Evaluating Machine Learning Approaches for Personality Recognition: Classifying Authors as Neurotic or Calm from Text

This project explores machine learning techniques for classifying authors as "neurotic" or "calm" based on their writing, using the MyPersonality and Essays datasets. We evaluate different feature extraction methods (one-hot encoding, term frequency, and TF-IDF) and classifiers (SVM, k-NN, Random Forest, and Gradient Boosting) to identify effective approaches for personality recognition from text. The best-performing model achieved an F1-score of 0.51, demonstrating competitive results but highlighting the potential for further improvement by incorporating external resources and more advanced techniques.

Introduction

This project investigates machine learning techniques to classify authors as "neurotic" or "calm" based on their writing, using the MyPersonality and Essays datasets from the Workshop on Computational Personality Recognition (Celli et al., 2013). Previous efforts have struggled to achieve high accuracy, often producing results near random chance. The goal is to evaluate different feature extraction and classification methods to identify effective approaches for personality recognition from text.

Background

This project uses two datasets: MyPersonality and Essays. Both contain labeled documents for personality recognition based on the Big Five traits: Openness (insightful vs. unimaginative), Conscientiousness (organized vs. careless), Extraversion (sociable vs. shy), Agreeableness (friendly vs. uncooperative), and Neuroticism (neurotic vs. calm).

Table 1. Sample Documents.

Label	Sample
	MyPersonality
Calm	Was stuck between reality and a dream...unpleasant. Let's go to Italia~~ Is going back to the homeland "Oh my God dude, doing 5 tomorrow is going to feel like sex!"
Neurotic	Just bought a cute pair of purple pumps :) Not yet - going to movies tonight... is sick in bed. Coughing like and old hag and looking decidedly ropey
	Essays
Calm	Well, I feel good about the fact that I am getting this assignment done well before it is due. Today is one of those days that I feel really motivated to do my homework
Neurotic	Well, right now I just woke up from a mid-day nap. It's sort of weird, but ever since I moved to Texas, I have had problems concentrating on things

The MyPersonality dataset contains about 9,900 Facebook statuses from 250 authors, while Essays includes roughly 2,500 essays from unique authors. MyPersonality is more imbalanced: 39.6% of authors are neurotic and 60.4% calm, while Essays has a balanced distribution of 50% neurotic and 50% calm (Figure 1).

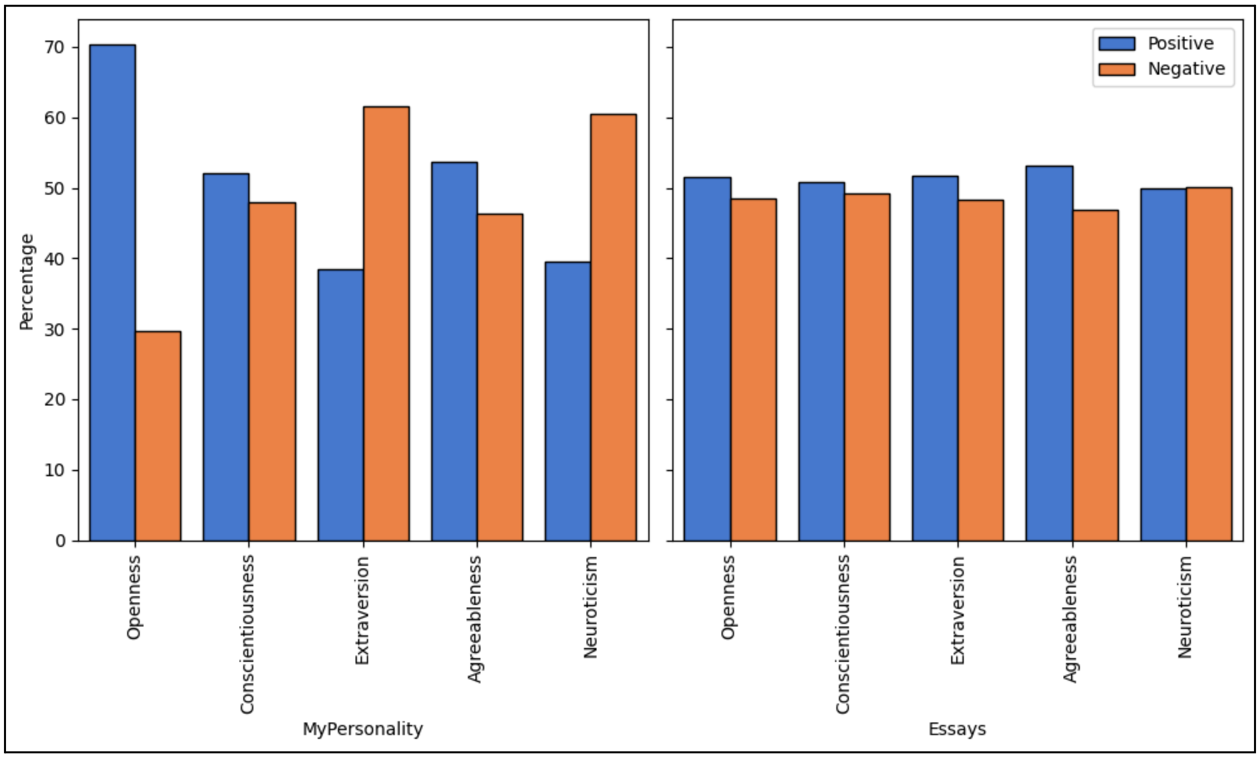

Figure 1. Author Distributions by Trait.

Methods

The data was loaded using Windows-1252 encoding, converted to single-precision floating-point numbers and unsigned integers, and missing values were ignored. Documents from the same author were concatenated into single documents.

A pipeline was built combining a vectorizer and classifier. The vectorizer transforms documents into feature vectors, and the classifier makes predictions based on these vectors. We selected and tuned both vectorizers and classifiers to optimize performance.

Three types of vectorizers were tested: one-hot encoding, term frequency (TF), and term frequency-inverse document frequency (TF-IDF). For each vectorizer, we adjusted the n-gram range (unigrams, bigrams), maximum document frequency (1.0, 0.5), and minimum document frequency (1, 2).

Four classifiers were tested: Support Vector Machine (SVM), k-Nearest Neighbors (k-NN), Random Forest, and Gradient Boosting. For SVM, we tuned the regularization strength (0.01, 0.1, 1, 10, 100) and kernel type (linear, RBF). For k-NN, we adjusted the number of neighbors (1, 3, 5, 7, 9) and distance metric (Euclidean, cosine). For Gradient Boosting, we tuned the maximum tree depth (1, 2, 3, 4, 5).

A grid search tested all combinations of these parameters (Figure 2). Each model was scored using balanced accuracy with 3-fold cross-validation on the MyPersonality dataset. The highest-scoring model was retrained on the MyPersonality data and tested on the Essays dataset.

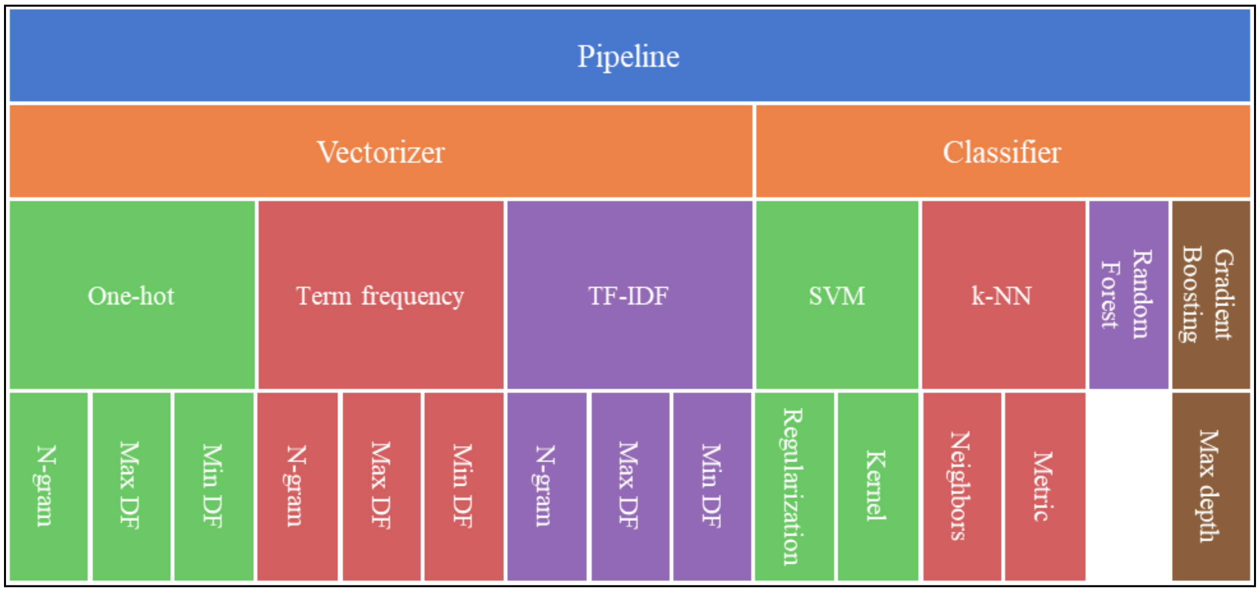

Figure 2. Parameter Search Space.

Results

A total of 624 unique models were trained during the grid search. The mean balanced accuracy was 0.50 (SD = 0.02), with scores ranging from 0.43 to 0.57 and an interquartile range (IQR) of 0.49–0.51 (Figure 3).

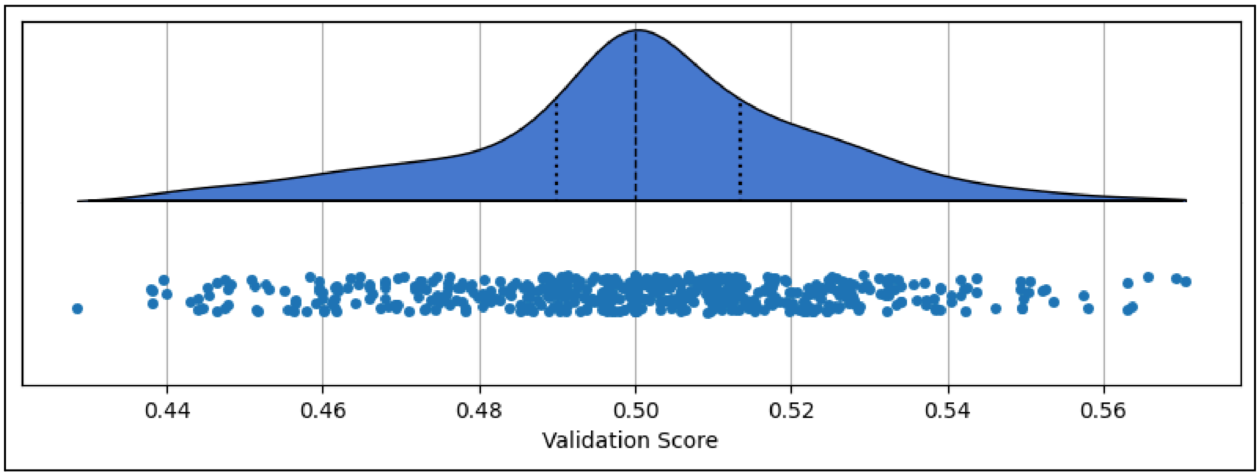

Figure 3. Validation Score Distribution.

Each vectorizer was evaluated across 208 models. The one-hot encoding vectorizer had a mean score of 0.50 (SD = 0.02), with scores ranging from 0.44 to 0.55 and an IQR of 0.48–0.51. The term frequency vectorizer achieved a mean score of 0.50 (SD = 0.03), with scores ranging from 0.44 to 0.57 and an IQR of 0.49–0.52. TF-IDF performed similarly, with a mean score of 0.50

(SD = 0.02) and scores ranging from 0.43 to 0.57, with an IQR of 0.50–0.51. All vectorizers showed similar performance, with TF-IDF having the smallest IQR but also the most extreme outliers (Figure 4).

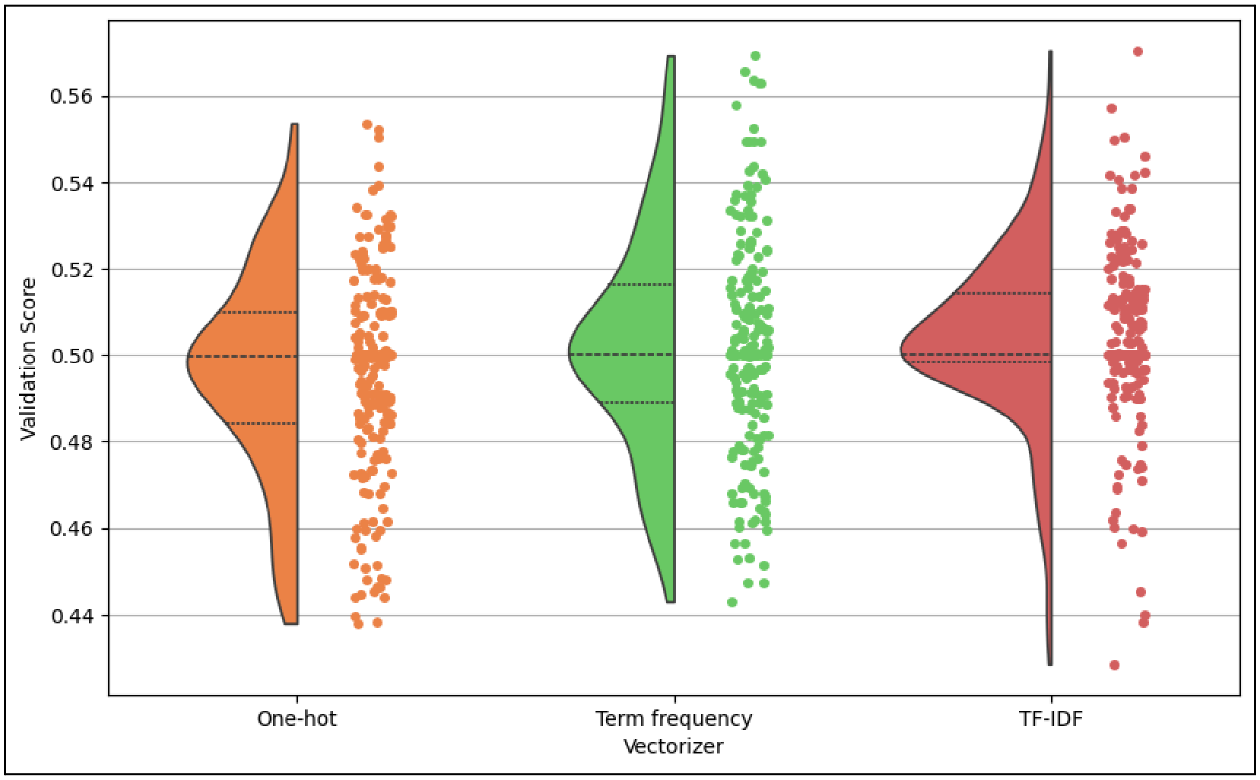

Figure 4. Validation Score Distributions by Vectorizer.

For classifiers, SVM was evaluated in 240 models, with a mean score of 0.50 (SD = 0.02), scores ranging from 0.45 to 0.56, and an IQR of 0.50–0.51. The k-NN classifier performed slightly better, with a mean score of 0.51 (SD = 0.02), scores ranging from 0.45 to 0.57, and an IQR of 0.49–0.52. Random Forest showed the best mean score of 0.52 (SD = 0.01), with scores ranging from 0.49 to 0.55 and an IQR of 0.51–0.53. Gradient Boosting had the lowest mean score of 0.47 (SD = 0.02), with scores ranging from 0.43 to 0.53 and an IQR of 0.46–0.49.

Random Forest had the best overall performance, followed by SVM and k-NN, which performed similarly. K-NN achieved the best individual model score but showed less consistency. Gradient Boosting performed the worst (Figure 5).

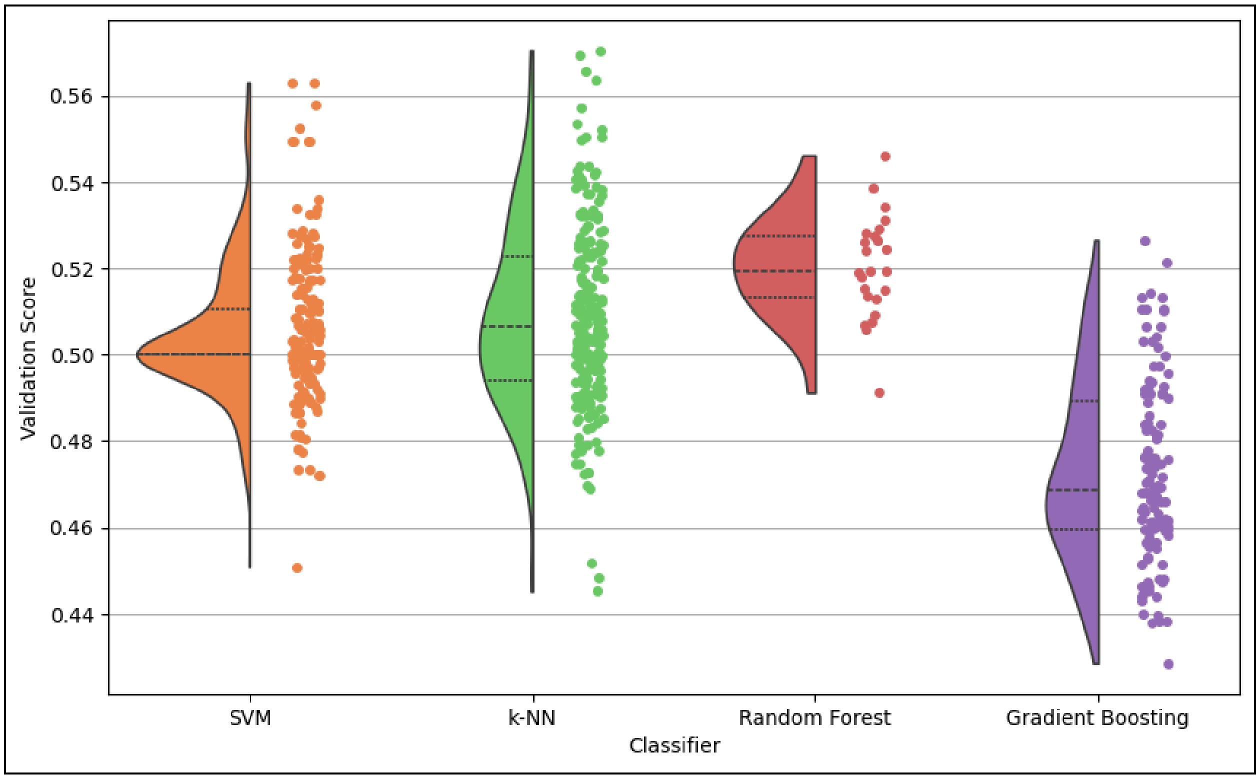

Figure 5. Validation Score Distributions by Classifier.

The best model used the TF-IDF vectorizer with bigrams, a maximum document frequency of 0.5, and a minimum document frequency of 2. The k-NN classifier used 9 neighbors and the Euclidean distance metric (Table 2). This model achieved an F1-score of 0.51 on the test data, indicating balanced but low accuracy, similar to random guessing (Table 3).

Table 2. Best Model Parameters.

TF-IDF Vectorizer			k-NN Classifier	
N-gram	Max DF	Min DF	k	Metric
Bigrams	0.5	2	9	Euclidean

Table 3. Performance of Best Model on Essays.

Label	Precision	Recall	F1-Score	Support
Calm	0.51	0.47	0.49	1235
Neurotic	0.51	0.55	0.53	1233
Accuracy			0.51	2468
Macro Average	0.51	0.51	0.51	2468
Weighted Average	0.51	0.51	0.51	2468

Discussion

This project explored machine learning models for personality recognition from text, using the MyPersonality and Essays datasets. The best-performing model achieved an F1-score of 0.51, suggesting competitive performance, but there is still significant room for improvement. This section discusses classifier performance, compares results with previous work, identifies limitations, and proposes future directions.

SVM showed consistent performance, particularly when paired with the term frequency vectorizer using unigrams or bigrams, with a maximum document frequency of 1.0 and a minimum document frequency of 2. The linear kernel and a regularization strength of 10 provided the best balance between bias and variance. These results suggest that SVM is effective for this task with a medium-to-high regularization strength and a term frequency-based representation.

k-NN performed best overall, especially with TF-IDF or term frequency vectorizers and bigrams. The optimal parameters were a maximum document frequency of 0.5 and a minimum document frequency of 2, which reduced the feature space while maintaining performance. The distance metric had limited impact, with both Euclidean and cosine distances performing similarly, although cosine distance is generally more effective in text-based tasks. For TF-IDF, the best results were achieved with 9 neighbors, while 3 neighbors were optimal for term frequency. These findings suggest that k-NN benefits from a larger neighborhood when using TF-IDF features.

Random Forest showed stable performance but did not outperform k-NN or SVM. The choice of vectorizer had minimal effect, though TF-IDF with bigrams provided slightly better results. The optimal configuration was a maximum document frequency of 1.0 and a minimum document frequency of 2, balancing feature richness and efficiency. Random Forest did not provide a clear advantage, suggesting it is less suited for this task compared to k-NN and SVM.

Gradient Boosting was the least effective classifier, with performance significantly lower than the others. Despite using TF-IDF and bigrams, it showed high variability and did not appear in the top models, indicating it is unsuitable for personality recognition from text.

Our results align with previous work on personality recognition. Mohammad et al. 2013 reported an accuracy of 0.57 using SVM on the Essays dataset, and Iacobelli et al. 2013 achieved 0.56 using Naïve Bayes. Our models, particularly SVM and k-NN, performed similarly, showing that our approach—using term frequency and TF-IDF with bigrams—is competitive with established methods.

However, our best model achieved an F1-score of 0.51, lower than the 0.57 reported by Mohammad et al. 2013, who combined unigrams with external resources like the Twitter Hashtag Lexicon. Markovic et al. achieved an F1-score of 0.90 using SVM and boosting, incorporating a much larger feature space with social and demographic data, lexical resources, and linguistic features. These studies highlight the potential for improving our results by integrating such resources.

One limitation of our project is the absence of external lexical resources like LIWC, MRC, or the Twitter Hashtag Lexicon, which could improve performance. Incorporating these resources would enhance the model's ability to capture semantic and psychological cues, improving accuracy. Social media and demographic features could further enrich the feature space, as demonstrated by Markovic et al. 2013.

Another limitation is the relatively small and imbalanced dataset. Expanding the dataset with more diverse sources could enhance robustness. Incorporating multimodal data, such as text with author demographics, could improve performance by allowing the models to extract more meaningful patterns.

The feature extraction process and hyperparameter tuning were crucial to our results. The best configurations for k-NN and SVM were sensitive to the choice of vectorizer, n-grams, and regularization parameters. Advanced techniques, such as word embeddings or deep learning models, could provide deeper insights into the semantic structure of text and further improve performance.

Conclusion

This project demonstrates that k-NN and SVM classifiers, paired with TF-IDF or term frequency vectorizers and bigram features, can achieve competitive results for personality recognition, with an F1-score of 0.51. However, there is significant potential for improvement. Incorporating external lexical resources, social media data, and demographic features could enhance accuracy. Additionally, adopting advanced techniques like word embeddings or deep learning could enable better semantic understanding of text. Future work should explore these avenues to create more robust models for personality recognition in real-world applications.

References

Celli, F., Pianesi, F., Stillwell, D., & Kosinski, M. (2013). Workshop on computational personality recognition: Shared task. In Proceedings of the International AAAI Conference on Web and Social Media (Vol. 7, No. 2, pp. 2-5).

NLTK. (n.d.). Porter stemmer. Retrieved December 11, 2024, from https://www.nltk.org/api/nltk.stem.porter.html

NLTK. (n.d.). WordNetLemmatizer. Retrieved December 11, 2024, from https://www.nltk.org/api/nltk.stem.WordNetLemmatizer.html

Rehurek, R. (n.d.). Doc2Vec. Retrieved December 11, 2024, from https://radimrehurek.com/gensim/models/doc2vec.html

Rehurek, R. (n.d.). Word2Vec. Retrieved December 11, 2024, from https://radimrehurek.com/gensim/models/word2vec.html

Scikit-learn. (n.d.). balanced_accuracy_score. Retrieved December 11, 2024, from https://scikit-learn.org/dev/modules/generated/sklearn.metrics.balanced_accuracy_score.html

Scikit-learn. (n.d.). CountVectorizer. Retrieved December 11, 2024, from https://scikit-learn.org/1.5/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

Scikit-learn. (n.d.). GradientBoostingClassifier. Retrieved December 11, 2024, from https://scikit-learn.org/dev/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

Scikit-learn. (n.d.). GridSearchCV. Retrieved December 11, 2024, from https://scikit-learn.org/dev/modules/generated/sklearn.model_selection.GridSearchCV.html

Scikit-learn. (n.d.). KNeighborsClassifier. Retrieved December 11, 2024, from https://scikit-learn.org/dev/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Scikit-learn. (n.d.). RandomForestClassifier. Retrieved December 11, 2024, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Scikit-learn. (n.d.). StratifiedKFold. Retrieved December 11, 2024, from https://scikit-learn.org/dev/modules/generated/sklearn.model_selection.StratifiedKFold.html

Scikit-learn. (n.d.). SVC. Retrieved December 11, 2024, from https://scikit-learn.org/dev/modules/generated/sklearn.svm.SVC.html

Scikit-learn. (n.d.). TfidfVectorizer. Retrieved December 11, 2024, from https://scikit-learn.org/1.5/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html