pre-processing stage, the initial EDA exploratory data analysis was performed to analyze the dataset. To understand the relevant columns is the primary step. Two sections were handled in this project - one is the Subreddit's submissions and the other one is the Subreddit's Comments. In this stage, each of the datasets were analyzed to see a way to merge them to proceed with the subsequent steps. Based on the parent_id and the link_id the 2 datasets were merged together to further consider them for the modeling purpose.

In the Subreddit's Comments dataset, data cleaning was performed to filter out the nulls or the NaN values, identified and removed comments like [removed] and [deleted] from the body column, dropped all the other numerical and irrelevant columns to retain the 'parent_id', 'link_id' and 'body' columns and filtered only the top level comments as it is the primary focus of this project. From the Subreddit's Post dataset, self_text and name_id were the columns retained. Figure 13 shows the columns filtered from the subreddit submissions dataset and Figure 14 shows the columns filtered from the subreddit comments dataset.

	selftext	name
0	[removed]	t3_187xb8a
1	[removed]	t3_187xbpt
2	In the world we live in today; I feel like pe...	t3_187xdf6
3	[removed]	t3_187xdk9
4	I have never been on here asking for help befo...	t3_187xeed

Fig. 13. Dataset Subreddit's Submission with filtered columns

	link_id	body
0	t3_187qp3s	I think many people make the mistake of thinki...
1	t3_187pj8a	NTA, HOWEVER, windfalls like that could easily...
2	t3_187n1wl	Nta. Sue him. I once heard a guy sued a family...
5	t3_187qgh4	Pretty sure I read this story a month or so ago.
8	t3_187n1wl	Yta \n\nI live on a road where the limit is 35...

Fig. 14. Dataset Subreddit's Comments with filtered columns

E. Data Labeling

As the project focuses on supervised machine learning, the labels are highly important for the text classification problem. In the comments dataset, only the post's top level comments are considered for further exploration whereas the nested comments are out of scope of the project. The idea behind this is that, only the top level comments are moderated by the subreddit's admins and community guidelines such that they contain the opinion label of the comment creators.

The machine learning models are further trained with the labels created in this phase. From the subreddit's Comments dataset, the 'body' column was used to filter out the tags and classify them into 2 groups for the final labelling. Tags such as 'nta', 'ywnbta', 'yntah', 'ynta' and 'nah' are labeled as 'NTJ' referring to 'Not The Jerk' and the tags such as 'eah', 'ehs', 'esh', 'yta', 'yat', 'yah', 'ywbta' , 'tah' , 'ah' and 'ytah' are labeled as 'YTJ' referring to 'You're The Jerk'. The comments were then aggregated and grouped based on the label and post_id columns, such that the label with the majority count was chosen as the final_label for each post. This ensured that the final comments dataset for labeling contained one label per post id for the dataset.

Figure 15 shows the dataset after the comments dataset was cleaned and labeled.

This comments dataset is then merged with the submissions dataset using the post_id column to get the combined final dataset for further analysis and model building.

Figure 16 shows the labeled submissions dataset after the submissions dataset was cleaned and merged with the comments dataset.

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

Fig. 15. Comments Dataset: Cleaned and Labeled

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

Fig. 16. Submissions Dataset: Cleaned and Merged

Figure 17 shows the majority count of 'NTJ' and 'YTJ' labels in the dataset. This is observed to know the weightage of the labels so as to proceed with the concurrent analysis.

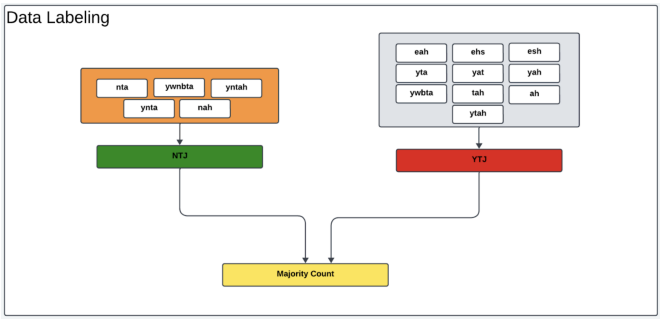

Fig. 17. Data Labeling: 'NTJ' and 'YTJ'

F. Feature Engineering

The NLTK packages are imported for the purpose of introducing the parts of speech (POS) to the context - like Noun, Verb, Adjective and Adverb. Removal of the stop words from the sentences was performed as stopwords are less influential and add even less value to the context. By doing so it makes the dataset less noisy. While performing the lemmatization technique, the words are normalized to be considered by the machine learning models.

As a final step in feature engineering, TF-IDF was performed as a feature extraction process where the Term Frequency identifies highly recurring words in the document and considers them as important while the Inverse Document Frequency technique identifies the most occurrences of the words and considers them as a least important feature. Equation 1 shows the formula for TF-IDF.

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t, D) \tag{1}$$

where:

- $\text{TF}(t, d)$ is the term frequency of term t in document d,
- $\text{IDF}(t, D)$ is the inverse document frequency of term t across a set of documents D.

G. Modeling Phase

1) Data Preparation for Modeling: After performing the data transformation, the dataset is split into 3 sets - train, validation and test sets. Here, the split ratio considered is 70:15:15 where the training set weighed 70%, the validation set weighed 15% and the testing set weighed 15%. Stratified sampling is performed to avoid any biases towards 'NTJ' which are in majority and preserve the original class proportions for the train, test and validation sets for better predictions.

The training set will be used to train the model, the test set will be used to hyper tune the models and the test data is set aside for the final stage of evaluating the models on completely unseen data.

Figure 18 shows the class distribution in each splits of the dataset - Original dataset, Train, Validation and Test sets.

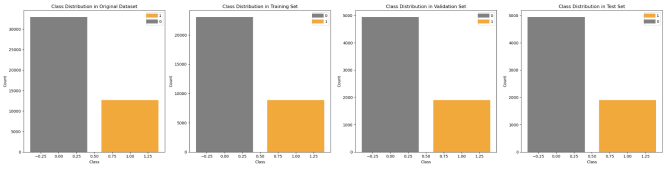

Fig. 18. Stratified Splitting of Dataset Into Train/Validation/Test Datasets.

2) Data Modeling: Before training the model, 2 more steps were performed - label encoding and column transformer. Label encoder was used to convert the labels (categorical variables) to numerical variables. Since we are handling the binary classification of "NTJ" referring to "No" and "YTJ" referring to "Yes", converting these categorical variables to numerical variables as 1 and 0 respectively is an important step. The column transformer helps in transforming the text column into TF-IDF generated features and dropping the other unspecified columns.

The 5 models that were chosen for this binary classification of text project are Random Forest (an ensemble model), Logistic Regression, Decision Tree, Support Vector Classifier and XGBoost which are the base models.

Random forest: - A widely used ensemble learning technique in machine learning for classification and regression tasks. During training, it generates numerous decision trees and combines their predictions to increase accuracy and avoid overfitting. In text classification, it can be used to categorize words based on the features retrieved from them. The mathematical equation 2 is

$$Y_{\text{RF}}(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x) \tag{2}$$

Where:

- $Y_{\text{RF}}(x)$ is the Random Forest prediction for input x.

- $T_b(x)$ is the prediction of the b-th decision tree.
- B is the number of trees in the forest.

Random forests is appreciable for inherently handling missing data without any special imputation techniques. However, the RF are more interpretable than some algorithms, the understanding of their inner workings always remains to be complex and additionally it is difficult to even understand the specific reasons behind those predictions.

Logistic Regression: - A statistical method for binary classification tasks in which the target variable has two possible outcomes. It models the likelihood that a given input belongs to a particular class using the logistic function. It usually reacts well when the characteristics and the target variable have a linear connection, or when a linear function can accurately capture the relationship. It is computationally efficient and easy to grasp. In datasets with complex interactions or highly nonlinear properties, its performance may be inferior than that of more advanced models. The mathematical equation 3 is given as

$$Y_{\text{LR}}(x) = \frac{1}{1 + e^{-(\beta_0 + \beta^T x)}} \tag{3}$$

Where:

- $Y_{\text{LR}}(x)$ is the probability of the positive class for input x.
- β_0, β are the model coefficients.

Training a logistic regression model is relatively fast and it is suitable for large datasets. Although the Logistic regression outputs a probability between 0 and 1, it might not directly represent the true probability of an instance that is belonging to a particular class.

Decision Tree: -A popular supervised learning method that may be used in both classification and regression applications. With internal nodes for feature tests, branches for test results, and leaf nodes for class labels or regression values, it resembles a tree. Because decision trees are easy to understand and evaluate, they are useful for exploratory data analysis. It works with both categorical and numerical data, and it frequently responds well to a variety of datasets. They can capture complex relationships between features and target variables while being simple to read and depict. On the other hand, decision trees often overfit the training set, especially if the tree is excessively large. The mathematical equation 4 is given as

$$Y_{\text{DT}}(x) = \text{leaf\_node}(x) \tag{4}$$

Where:

- $Y_{\text{DT}}(x)$ represents the Decision Tree output for input x.
- leaf_node(x) is the value at the leaf node for input x.

Decision trees has the ability to identify the important features which can yield best predictions. However its the downside is even when making small changes to the training data might lead to a significant changes in the decision tree structure.

Support Vector Machines: - SVM maps data to a high-dimensional feature space, allowing data points to be classified even when they are not otherwise linearly separable. After identifying a separator between the categories, the data is converted so that the separator can be drawn as a hyperplane. Following that, fresh data features can be utilized to predict which group a new record should belong to. The mathematical equation 5 is given as

$$Y_{\text{SVM}}(x) = \text{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b\right) \tag{5}$$

Where:

- $Y_{\text{SVM}}(x)$ is the SVM prediction for input x.
- α_i are the Lagrange multipliers from the solution of the dual problem.
- y_i are the labels of the training examples.
- x_i are the training examples.
- $K(x_i, x)$ is the kernel function used in the SVM.
- b is the bias term.
- sgn is the sign function.

SVMs is sound in achieving a higher accuracy on classification tasks, especially for small datasets and can handle high-dimensional data effectively. But they are sensitive to outliers causing performance impact.

Extreme Gradient Boost: - XG Boost uses decision trees as base models to build tree ensembles to improve prediction accuracies.For the purpose of text classification problem, the project is explored on XG Boost algorithm where the tree takes a set of features extracted from the text document as input and outputs a predicted class label. The algorithm then trains multiple trees on the training data and aggregates those predictions to produce the final prediction for each text document. The mathematical equation 6 is given as

$$Y_{\text{XGB}}(x) = \sum_{k=1}^{K} \alpha_k f_k(x) \tag{6}$$

Where:

- $Y_{\text{XGB}}(x)$ is the XGBoost prediction for input x.
- $f_k(x)$ is the prediction of the k-th boosted tree.
- α_k is the weight of the k-th tree.
- K is the total number of boosted trees.

Although XG Boost can handle complex non-linear relationships between features and the target variable its Hyperparamter Tunings can be a challenging aspect and it requires further experimentations.

Three different scenarios of modeling were executed - Baseline modeling, Hyperparameter Tuned modeling and SMOTE, TSVD based hyperparameter tuned modeling.

These scenarios were chosen for the following reasons:

Baseline: - It helps understand the performance of the models with the default parameters offered for each model.

Hyperparameter Tuned Model: - This helps in understanding how tuning the different parameters for each model will help in improving the overall performance of the model, using a 3 fold cross validation technique with the help of GridSearchCV.

Figure 19 shows the various parameters that were passed for each model for Hyperparameter Tuning using GridSearchCV.

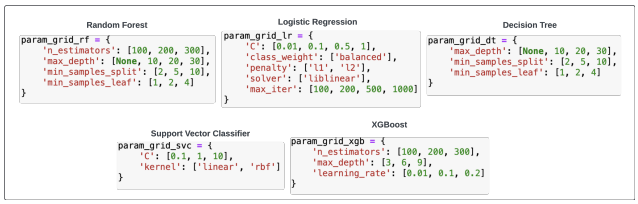

Fig. 19. Modeling: Hyperparameter tuning with GridSearchCV

Grid Searches are generally intensive and computationally expensive. They go over every sequence of the hyperparameter tuning to train the model on the training set and finally select the average best performing hyperparameter tuned combinations which can further be refined.

SMOTE Hyperparameter tuned model: - The dataset contains classes that are highly imbalanced. To handle this, SMOTE was employed to generate synthetic data for the class with lower occurrence. SMOTE refers to the Synthetic Minority Oversampling technique that helps to balance the dataset when one class is highly spread over the other class.

Figure 20 shows the class distribution before and after using SMOTE on the train dataset. Next, the top 1000 important

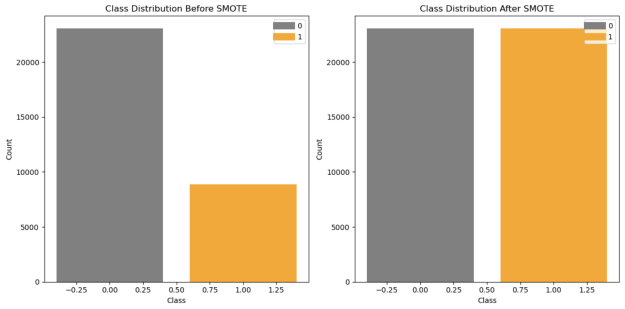

Fig. 20. Effect of SMOTE on Train Dataset

features were selected from 40k features that resulted from the TF-IDF vectorization. This will help reduce the load on the model building, by choosing only 1000 columns that have the most relevant features.

H. Evaluation Phase

For evaluation the models, the evaluation metrics that are chosen are accuracy, precision, recall and f1 score. **Accuracy:** - Accuracy is the ratio of the number of accurately classified data instances over the total number of data instances. Equation 7 as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total Samples}} \tag{7}$$

where:

- TP is the True Positives,
- TN is the True Negatives,
- FP is the False Positives
- FN is the False Negatives

Precision: Precision is the proportion of positive predictions over predicted true positives and false positives. This metric focuses on reducing the false positives. Equation 8 as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{8}$$

Being a classification project, the evaluate a model's efficiency can be more appropriately portrayed by the Precision metric. Figure 21 shows the average weighted precision for the test dataset for each model based on the different scenarios.

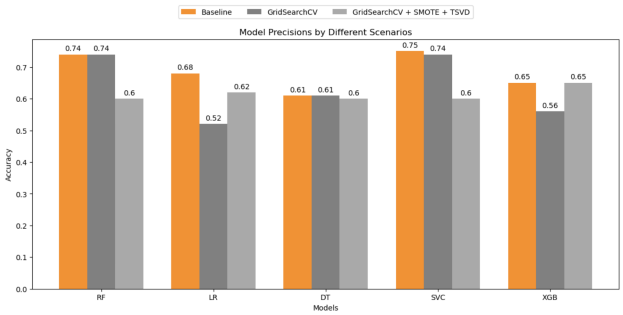

Fig. 21. Average Weighted Precision For Different Scenarios of Models.

From the Figure 21, it is observed that the Hyperparameter tuned models using SMOTE and TSVD did not perform well compared to the other models. This could be due to the fact that the sparse matrix features fully contribute to the improved precision, whereas a subset of these features leads to a decline in the precision.

The precision of each model is evaluated in every scenario and the best performing scenario for each model is chosen for the model deployment. The Random Forest, Decision Tree and SVC performed better during the Hyperparameter tuned model, whereas, the Logistic Regression and the XGBoost rendered better outcomes during the Baseline model.

Recall: True Positive Rate (TPR) is the measure of the actual positive cases that are correctly predicted by the model. Recall metric is also called as sensitivity metric. This metric focuses on reducing the false negatives.Equation 9 as follows:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{9}$$

F1 Score: The harmonic mean of Precision and Recall as it brings in a balance between the model's ability to identify true positives (recall) and avoid false positives (precision). It is a good metric when the classes are imbalance as it considers both the precision and recall. Equation 10 as follows:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{10}$$

Confusion Matrix: To precisely visualize the capability of the model to predict the labels to that of the actual labels. 4 quadrants are split to denote TP, TN, FP, FN labels across the predicted labels (x-axis) and actual labels (y-axis)

The confusion matrix is shown for the best models chosen. Figure 22shows the confusion matrix for the Hyperparameter tuned model on Random Forest algorithm.

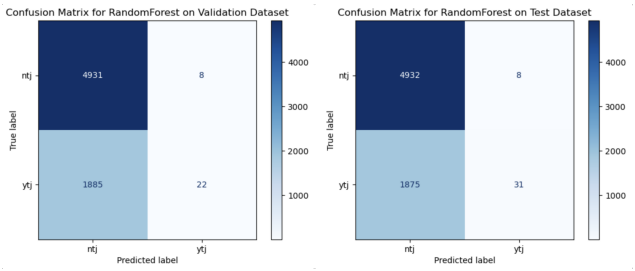

Fig. 22. Confusion Matrix: Hyperparameter Tuned - Random Forest

Figure 23 shows the confusion matrix for the Baseline model on Logistic Regression algorithm

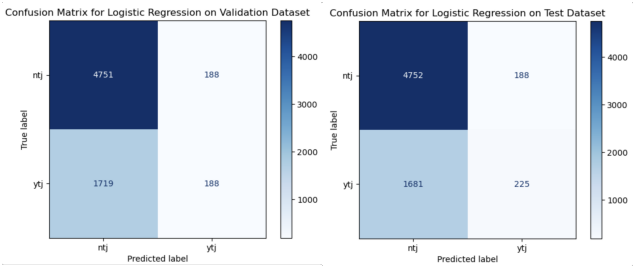

Fig. 23. Confusion Matrix: Baseline - Logistic Regression

Figure 24 shows the confusion matrix for the Hyperparameter tuned model on Decision Tree algorithm

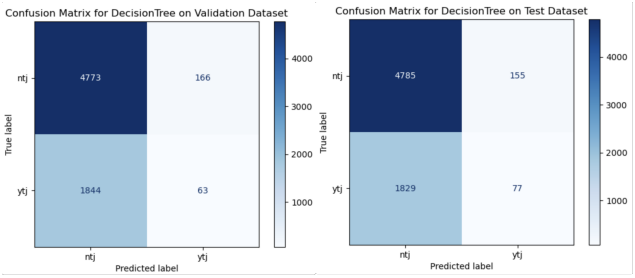

Fig. 24. Confusion Matrix: Hyperparameter Tuned - Decision Tree

Figure 25 shows the confusion matrix for the Hyperparameter tuned model on SVM algorithm

Figure 26 shows the confusion matrix for the Hyperparameter tuned model on XG Boost algorithm

From the confusion matrices, it can be observerd that all the models are struggling to rightly classify the 'Yes' class when compared to the 'No' class. One possible reason for this could be that the models are biased due to the imbalanced classes in the original dataset.

The Table I shows the performance scores for the Baseline models across all the 5 machine learning algorithms. The Accuracy of the Validation set is captured for all the Machine Learning Algorithm models. While the Precision, Recall and

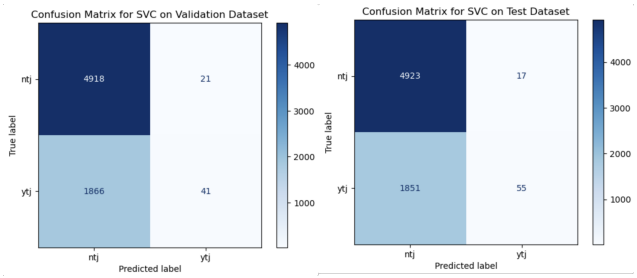

Fig. 25. Confusion Matrix: Hyperparameter Tuned - SVM

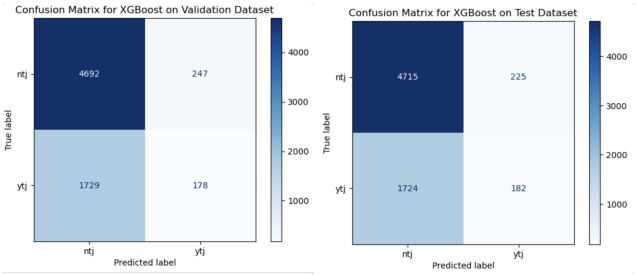

Fig. 26. Confusion Matrix: Hyperparameter Tuned - XG Boost

F1 score for both Validation and Test sets are captured for both the class labels.

TABLE I
BASELINE MODEL PERFORMANCE SUMMARY

ML Algo.	Acc. Val	Class	Precision		Recall		F1 Score	
			Val	Test	Val	Test	Val	Test
RF	0.72	0	0.72	0.72	1	1	0.84	0.84
		1	0.78	0.78	0.01	0.02	0.02	0.03
LR	0.72	0	0.73	0.74	0.96	0.96	0.83	0.84
		1	0.5	0.54	0.1	0.12	0.16	0.19
DT	0.61	0	0.73	0.73	0.73	0.74	0.73	0.74
		1	0.3	0.31	0.3	0.3	0.3	0.3
SVM	0.72	0	0.72	0.73	1	1	0.84	0.84
		1	0.66	0.76	0.02	0.03	0.04	0.06
XGB	0.71	0	0.73	0.73	0.95	0.95	0.83	0.83
		1	0.42	0.45	0.09	0.1	0.15	0.16

The Table II shows the performance scores for the GridSearchCV hyperparameter tuned models across all the 5 machine learning algorithms. The Accuracy of the Validation set is captured for all the Machine Learning Algorithm models. While the Precision, Recall and F1 score for both Validation and Test sets are captured for both the class labels.

The Table III shows the performance scores for the SMOTE+ TSVD GridSearchCV hyperparameter tuned models across all the 5 machine learning algorithms. The Accuracy of the Validation set is captured for all the Machine Learning Algorithm models. While the Precision, Recall and F1 score

TABLE II
HYPERPARAMETER MODEL PERFORMANCE SUMMARY

ML Algo.	Acc. Val	Class	Precision		Recall		F1 Score	
			Val	Test	Val	Test	Val	Test
RF	0.72	0	0.72	0.72	1	1	0.84	0.84
		1	0.73	0.79	0.01	0.02	0.02	0.03
LR	0.72	0	0.72	0.72	1	1	0.84	0.84
		1	0	0	0.1	0	0	0
DT	0.71	0	0.72	0.72	0.97	0.97	0.83	0.83
		1	0.28	0.33	0.03	0.04	0.06	0.07
SVM	0.72	0	0.72	0.73	1	1	0.84	0.84
		1	0.66	0.76	0.02	0.03	0.04	0.06
XGB	0.72	0	0.72	0.72	1	1	0.84	0.84
		1	0.67	0.14	0	0	0.01	0

for both Validation and Test sets are captured for both the class labels.

TABLE III
SMOTE, TSVD APPLIED HYPERPARAMETER TUNED MODEL PERFORMANCE SUMMARY

ML Algo.	Acc. Val	Class	Precision		Recall		F1 Score	
			Val	Test	Val	Test	Val	Test
RF	0.69	0	0.73	0.72	0.90	0.98	0.81	0.83
		1	0.37	0.28	0.16	0.20	0.22	0.30
LR	0.59	0	0.78	0.74	0.61	0.50	0.68	0.60
		1	0.35	0.30	0.56	0.55	0.43	0.39
DT	0.57	0	0.73	0.72	0.62	0.64	0.67	0.68
		1	0.30	0.28	0.41	0.36	0.34	0.32
SVM	0.71	0	0.74	0.72	0.90	0.81	0.82	0.76
		1	0.44	0.28	0.20	0.19	0.27	0.23
XGB	0.69	0	0.75	0.72	0.85	0.93	0.80	0.81
		1	0.40	0.29	0.26	0.07	0.31	0.12

From all the 3 phases, Random Forest has yielded a consistent outcome by accurately predicting the text classification as “Yes” and “No” for the given user text input. However, the GridSearchCV Hyperparameter tuned model is chosen to be the best as it rendered a of 71% for Decision Tree algorithm and rest of all the algorithms holds 72%. As the Random Forest model works best on the unseen data, it shows more stability in handling the user text to render the most accurate text classification.

The Table IV shows the consistent behavior of the Random Forest algorithm to show 74% average weighted precision value in Test sets for Baseline and Hyperparameter tuned model. Although, Logistic Regression has shown poor performance out of all the models, yet, it holds 52% of average weight that involves a benefit of doubt to its inability to perform well on high dimensional spaces to cover the large feature space.

TABLE IV
AVERAGE WEIGHTED PRECISION SCORES OF ALL MODELS AND ALGORITHMS

Model	ML Alg	Weighted Precision	
		Validation	Test
Baseline	Random Forest	0.74	0.74
	Logistic Regression	0.67	0.68
	Decision Tree	0.61	0.61
	Support Vector Classifier	0.69	0.75
	XGBoost	0.64	0.65
HP	Random Forest	0.73	0.74
	Logistic Regression	0.52	0.52
	Decision Tree	0.60	0.61
	Support Vector Classifier	0.71	0.74
	XGBoost	0.71	0.56
HP SMOTE TSVD	Random Forest	0.63	0.60
	Logistic Regression	0.66	0.62
	Decision Tree	0.61	0.60
	Support Vector Classifier	0.66	0.60
	XGBoost	0.60	0.65

III. DEPLOYMENT PHASE

Streamlit, an open-source app, runs on python scripts to transform the script file to an attractive user interface for showcasing the ability of the built models. The user can experiment with their desired input text to visualize the models’ prediction ability on deployed Streamlit ’Jerk Prediction App’.

While building the ML models, joblib files are generated appropriately for the models and the transformations. The joblib files for label encoder and column transformer ensure that the text entered by the user in the web application is processed consistently like how the model’s training dataset was processed. The 5 best models that were chosen previously will be deployed using their respective joblib files. This gives the user the opportunity to see the outputs for all the models from this project and make their decision based on the responses.

Currently, the Streamlit application is installed and hosted on a local machine. The app.py Python script is created by defining the desired app layout, Streamlit components, and functionalities, including the necessary joblib files. Once this is done, the Streamlit app will be ready to launch for deployment.

By running the ‘streamlit run app.py’ from the local terminal, the web UI can be accessed. Figure27 shows the deployed user web interface.

Figure28 demonstrates the deployment segment from the best chosen models.

IV. CONCLUSION

These machine learning models have substantially improved the project by the analysis and prediction of community user judgments on Reddit’s r/AmItheJerk subreddit. The models gave good metrics, getting to satisfactory levels in terms of accuracy, precision, recall, and F1 score against which

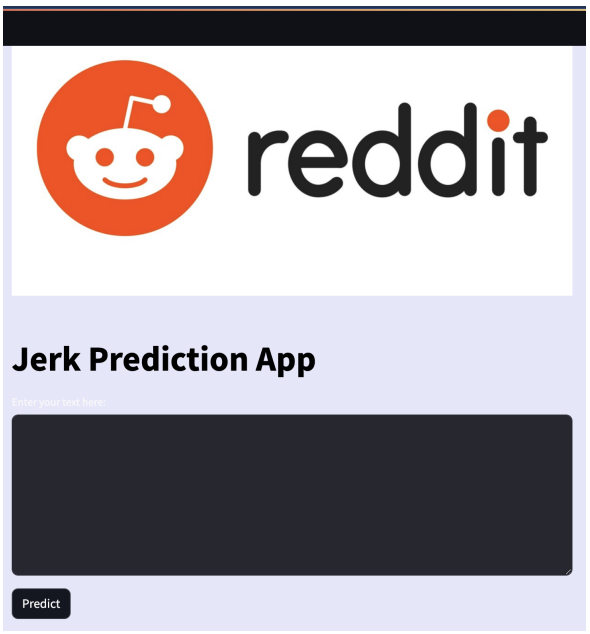

Fig. 27. Web Application UI for Model Deployment using Streamlit.

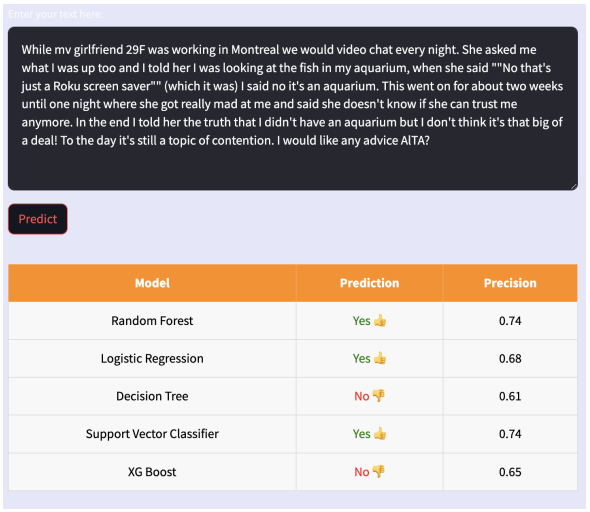

Fig. 28. Model Prediction using Deployed Models in Streamlit.

they presented the simulation of human judgment processes. Predicting 'No' classes were much easier for all models than the 'Yes' class. However, advanced challenges still exist as the system can improve its capability to find the real nuances and context of ideas behind human judgments in moral and ethical judgments in general. Overall, this project has proved the possibility of the use of ML and NLP methodologies in the simulation of social media responses and possible substitution of time-consuming human activity.

V. Future Scope

Moving forward, a more prudent step would be to implement new, advanced NLP techniques such as BERT or GPT, which will enable the model to empathize with the subtle linguistic and contextual ideas in the user submissions. Other ensemble models and deep learning methods can we investigated to get better precision in classification, especially for advanced and complex scenarios. Moreover, increasing the number of posts and responses in the data and class distribution in the training data would be useful and necessary for the training of more robust and unbiased models. Lastly, actual real-time use and feedback by users will help in further improvements and user satisfaction with automated responses

References

[1] M. R. Jamnik and D. J. Lane, "The Use of Reddit as an Inexpensive Source for High-Quality Data," *Practical Assessment, Research, and Evaluation*, vol. 22, no. 1, p. 5, 2017. DOI: 10.7275/j18t-c009. URL: https://doi.org/10.7275/j18t-c009

[2] N. Proferes, N. Jones, S. Gilbert, C. Fiesler, & M. Zimmer, "Studying Reddit: A Systematic Overview of Disciplines, Approaches, Methods, and Ethics," *Social Media + Society*, vol. 7, no. 2, 2021. DOI: 10.1177/20563051211019004. URL: https://doi.org/10.1177/20563051211019004

[3] K.E. Anderson, "Ask me anything: what is Reddit?," *Library Hi Tech News*, vol. 32, no. 5, pp. 8-11, 2015. DOI: 10.1108/LHTN-03-2015-0018. URL: https://doi.org/10.1108/LHTN-03-2015-0018

[4] A.N. Medvedev, R. Lambiotte, J.C. Delvenne, "The Anatomy of Reddit: An Overview of Academic Research," in F. Ghanbarnejad, R. Saha Roy, F. Karimi, J.C. Delvenne, B. Mitra (eds), *Dynamics On and Of Complex Networks III. DOOCN 2017. Springer Proceedings in Complexity.* Springer, Cham, 2019. DOI: 10.1007/978-3-030-14683-2_9. URL: https://doi.org/10.1007/978-3-030-14683-2_9

[5] E. Gilbert, "Widespread underprovision on Reddit," in *Proceedings of the 2013 Conference on Computer Supported Cooperative Work (CSCW '13)*, Association for Computing Machinery, New York, NY, USA, pp. 803-808, 2013. DOI: 10.1145/2441776.2441866. URL: https://doi.org/10.1145/2441776.2441866

[6] T. Weninger, X. A. Zhu, and J. Han, "An exploration of discussion threads in social news sites: a case study of the Reddit community," in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '13)*, Association for Computing Machinery, New York, NY, USA, pp. 579-583, 2013. DOI: 10.1145/2492517.2492646. URL: https://doi.org/10.1145/2492517.2492646

[7] A. Pradhan, S. Prabhu, K. Chadaga, S. Sengupta, and G. Nath, "Supervised Learning Models for the Preliminary Detection of COVID-19 in Patients Using Demographic and Epidemiological Parameters," *Information (Switzerland)*, 2022. DOI: 10.3390/info13070330. URL: https://doi.org/10.3390/info13070330