Classifying Reddit Mental-Health Posts Using TF-IDF Features and Classical Machine Learning Models

1. Introduction

This project explores the automatic classification of mental-health–related Reddit posts using traditional machine learning algorithms applied to linguistic features (TF-IDF, readability, word count, unique words, etc.). The data consist of user posts from ten mental-health subreddits—addiction, alcoholism, anxiety, autism, bipolarreddit, bpd, ptsd, schizophrenia, socialanxiety and suicidewatch—each chosen to represent a different mental-health condition or concern. The aim is to predict the originating subreddit (as a proxy for the underlying condition) from a set of numeric features derived from the post text. The purpose of this project is to compare mental health-focused communities on social media and to analyze any significant linguistic features present in those communities.

The problem is challenging for several reasons. First, the dataset is highly imbalanced: some subreddits have over twenty thousand posts while others have fewer than two thousand. Second, the feature space is high-dimensional and sparse, because it is based on TF-IDF representations of the text. Third, the classes are not cleanly separable conceptually as people with different diagnoses may discuss similar themes which increases overlap in feature space.

To address these challenges, we have proceeded with technical analysis through several stages. The raw CSV files are loaded and combined, the resulting dataset is cleaned and inspected, and a set of engineered features and feature-selection steps are applied. The data are then split into train, validation, and test sets using stratified sampling. Three main models are evaluated: a baseline multinomial Logistic Regression model, a tuned Logistic Regression model with hyperparameter search, a Linear SVM wrapped in probability calibration, and a Random Forest classifier with parameters chosen specifically for sparse, imbalanced data. Their performance is compared using accuracy, detailed classification reports, confusion matrices, and ROC/AUC metrics.

2. Data and Pre-Processing

2.1 Data sources and loading

The analysis begins in Google Colab by mounting Google Drive and listing the contents of a project directory containing ten CSV files with names such as addiction_post_features_tfidf_256.csv, anxiety_post_features_tfidf_256.csv, and so on. Each file contains numeric features for posts from one subreddit, including 256-dimensional TF-IDF features and potentially additional readability or linguistic measures. The files are read and concatenated into a single DataFrame full_df.

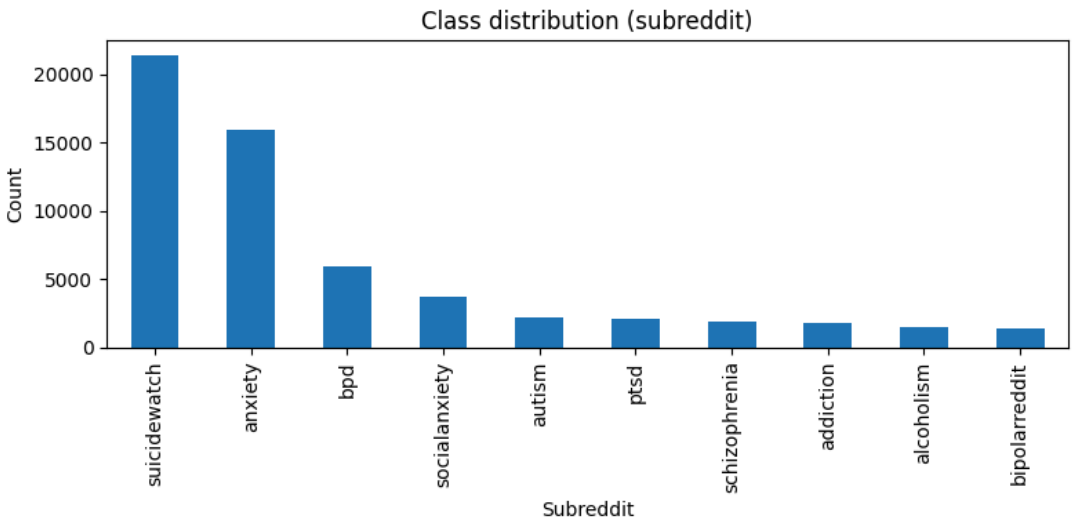

Figure 1. Class distribution

After concatenation, the combined dataset contains 57,727 posts. The class distribution is markedly imbalanced: for example, the suicidewatch subreddit contributes approximately 21,410 posts, anxiety contributes about 15,896, and bpd about 5,973, whereas bipolarreddit and schizophrenia contribute only around 1,368 and 1,683 posts, respectively; the comparison between classes can be seen in Figure 1. This uneven distribution can skew the resulting models and needs to be accounted for when training the models.

2.2 Data cleaning

The vast majority of features are numeric, as expected from TF-IDF and related scores. Where missing values occur in numeric columns, they are filled with zeros. This choice is appropriate for TF-IDF features, where a zero naturally corresponds to "term absent," and prevents problems for algorithms that do not handle NaNs.

The target label is the subreddit column. We did not observe substantial missingness in this column, so all rows are retained.

To explore class imbalance, the value counts of subreddit are plotted as a bar chart. This visualization confirms that posts from suicidewatch and anxiety dominate, with smaller but still substantial contributions from bpd, social anxiety, and ptsd, and relatively few examples from autism, bipolar, schizophrenia, addiction, and alcoholism.

2.3 Feature engineering and feature selection

From the full set of columns, the label column is excluded and all remaining numeric columns are treated as candidate features. These encompass the TF-IDF features as well as any additional numeric attributes.

A simple but informative engineered feature is introduced: for each post, the sum of all TF-IDF features is computed to form tfidf_sum. This scalar represents the overall TF-IDF magnitude of a post, loosely related to its length and lexical intensity. To capture non-linear effects in a way that tree-based models can exploit, tfidf_sum is then discretized into four quartiles using pd.qcut and stored as tfidf_sum_bin, an integer from 0 to 3.

Next, constant features are removed using sklearn.feature_selection.VarianceThreshold. Features with exactly zero variance across the dataset are deleted because they cannot help discriminate between classes. This step reduces dimensionality and improves computational efficiency without sacrificing information. The resulting matrix contains several hundred numeric features (roughly in the mid-300s), which serve as the final input representation X. The label vector y is taken directly from the subreddit column.

2.4 Data Representation

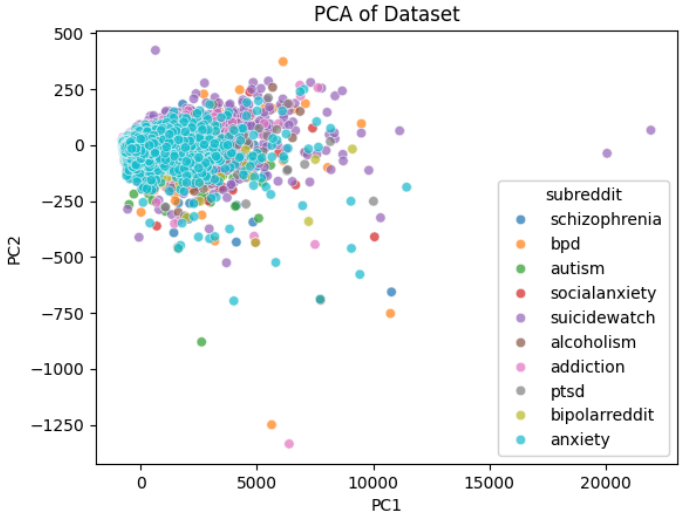

Figure 2. Principal Component Comparison

Given that the data has an extremely high dimensionality, it is difficult to effectively represent the data in a meaningful way. Principal component analysis (PCA) is the process of reducing the dimensionality of a dataset while maintaining the overall variance between entries. Figure 2 is a scatter plot of the PCA performed on linguistic data and shows that there is relatively little variance between communities and between posts in post language.

2.5 Train–validation–test split

To support fair model selection and evaluation, the dataset is partitioned into training, validation, and test sets using train_test_split with stratification on y. Approximately 70% of the data are allocated to training, with the remaining 30% split evenly between validation and test. The use of stratification ensures that each split preserves the original class proportions, which is crucial when minority classes are small.

3. Models and Training Procedures

3.1 Baseline multinomial Logistic Regression

Logistic Regression is a linear classification model that estimates the probability of each class using a softmax function applied to a weighted sum of the input features. In multinomial form, it learns a separate linear decision boundary for every class and chooses the class with the highest predicted probability. Because TF-IDF text data is high-dimensional and often linearly separable, logistic regression is an effective and computationally efficient model for document classification.

The Advantages of this model include interpretability (coefficients show which features influence predictions), fast training on high-dimensional sparse data, robustness to multicollinearity, and good generalization when combined with regularization.

Disadvantages include the inability to capture non-linear relationships, sensitivity to irrelevant features, and weaker performance compared to tree-based or neural models when interactions or non-linear feature effects are important.

The first model is a straightforward multinomial Logistic Regression, fitted with the SAGA solver and class_weight="balanced" to address class imbalance. This baseline uses default regularization (C = 1.0) and is trained directly on the training features. When evaluated, this baseline performs poorly: the validation accuracy is roughly 0.34 and the test accuracy is similarly around 0.34. The macro-averaged precision, recall, and F1-score on the test set are approximately 0.46, 0.34, and 0.35, respectively, as seen in Figure 3. The classification report shows low macro-averaged precision and recall, indicating that the untuned model fails to make effective use of the high-dimensional feature space. These results justify the need for more careful tuning and more powerful models.

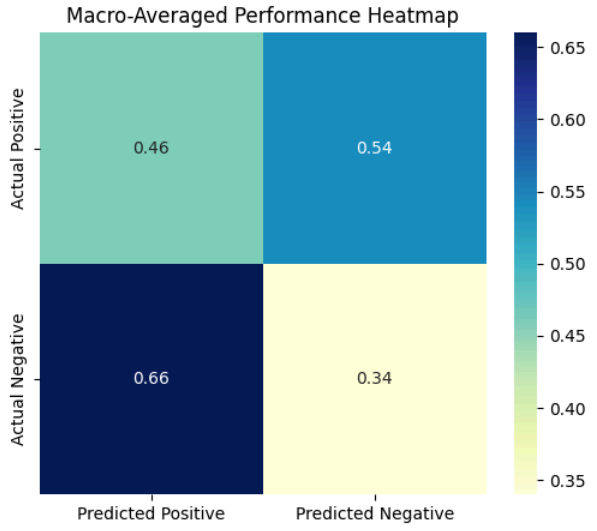

Figure 3. Resulting performance heatmap of the logistic regression

3.2 Tuned Logistic Regression with cross-validation and ROC/AUC

To improve on the baseline, a more systematic Logistic Regression model is trained and tuned via cross-validation. The base estimator remains multinomial Logistic Regression with the SAGA solver and balanced class weights, but the regularization strength parameter C is now selected using GridSearchCV. A grid of C values ({0.1, 1.0, 3.0, 10.0}) is explored using five-fold StratifiedKFold cross-validation on the training set. The scoring metric is macro-averaged AUC under the one-vs-rest scheme, which treats each class as positive against all others and then averages AUC across classes.

The grid search identifies an optimal C of 10.0. This tuned model is then used for all further Logistic Regression evaluations. On the test set, the tuned Logistic Regression model produces a confusion matrix with clear diagonal structure and relatively few gross misclassifications. The micro-average ROC curve lies substantially above the diagonal; its AUC is approximately 0.75. The macro-averaged AUC values are about 0.74 on the training set, 0.73 on the validation set, and 0.74 on the test set, suggesting that the model generalizes well and does not overfit substantially.

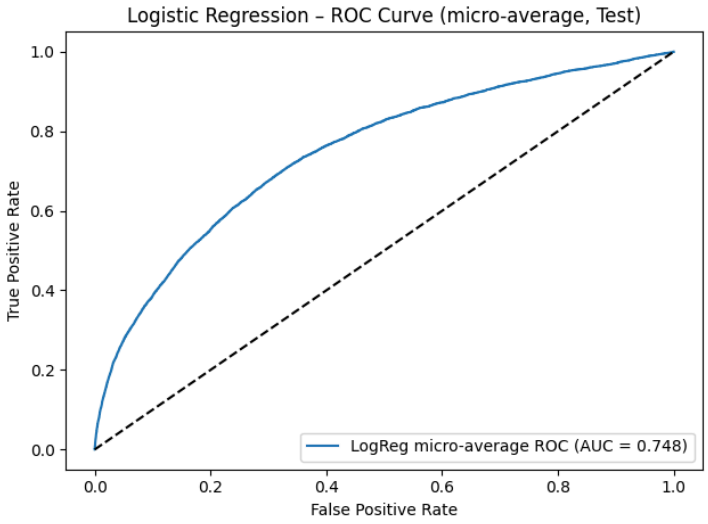

Figure 4. ROC curve of the tuned logistic regression model

These results show that Logistic Regression is capable of capturing strong linear structure in the TF-IDF feature space when appropriately regularized and tuned.

3.3 Linear SVM with probability calibration

A Support Vector Machine (SVM) is a discriminative classifier that identifies the hyperplane that maximizes the margin between classes in a high-dimensional feature space. The Linear SVM, specifically, uses a linear decision boundary and is well suited for sparse TF-IDF text representations, where the number of features is large relative to the number of samples.

The primary advantage of SVMs is their ability to generalize well by maximizing the margin, which often leads to strong performance even when classes overlap. Linear SVMs are also computationally efficient on high-dimensional data, scale well to tens of thousands of features, and are robust to irrelevant features.

However, SVMs have disadvantages: they do not directly output calibrated probabilities, they struggle with severe class imbalance unless class weights are adjusted, and they cannot capture non-linear features.

3.4 Hyperparameter Tuning and Cross-Validation

The base classifier used was LinearSVC, which does not support probability outputs by default. To address this, it was wrapped in a CalibratedClassifierCV using sigmoid (Platt) scaling with 3-fold internal cross-validation. Although Linear SVMs can be tuned using grid search over parameters such as the regularization constant C, in this project the primary goal was to obtain calibrated probability estimates for ROC/AUC computation. The calibration step itself trains multiple SVMs during cross-validation and therefore implicitly performs a form of model selection by averaging across folds. Class weights were set to "balanced" to address the substantial class imbalance in the dataset. Future work could include explicit tuning of C via StratifiedKFold cross-validation; however, computational cost was high due to the large dataset and repeated model fitting during calibration.

The calibrated SVM was trained on the same training features as Logistic Regression and evaluated on the validation and test sets. The validation accuracy is approximately 0.75 and the test accuracy is around 0.74. The macro-averaged precision, recall, and F1-score on the test set are approximately 0.73, 0.74, and 0.72, respectively, as can be seen in Figure 5. The classification report reveals that the SVM achieves strong precision and recall for the dominant classes (anxiety and suicidewatch) as well as respectable performance on several minority classes. For example, anxiety posts obtain high recall, indicating that the model correctly recognizes the majority of anxious posts, while suicidewatch posts show high recall and reasonably high precision.

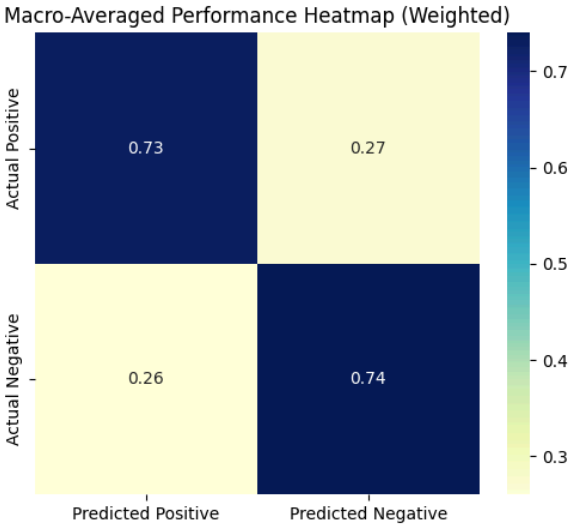

Figure 5. Resulting performance heatmap of the linear SVM

Calibrated probabilities are also examined for interpretability. For selected test examples, the top predicted subreddits and their probabilities are printed. In one example, the true label is autism, and the model assigns the highest probability (around 0.49) to autism, followed by smaller but non-negligible probabilities for schizophrenia, anxiety, bpd, and suicidewatch. In another example, the true label is suicidewatch, and the model correctly assigns a very high probability to suicidewatch (over 0.70), with far lower probabilities for other classes. These examples demonstrate that the calibrated SVM not only provides accurate point predictions but also meaningful uncertainty estimates.

Example Probability Outputs:					
True Subreddit:	Autism	True Subreddit:	Suicidewatch	True Subreddit:	Anxiety
Top Predicted:		Top Predicted:		Top Predicted:	
Autism	48.7%	Suicidewatch	73.6%	Anxiety	76.9%
Schizophrenia	19.3%	BPD	20.9%	Suicidewatch	11.4%
Anxiety	14.1%	Anxiety	3.1%	Social Anxiety	5.1%
BPD	5.7%	Social Anxiety	0.8%	Addiction	2.0%
Suicidewatch	5.2%	PTSD	0.4%	Autism	1.8%

Table 1. SVM Example probability outputs

Overall, the calibrated Linear SVM outperforms even the tuned Logistic Regression in terms of accuracy and macro-averaged F-scores. It serves as the strongest of the linear models evaluated.

3.5 Random Forest classifier

A Random Forest classifier was selected as the primary non-logistic model because it is capable of capturing complex, non-linear interactions among features that linear models such as Logistic Regression and SVM cannot easily represent. A Random Forest is a method that constructs a large number of decision trees, each trained on a bootstrap sample of the data and considering only a random subset of features at each split. The final prediction is obtained through majority voting across all trees. This design reduces variance, stabilizes predictions, and allows the model to learn heterogeneous relationships present in the TF-IDF feature space.

Random Forests offer several advantages: they naturally model non-linear decision boundaries, are robust to noise, and tend not to overfit as severely as single decision trees due to the averaging across many trees. They can capture interactions between features that linear methods completely ignore.

However, they also come with disadvantages. They are computationally expensive when trained with hundreds of trees on high-dimensional, sparse data such as TF-IDF vectors. They can require substantial memory, and they provide limited interpretability compared with coefficient-based linear models.

Moreover, Random Forests may struggle with highly imbalanced datasets, as the individual trees still tend to favor majority classes.

Hyperparameters for the Random Forest were tuned using a stratified 5-fold cross-validation procedure. A grid of candidate values for the number of trees, maximum tree depth, maximum number of features per split, and minimum leaf size was evaluated using GridSearchCV. The cross-validated tuning process selected a model with 800 trees, a maximum depth of 35, max_features ≈ 0.30, and min_samples_leaf = 2, with class weights set to "balanced" to address class imbalance. These settings were chosen to limit overfitting while still enabling individual trees to learn meaningful non-linear relationships in the data.

When evaluated, the Random Forest achieved a validation accuracy of approximately 0.71 and a test accuracy of about 0.71. The macro-averaged precision, recall, and F1-score on the test set are approximately 0.61, 0.55, and 0.56, respectively, as can be seen in Figure 6.

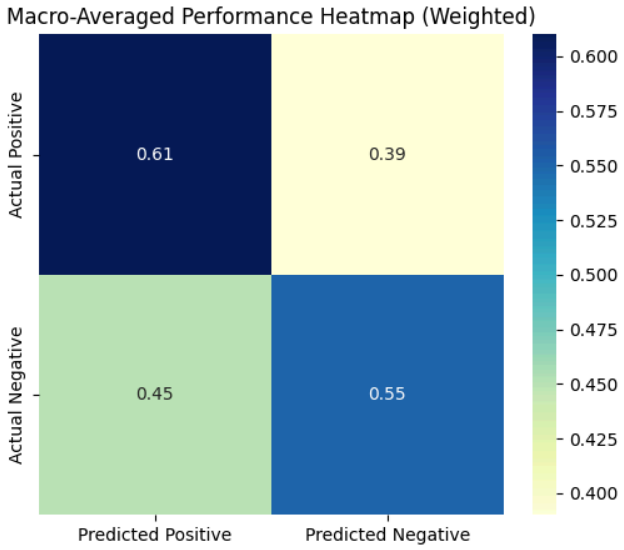

Figure 6. Resulting performance heatmap of the Random Forest model

Training the Random Forest came with several challenges. First, the high dimensionality and sparsity of TF-IDF features significantly increased computational and memory demands, making training with hundreds of deep trees slow. Second, despite the use of balanced class weights, the model continued to exhibit bias toward majority classes, a common issue for ensemble tree methods. Additionally, ensemble models provide limited interpretability, making it more difficult to understand which linguistic features drive particular predictions. Nevertheless, the Random Forest provided competitive performance and captured non-linear structure that linear models were unable to learn.

4. Results and Comparative Analysis

The initial multinomial Logistic Regression, trained without tuning, performs poorly, with accuracy near one-third and low macro-F1 scores. Introducing hyperparameter tuning dramatically improves Logistic Regression. Selecting a larger C value via cross-validation leads to substantially higher AUC values and

much more interpretable confusion matrices. The tuned model achieves macro AUC values around 0.74 across train, validation, and test, suggesting that the linear decision boundaries align well with the underlying structure of the TF-IDF features.

The calibrated Linear SVM further improved performance. Its validation and test accuracies around 0.75 and 0.74, respectively, surpass the Logistic Regression model. The SVM appears to separate most classes more cleanly, particularly the large ones, and yields robust calibrated probabilities that can be interpreted as confidence scores. Because both models are linear in the feature space, this improvement is likely due to the max-margin nature of SVMs, which often generalize better than logistic models in high-dimensional settings.

The Random Forest model, while conceptually quite different, actually matches the SVM in accuracy on this dataset, reaching approximately 0.71 accuracy on the test set. However, its macro-averaged recall and F1 scores remain lower than might be expected from its accuracy alone, because its predictions are still somewhat biased toward over-represented classes. Moreover, because the feature space is extremely sparse, the interpretability and stability of individual trees are limited.

For making predictions, the SVM model performed far better than the other models tested and managed to perform well consistently across all classes, regardless of the uneven distribution.

5. Discussion

We learned several things from this project and its difficulties. First, appropriate preprocessing, especially with datasets as uneven and high-dimensional as this one, is crucial for reliable modeling. Second, model selection and hyperparameter tuning meaningfully influence performance. A naive Logistic Regression underperforms badly, whereas a tuned Logistic Regression and a calibrated SVM perform very well.

For stakeholders such as clinicians, moderators, or platform designers, the calibrated SVM appears to be the best model for predictive purposes. Additionally, the ability of the SVM to present, for each post, a ranked list of likely subreddits with associated probabilities makes it possible to design decision-support tools.

6. Conclusion and Future Work

This project demonstrates that classical machine learning methods can effectively classify Reddit mental-health posts into ten subreddit-based categories when supplied with appropriate TF-IDF features and careful preprocessing. Among the models evaluated, the calibrated Linear SVM and the tuned Logistic Regression provide the best balance of accuracy, AUC, and interpretability, while a well-configured Random Forest offers accuracy but somewhat less balanced multi-class performance.

There are several avenues for future work. More sophisticated tree-based methods such as gradient boosting (XGBoost or LightGBM) could be used. On the feature side, moving beyond TF-IDF to contextual embeddings from transformer-based models such as BERT may yield further performance improvements by incorporating semantic information rather than purely lexical counts. Finally, advanced

techniques for handling class imbalance, such as focal loss, cost-sensitive learning, or synthetic oversampling could further improve performance.