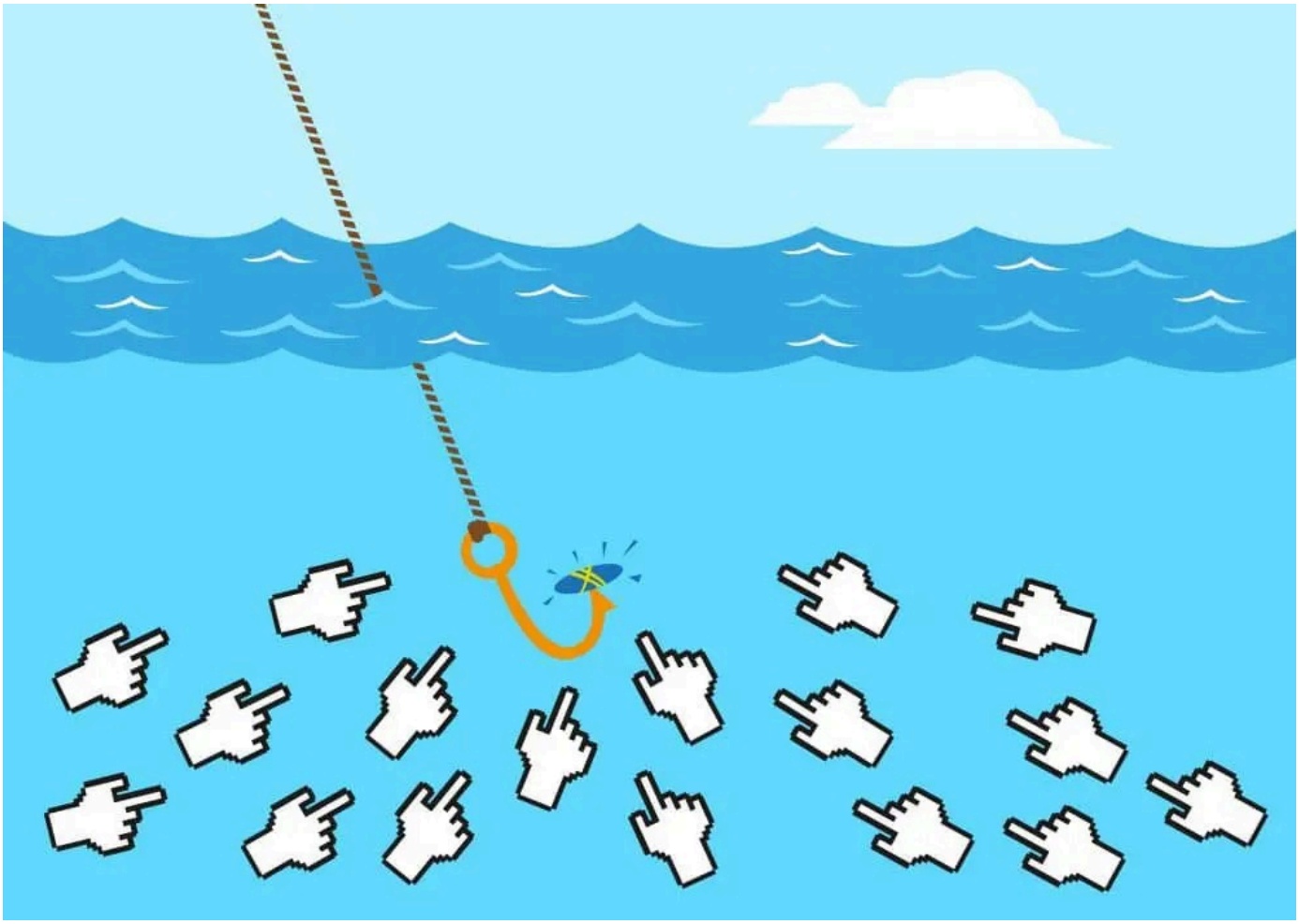
Clickbait Detector - Final



cyberhoot.com

Introduction and Background

In the digital age, where content is monetized based on views and clicks, content creators often use clickbait—misleading or exaggerated titles to attract attention. This practice frustrates users and spreads misinformation. Detecting and mitigating clickbait is crucial to enhance content quality and user experience. Current models struggle to generalize across various types of media, such as news articles and video titles. To detect clickbait, various proposals have been tested, including simpler supervised classifiers (e.g. Random Forest [1]), and more advanced deep learning methods (e.g. Convolutional Neural Networks [2]). Publicly available clickbait datasets also vary, with differing feature sets and sample sizes, from social media ranging from YouTube [3] to news sites [4].

This project aims to build a more robust AI model to detect clickbait across different media platforms by analyzing common features of deceptive headlines. By combining data preprocessing and machine learning, the goal is to develop a versatile solution applicable to a wide range of online content [5].

Problem Definition

Monetization of online media has created a massive market for consumer attention, pushing creators to make their content as clickable as possible. Users decide which media to consume based on short titles, leading to the rise of clickbait, where titles are exaggerated or misleading to attract users. The aim of this project is to create an AI model that detects clickbait, which could potentially block such content. Previous efforts have focused on detecting clickbait in specific media forms [2], but this project seeks to uncover general insights into clickbait techniques and apply them across different platforms. The model will be trained on a large dataset of clickbait articles and tested to evaluate its efficacy.

Methods

In this project, data cleaning and feature engineering steps have been implemented to prepare text data for a clickbait detection dataset. Data cleaning has started by filtering out entries with fewer than 20 characters through the is_text_not_too_short function, ensuring that only substantial text samples are retained for further processing. This function verifies the length of each entry, discarding short and potentially irrelevant samples. Additionally, the clean_text function has been applied to standardize the text format by removing any leading and trailing whitespace, promoting consistency across the dataset.

For feature engineering, the add_word_and_character_features function has been developed to construct text-based features that may enhance model performance. Key features such as character_count and word_count have been included to capture the length and verbosity of each text entry, while average_word_length has been introduced to provide insights into word complexity. Ratios for stylistic elements, such as uppercase_ratio, numeral_ratio, and punctuation_ratio, have been calculated to reflect structural aspects often seen in clickbait. Additionally, linguistic analysis has been performed by including ratios of various parts of speech—adjective_ratio, adverb_ratio, interjection_ratio, noun_ratio, pronoun_ratio, proper_noun_ratio, and verb_ratio—which capture language patterns that may characterize clickbait content.

Together, these features have been designed to provide a comprehensive representation of the text, enhancing the model's ability to distinguish between clickbait and non-clickbait text based on structural, stylistic, and linguistic attributes.

First - PCA:

Now that the data has been preprocessed, model implementation and training can proceed. A combination of one unsupervised and two supervised learning approaches have been selected to explore different aspects of the dataset and achieve a comprehensive analysis. For the unsupervised model, Principal Component Analysis (PCA) has been applied to reduce the dimensionality of the feature set, allowing for an exploration of the underlying structure without relying on labeled data. PCA has been selected to identify key patterns and correlations within the features, assisting in feature selection and reducing redundancy.

Second - RandomForest:

The second supervised model has been implemented using a Random Forest algorithm, chosen for its robustness and capability to handle high-dimensional data while capturing complex feature interactions—critical considerations given the diverse set of linguistic and structural features. Additionally, Random Forest's resilience to overfitting has made it well-suited for accurately distinguishing between clickbait and non-clickbait across varied text samples.

Third - SVM:

The second supervised model has been implemented using a Support Vector Machine (SVM) with a radial basis function (RBF) kernel. SVM was chosen for its ability to handle high-dimensional feature spaces effectively, which is particularly useful given the diverse linguistic and structural attributes engineered in this dataset. The RBF kernel enables the model to capture non-linear relationships between features, making it well-suited for distinguishing between subtle patterns in clickbait and non-clickbait text.

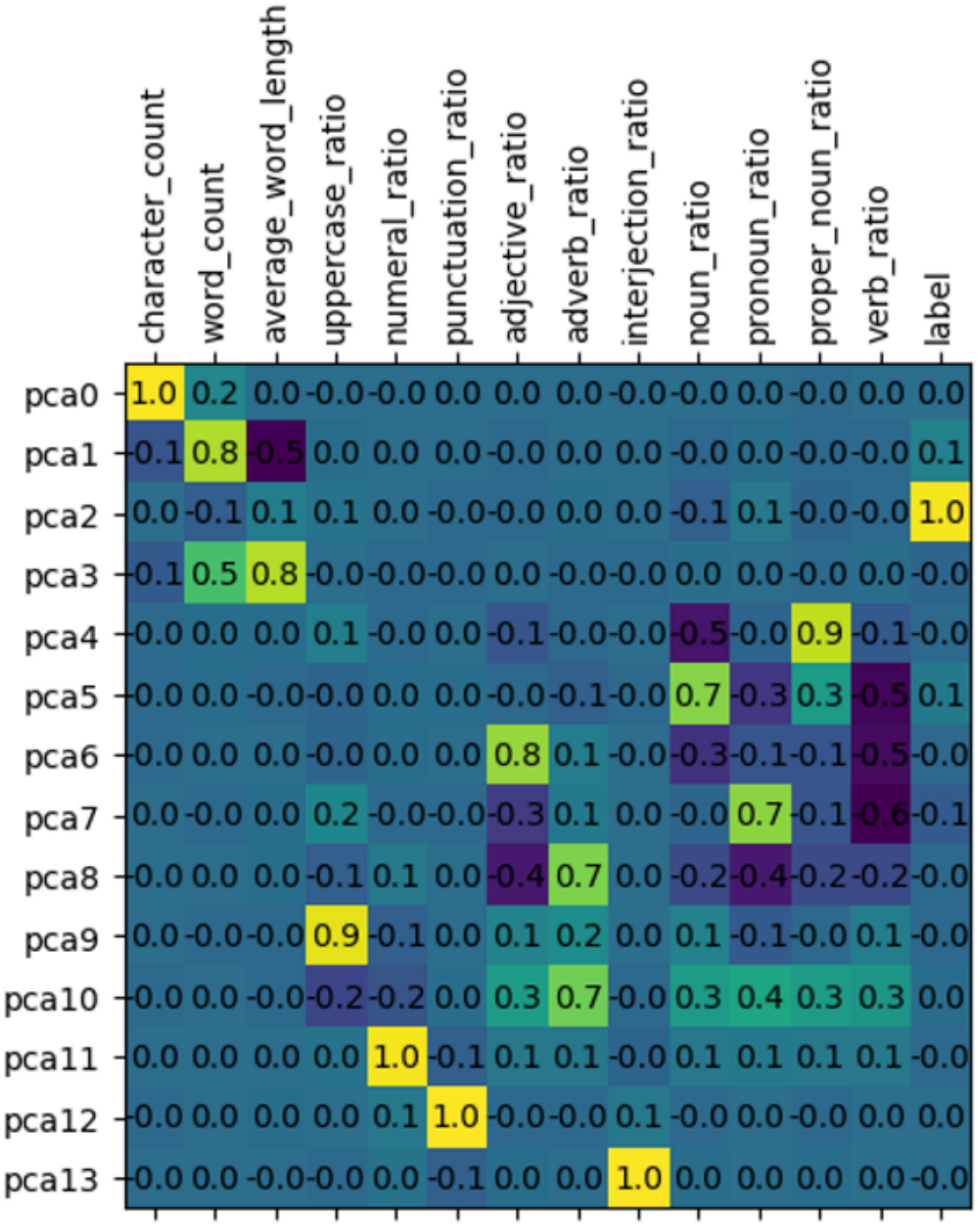
SVM optimizes a decision boundary by maximizing the margin between classes, ensuring robust generalization to unseen data. By tuning hyperparameters such as the regularization parameter C and the kernel coefficient gamma, the model achieves a balance between underfitting and overfitting. A higher value of C allows the model to classify training data more accurately, at the risk of overfitting, while a lower C value encourages a simpler decision boundary for better generalization.

By employing these three approaches, both interpretability and predictive accuracy have been targeted to improve the model's ability to identify clickbait content effectively.

Results and Discussion

PCA Results:

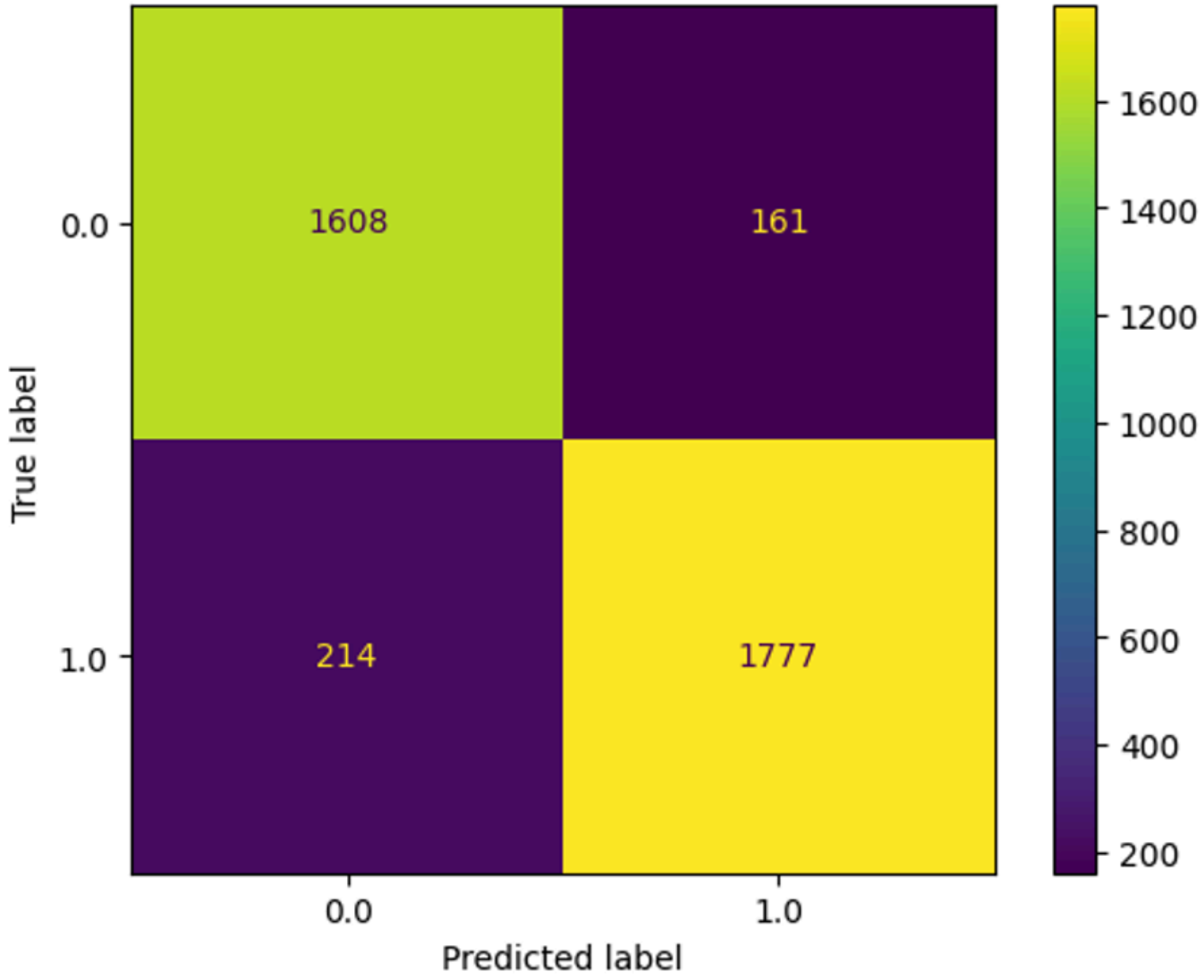
On the left of this visualization are the pca features ordered from the one which explains the most variance (pca0) to the one which explains the least variance (pca13). At the top are the features of our data, and the matrix shows which feature in our data the given pca feature represents. For example, since pca0 is mapped to character_count with a value of 1.0, we understand that pca0 is essentially representing the character count. Therefore, this visualization of our PCA results shows us that it considers character count to be the feature which retains the most variance. It would follow that word count, label, and average word length are the next most important variables in retaining variance, but our results show a significant drop off with pca0 explaining 98.7% of the variance followed by pca1 and pca2 which explain 1.14% and 0.08% of the variance respectively.



PCA Results

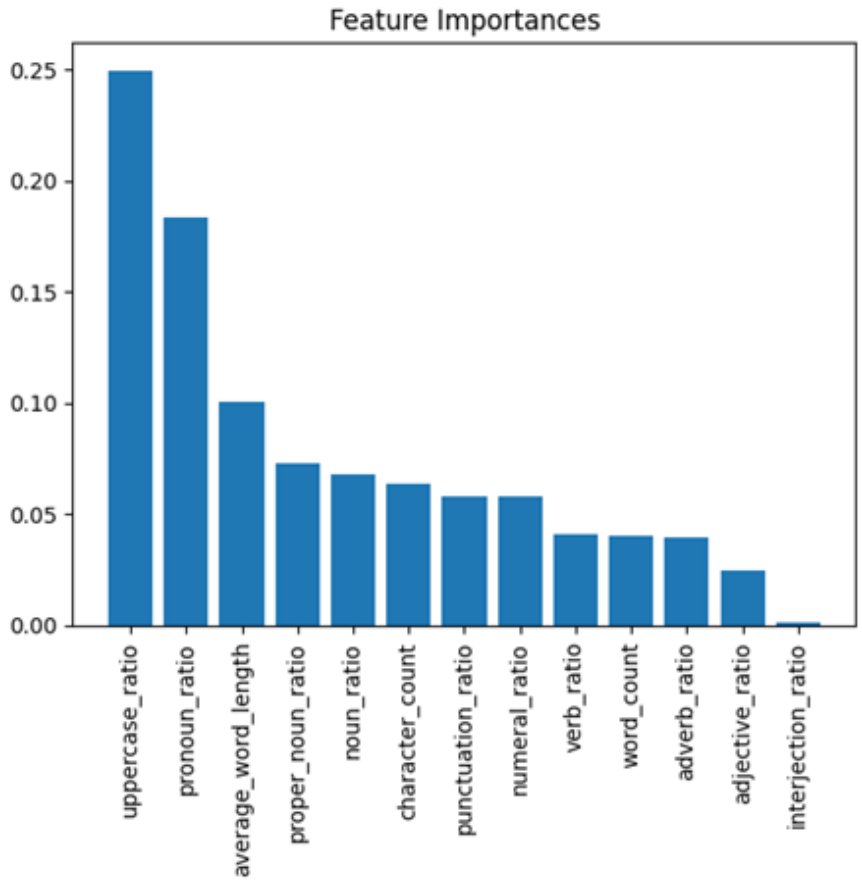
RandomForest Results:

Our RandomForest implementation scored a 99% training accuracy, a 99% validation accuracy, and a 90% test accuracy. This tells us that the model is very good at predicting whether or not a given video title is attempting to clickbait users. These results also potentially indicate slight overfitting since the training accuracy is nearly perfect while the test accuracy is a bit lower, but the predictions are still very accurate nevertheless.



RandomForest Results

The confusion matrix for our RandomForest implementation shows us that the model predicts slightly more false negatives than false positives. The difference isn't big enough to indicate a clear and distinct trend, especially considering how good the accuracy is overall. Thus, we do not believe there to be any distinct bias in the model from this data.



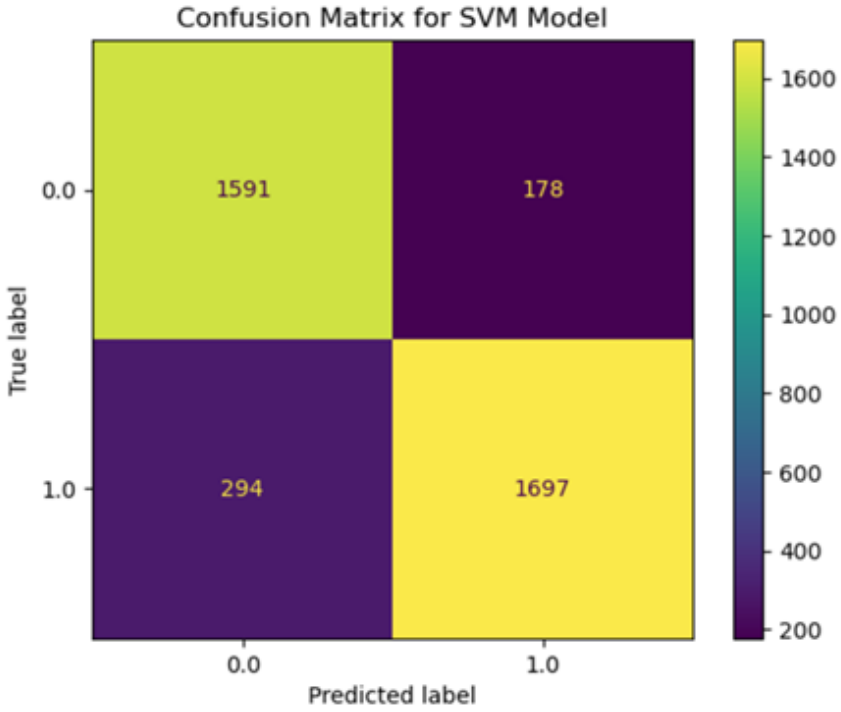
RandomForest Results

Conducting further analysis from our Random Forest model's results, we were able to visualize the features it found to be most important in the classification task. From our analysis it seems that the model determined the uppercase_ratio feature to be the most important in identifying clickbait. This makes sense because the capitalization of every letter in a word is often used to grab people's attention and can understandably be a good way to exaggerate a statement so someone clicks the title. This feature is followed by the pronoun ratio which could be explained by titles that want to raise questions in people's minds so that they click on the content. Unless there is a famous name that many people care about involved in the content, then the title could be more enticing if the identities are obfuscated so that consumers have to view the content to have this newly planted question answered. Overall, we can learn a great deal about not only identifying clickbait titles but also about what clickbaiters have determined to be the most effective strategies from the model's feature importance determinations.

SVM Results:

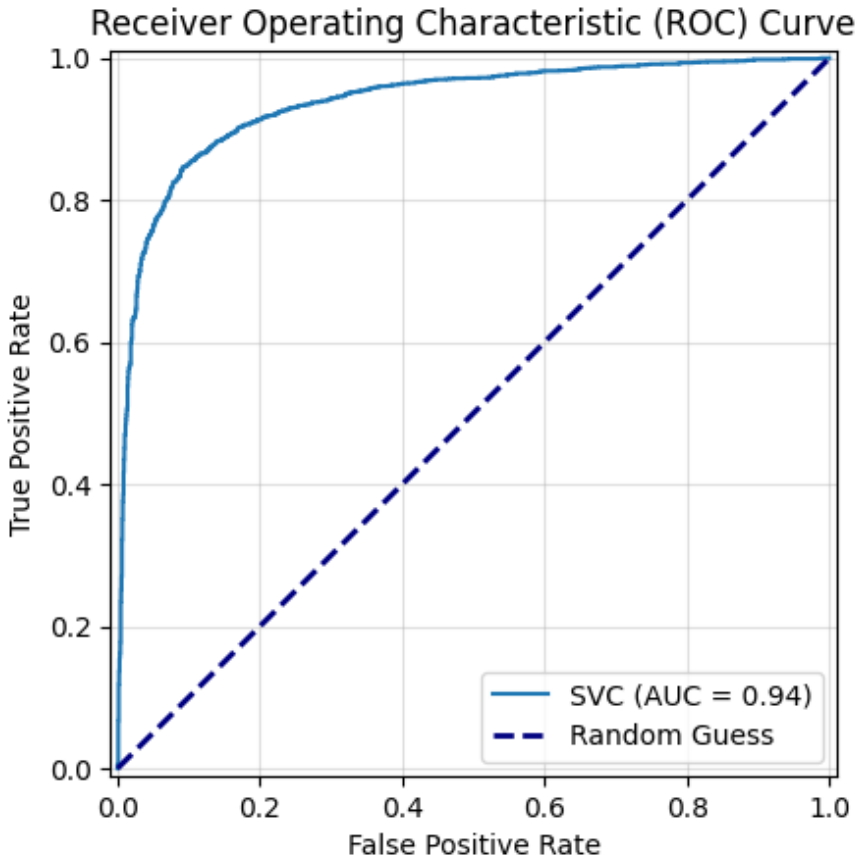
The results of the Support Vector Machine (SVM) model with an RBF kernel demonstrate its strong performance in the clickbait detection task. The model achieved an accuracy of 88.4% on the training set and 87.4% on the test set, indicating good generalization with minimal overfitting. The classification report shows high precision, recall, and F1-score for both classes. For the non-clickbait class (label 0), the precision, recall, and F1-score are 0.84, 0.90, and 0.87, respectively, while for the clickbait class (label 1),

these metrics are 0.90, 0.85, and 0.88. These results suggest that the model is slightly better at identifying clickbait content than non-clickbait content, likely due to the rich feature engineering process.



SVM Results

The Receiver Operating Characteristic (ROC) curve confirms the models ability to distinguish between the two classes, with an Area Under the Curve (AUC) of 0.88. This score demonstrates the models strong discriminative power and its capability to handle the datasets complexities effectively. Overall, the results reflect that the SVM model is robust and well-suited for the task, leveraging both its ability to capture non-linear relationships through the RBF kernel and the diverse feature set provided. However, further optimization or ensemble approaches might reduce the remaining misclassifications.



SVM Results

Model Comparison

The three models that we have implemented in this project are PCA, Random Forest, and SVM. PCA is an unsupervised learning method, so it can be used to explore the dataset and identify patterns. It can also serve as a way to reduce the dimensionality of the data and select the most important features to use with other learning methods. However, because unsupervised learning methods only have access to unlabeled data, PCA cannot be used to make predictions about the data labels. In our case, this means that our PCA model is unable to classify a title as clickbait or not.

Supervised learning methods, which include our Random Forest and SVM classifiers, are able to solve this problem. These methods can learn the relationships between the data and the labels during training and make predictions for classification. We will now compare our Random Forest and SVM models in detail.

In terms of performance, both the Random Forest and SVM models achieve high accuracies on the test set, with the Random Forest model performing slightly better. Examining the confusion matrices, we can see that the largest discrepancy lies in how the models classify positive examples corresponding to the “1” label (clickbait titles). Compared to the Random Forest model, the SVM model incorrectly classifies more clickbait titles as non-clickbait. So while the two models have similar precision scores, the SVM model obtains a significantly worse recall score.

SVM classifiers work well when the classes are cleanly separated in a high-dimensional space because they create smooth decision boundaries that maximize the gap between the classes. Based on the relatively poor performance of our SVM model, we can infer that this may not be the case for our dataset. Random Forest classifiers, on the other hand, can perform well even when the classes are not easily separable because the decision boundary can be made arbitrarily complex (at the risk of overfitting).

Another way in which our two supervised methods differ is the time needed to train the models and make predictions. The SVM model is much slower compared to the Random Forest model, taking approximately 34 times as long to train (70.9 seconds vs. 2.1 seconds locally), and about 71 times as much to classify the test set (0.9137 seconds vs. 0.0128 seconds). The Random Forest model is significantly faster because its trees can be trained in parallel, and sklearn's implementation of Random Forest leverages multi-threading. In contrast, the SVM model operates as a single-threaded algorithm. While large kernel matrices in SVMs (~30k×30k floats, approximately 3GB) can create a memory bottleneck if RAM is insufficient, this primarily affects memory rather than compute speed. On a machine with 16GB of RAM, a single-threaded Random Forest may have a runtime comparable to SVM, but parallelization makes Random Forest much more efficient in practice.

To summarize, the unsupervised PCA model provides insights on the features of our dataset, while among the two supervised models, the Random Forest classifier is more suitable for our task, both in terms of performance and time efficiency.

Next Steps

The next steps involve implementing a practical real-time clickbait detection tool, such as a browser extension or API integration, to filter or flag clickbait dynamically across various platforms. Expanding the dataset to include more diverse and multilingual content will enhance model generalization, while further optimization, such as exploring ensemble methods combining Random Forest and SVM, can improve accuracy and recall. Additionally, refining the Streamlit interface with interactive features will provide users with an intuitive way to upload datasets, view predictions, and interpret feature importance, ensuring the tool usability and effectiveness.

References

[1] M. Potthast, S. Köpsel, B. Stein, and M. Hagen, ‘Clickbait Detection’, in Advances in Information Retrieval, 2016, pp. 810–817.

[2] A. Agrawal, "Clickbait detection using deep learning," 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), Dehradun, India, 2016, pp. 268-272, doi:

10.1109/NGCT.2016.7877426.

[3] A. Zavalny, “Youtube Clickbait Classification,” Kaggle, 2021.

[4] Y. Kashnitsky, ‘Clickbait news detection’. Kaggle, 2019.

[5] R. Raj, C. Sharma, R. Uttara and C. R. Animon, "A Literature Review on Clickbait Detection Techniques for Social Media," 2024.

Gantt Chart

Access the Chart

Contributions

Contributions Table