DOĞUŞ
UNIVERSITY

Author Classification Using Text Mining and Machine Learning Techniques

COME 448
Data Mining and Knowledge Discovery
Assoc.Prof.Dr.Aysun GÜRAN

202402001085, Soner Astan

May 7, 2025, Data Mining Course – Project Assignment

Draft Of Content

List of Figures

1 Introduction

In this chapter, we introduce the central problem of author classification and explain its relevance in the context of real-world applications. We explore why this topic is important in today's digital age and how machine learning techniques can contribute to solving it. The section also outlines the motivation for this project and the overarching goal of distinguishing authors through data-driven methods.

1.1 Problem

Author classification is the task of automatically identifying the author of a given text based on their writing style. This problem is highly relevant in digital forensics, copyright attribution, and automated content moderation. In an era where vast amounts of textual data are published daily, reliable author identification is essential for validating authorship, detecting plagiarism, and analyzing stylistic patterns in various contexts such as literature, academia, and social media.

1.2 Importance of the topic

The ability to accurately classify authors has wide-ranging applications in academia, publishing, legal investigations, and natural language processing research. With the increasing use of AI-generated and anonymous content, author attribution is becoming more critical than ever. Through machine learning and text mining techniques, we aim to develop robust models capable of learning stylistic cues from text and distinguishing authors with high accuracy. This project contributes to the field by evaluating various classical and modern models on Turkish textual data, incorporating both traditional features (TF-IDF, n-grams) and contextual embeddings (BERT).

2 Preprocessing

Preprocessing is a critical step in any text classification pipeline, as it transforms unstructured textual data into a numerical format that machine learning models can process effectively. This chapter outlines the dataset used, describes each step taken to clean and structure the text data, and explains the feature extraction methods that were applied. The overall goal was to ensure that the models could accurately learn from stylistic and semantic patterns in the texts, with minimal noise and maximum discriminative power.

2.1 The Dataset

The dataset used in this project consists of Turkish text samples written by multiple authors. Each text is annotated with an author label, indicating which individual wrote the passage. For machine learning compatibility and anonymity, these author names were replaced with numerical class labels (e.g., *0, 1, 2,* etc.).
To ensure reliable and generalizable evaluation, the dataset was split into training and test subsets using a stratified sampling strategy. This method preserves the distribution of author labels across both subsets, ensuring that each class is equally represented and that no model is biased due to class imbalance in the test data.

2.2 Preprocessing steps

Before extracting features from the text data, several preprocessing steps were applied to improve consistency and reduce noise:

- Stopword Removal: A predefined list of Turkish stopwords (e.g., “ve”, “ama”, “bu”) was removed from the text. These frequently used words contribute little to stylistic differentiation and were excluded to improve the quality of extracted features.
- Parallel Processing: To optimize performance, both preprocessing and feature extraction were executed in parallel across multiple CPU threads.

This significantly reduced runtime and enabled efficient handling of large volumes of data.

The cleaned and processed text data was then transformed into structured features using a combination of classical and modern techniques, which are detailed below.

2.3 Feature extraction methods

Feature extraction is the process of converting textual data into numerical representations that capture both syntactic and semantic information. In this project, two main types of features were used: traditional frequency-based methods (TF-IDF and n-grams) and contextual word embeddings from a pre-trained deep learning model (BERT).

2.3.1 TF-IDF

TF-IDF (Term Frequency – Inverse Document Frequency) is a classical statistical method that scores words based on their frequency within a document and their rarity across all documents. Terms that appear often in a single text but rarely across the corpus receive a higher weight, indicating their importance to that specific document.

In this project, TF-IDF vectors were constructed using a custom implementation that included:

- Double normalization (0.5): This technique normalized term frequencies to reduce the impact of document length and outliers.
- Smoothed IDF: A variation of IDF calculation that prevents division by zero and stabilizes scores for rare terms.

TF-IDF was computed across both word and character n-grams, providing a flexible representation of stylistic features.

2.3.2 n-grams

An n-gram is a contiguous sequence of *n* tokens from a given text. These sequences can be based on either words or characters:

- Word n-grams capture common multi-word phrases and stylistic patterns, such as “çok güzel” (“very beautiful”) or “bu nedenle” (“therefore”).
- Character n-grams detect recurring character patterns, suffixes, or stylistic markers within and across words.

For this project, the following n-gram features have been extracted:

- Word-based n-grams: Unigrams, bigrams, and trigrams (1-gram to 3-gram), with stopword removal.
- Character-based n-grams: Bigrams and trigrams (2-gram to 3-gram), without stopword filtering.

These features provide a robust statistical representation of textual style and are particularly effective for author identification tasks.

2.3.3 BERT

BERT (Bidirectional Encoder Representations from Transformers**)** is a state-of-the-art deep learning model developed by Google that captures rich, context-aware embeddings of text. Unlike TF-IDF and n-grams, BERT considers both the left and right context of a word in a sentence, enabling a deeper understanding of semantics and syntax.

In this project, we used the dbmdz/bert-base-turkish-cased model, which is specifically trained on Turkish language data. Each text was tokenized and processed by the BERT model, which generated contextualized embeddings for every token. These token embeddings were then mean-pooled across each sentence to obtain a single dense vector representation for each document.

BERT embeddings were particularly valuable for capturing semantic meaning and syntactic structure that is difficult to model using only frequency-based techniques.

3 Experiment

This chapter presents the experimental research process that was conducted in order to evaluate the performance of various machine learning models on the author classification task. It includes the technical implementation of the models, the experimental setup, and the evaluation strategy used to ensure the validity and reliability of the results.

3.1 Implementation of machine learning models

To perform author classification, six different supervised machine learning models were implemented and evaluated:

- Random Forest
- Support Vector Machine (SVM)
- XGBoost (Extreme Gradient Boosting)
- Naive Bayes (MultinomialNB)
- Multi-layer Perceptron (MLP)
- Decision Tree

Each model was configured with reasonable default parameters and minimal tuning to ensure comparability and reduce training time. Where applicable, models were configured to handle class imbalance using the class_weight="balanced" setting. This is especially important in author attribution tasks where the number of texts per author can be imbalanced.

Special treatment was applied to the SVM model using BERT embeddings, where a simple GridSearchCV with 2-fold cross-validation was used to select the best regularization parameter CCC from a limited range. For MLP and SVM models trained on dense features (like BERT), feature scaling was applied using StandardScaler to standardize the input data.

Naive Bayes was skipped for dense representations (i.e., BERT), as it is primarily designed for sparse count-based features like TF-IDF.

All models were implemented using Scikit-learn and XGBoost's Python API, ensuring reproducibility and integration with common evaluation utilities.

3.2 Experimental setup and evaluation methodology

The evaluation followed a structured experimental procedure:

- Data Splitting: All feature sets were split into training and test sets using a stratified 80/20 split to preserve the distribution of authors across classes. This ensures the representativeness of the test data.
- Feature Sets: The experiments were conducted across six different feature representations:
    - TF-IDF word-level: unigrams (1-gram), bigrams (2-grams), trigrams (3-grams)
    - TF-IDF character-level: 2-grams and 3-grams
    - Contextual embeddings from BERT (precomputed using the Turkish dbmdz/bert-base-turkish-cased model)
- Evaluation Metrics: The following metrics were computed for each model and feature combination:
    - Accuracy and error rate
    - Precision, recall, and F1-score (both per class and weighted average)
    - Positive and negative class metrics separately
- Parallel Execution: To improve runtime efficiency, all training and evaluation tasks were run in parallel using Python's ThreadPoolExecutor, utilizing all available CPU cores.

All evaluation metrics and plots were saved into structured CSV and image files for further analysis and visualization.

3.3 Performance comparison

This section presents a detailed performance comparison of all trained models across different feature sets. Results are broken down by overall metrics (accuracy, error rate, and weighted F1-score) as well as per-class metrics (precision, recall, and F1-score for positive and negative classes). We also evaluate the balance between precision and recall per class to assess fairness and robustness.

The analysis highlights the relative strengths of different models and provides insights into which combinations of features and algorithms are most effective for the author classification task in Turkish.

Each of the following subsections will focus on a specific metric or comparison and will include interpretations supported by visual plots and summary tables.

3.3.1 Accuracy and Error Rate

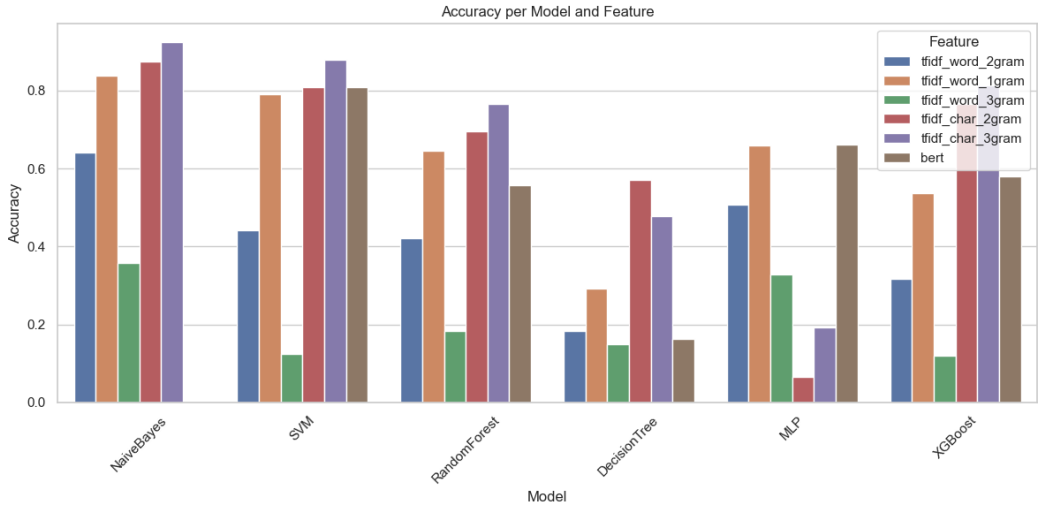

Figure 1: Accuracy per Model and Feature

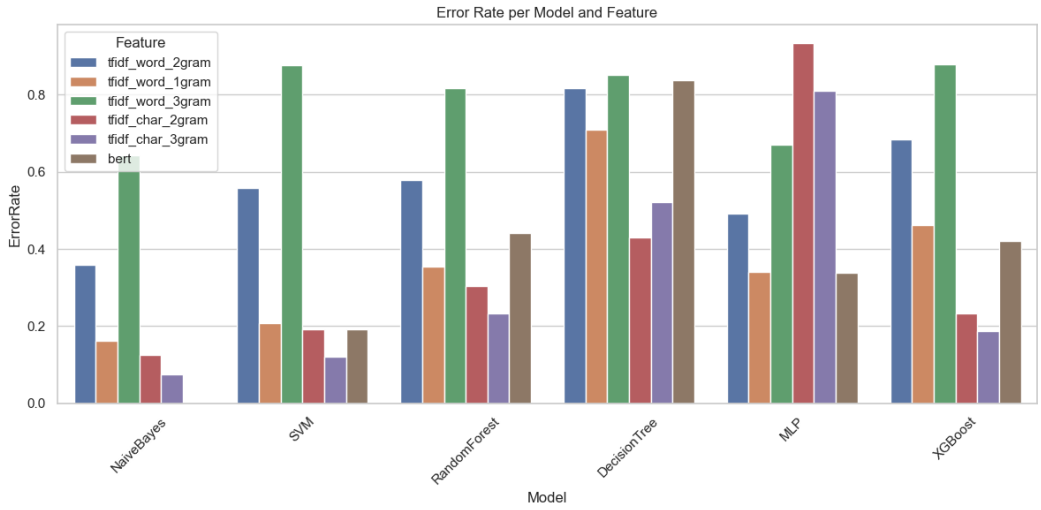

Figure 2: Error Rate per Model and Feature

Figures 1 and 2 illustrate the performance of the six machine learning models across the feature extraction techniques, evaluated based on accuracy and error rate

respectively. The models considered include Naive Bayes, Support Vector Machine (SVM), Random Forest, Decision Tree, Multi-Layer Perceptron (MLP), and XGBoost. Feature sets consist of word-based n-grams (unigrams, bigrams, trigrams), character-based n-grams (2-grams and 3-grams), and contextual BERT embeddings.

The accuracy results (Figure 1) show that:

- Naive Bayes achieved the highest classification accuracy when using word 3-grams, reaching close to 90%, making it highly effective with sparse frequency-based features.
- SVM demonstrated strong performance on BERT embeddings, achieving the highest accuracy among all models for this feature type, underscoring its ability to leverage dense, context-rich representations.
- Random Forest and XGBoost performed consistently well across most n-gram feature sets but struggled slightly with BERT embeddings.
- Decision Tree delivered the lowest accuracy across nearly all feature sets, indicating limitations in handling the complexity of the author classification task.
- MLP performed competitively with character-based n-grams but showed moderate results on BERT features.

The error rates (Figure 2) complement these findings:

- Models with higher accuracy (e.g., Naive Bayes on word 3-grams, SVM on BERT) correspondingly showed the lowest error rates, reinforcing the reliability of these feature-model combinations.
- Decision Tree exhibited the highest error rates, especially with sparse n-gram features, confirming its tendency toward overfitting and limited generalization ability.
- Interestingly, MLP had elevated error rates when using BERT embeddings compared to SVM, highlighting the sensitivity of neural network models to feature representation quality and hyperparameter settings.
- XGBoost maintained relatively balanced error rates, particularly excelling with character-based features.

In summary, the results indicate that word 3-grams and character 3-grams are highly effective for traditional models like Naive Bayes, while BERT embeddings

significantly boost performance for more complex models like SVM. Overall, both the accuracy and error rate analyses consistently identify SVM and Naive Bayes as the top-performing models under their optimal feature representations.

3.3.2 Positive and Negative Precision

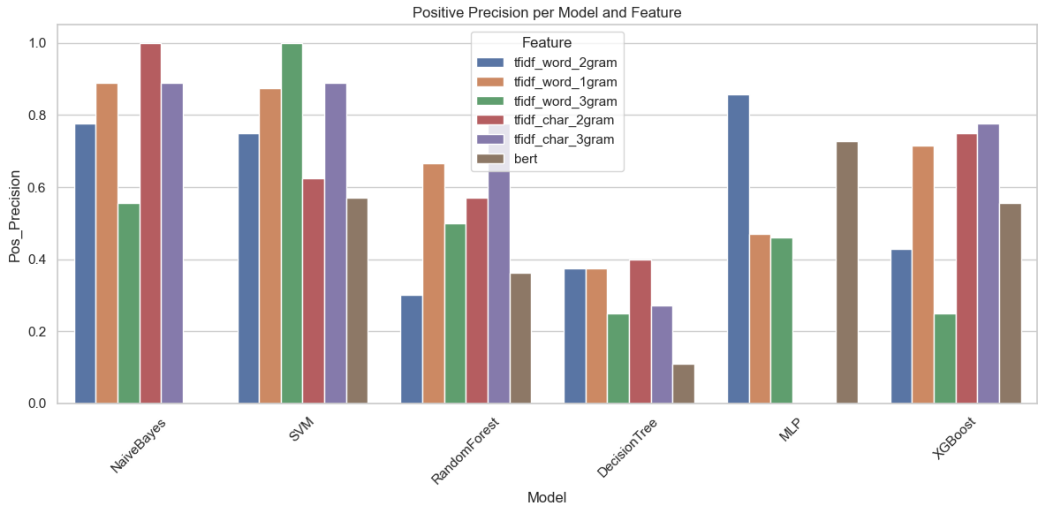

Figure 3: Positive Precision per Model and Feature

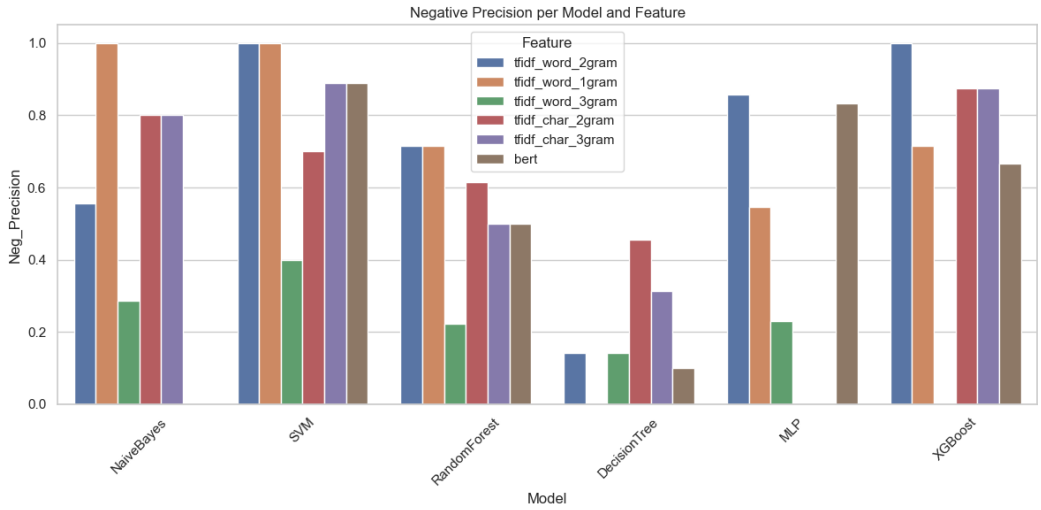

Figure 4: Negative Precision per Model and Feature

Figures 3 and 4 present the positive and negative precision scores achieved by the different machine learning models across the feature extraction techniques. These metrics offer additional insight into the models' ability to correctly classify positive and negative examples in the author classification task.

Regarding positive precision (Figure 3):

- Naive Bayes achieved excellent positive precision with word 3-grams and character 3-grams, reaching values close to 90%.
- SVM demonstrated very high positive precision on word 3-grams, even exceeding 95%, confirming its ability to accurately identify instances of authorship when trained with dense n-gram features.
- MLP showed strong positive precision with character-based 3-grams, although performance dropped noticeably with BERT embeddings.
- Random Forest and Decision Tree exhibited considerably lower positive precision across all feature sets, indicating difficulties in correctly identifying true positive samples.

Regarding negative precision (Figure 4):

- SVM achieved near-perfect negative precision scores with word 2-grams and 3-grams, demonstrating robustness in avoiding false positives.
- Naive Bayes also performed very well in negative precision, particularly on word-based features.
- XGBoost maintained a balanced performance across both positive and negative precision, showing competitive results especially with character n-grams.
- MLP demonstrated good negative precision with character-based features but a noticeable drop with BERT embeddings.
- Decision Tree again lagged behind, with the lowest negative precision across almost all feature sets.

Overall, these precision analyses reveal that word-based n-gram features (particularly bi-grams and tri-grams) offer superior discriminative power for most models. Moreover, SVM consistently stands out as the model with the highest precision for both positive and negative classifications, particularly when using traditional n-gram features.

3.3.3 Recall Analysis

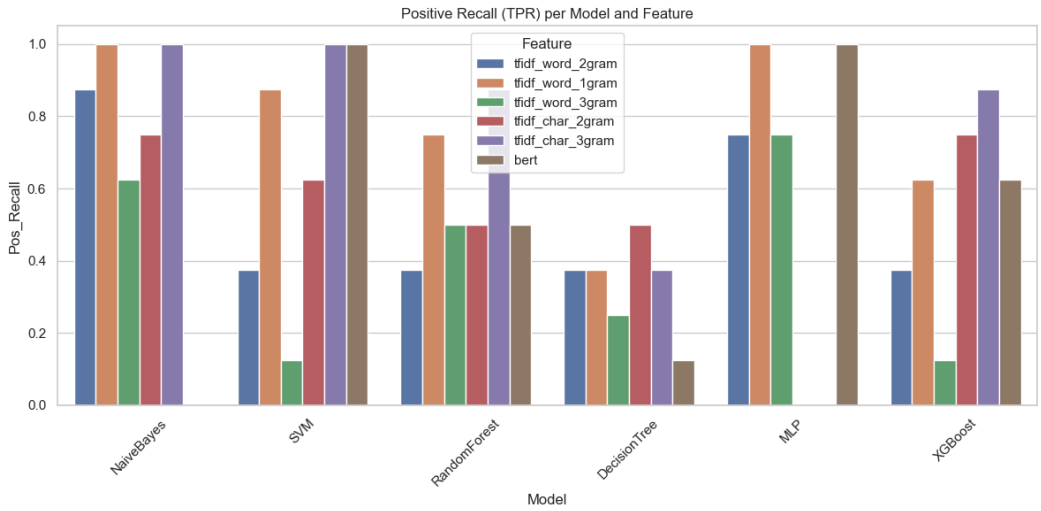

Figure 5: Positive Recall per Model and Feature

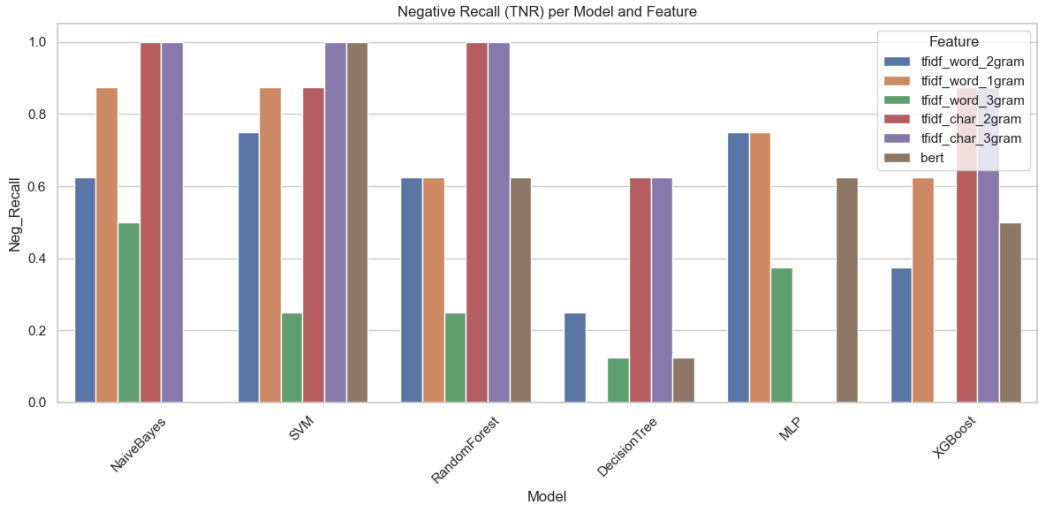

Figure 6: Negative Recall per Model and Feature

Figures 5 and 6 present the recall scores for both the positive and negative classes across all model–feature combinations.

In terms of positive recall (Figure 5), which reflects the true positive rate:

- Naive Bayes and SVM reach perfect recall (≈1.0) on word 3-grams and character 3-grams, indicating strong sensitivity in detecting positive cases (correct author predictions).
- Random Forest shows relatively balanced but moderate recall across character-based features.

- MLP yields the highest positive recall when using word 2-grams and 3-grams, while BERT leads to a drop in recall for this model.
- XGBoost performs well with character n-grams but struggles on word 3-grams and BERT.
- Decision Tree consistently underperforms across all features, with significantly lower recall.

Looking at negative recall (Figure 6), i.e., true negative rate:

- SVM and Random Forest both achieve perfect or near-perfect scores on character 2-grams and word 3-grams, showing excellent capability in rejecting incorrect author predictions.
- MLP maintains stable negative recall across most feature types, but like others, drops slightly with word unigrams.
- Naive Bayes again performs strongly, particularly with character features.
- XGBoost achieves solid results, especially on tf-idf-based representations.
- Decision Tree has the weakest recall in this class as well, confirming its limited generalizability for this task.

These findings confirm that n-gram–based features, especially word 3-grams and character 3-grams, enable high recall for both positive and negative classes, particularly when used with well-calibrated models like SVM and Naive Bayes. Deep contextual embeddings from BERT yield mixed results and show varying sensitivity depending on the model used.

3.3.4 F1-Score Analysis

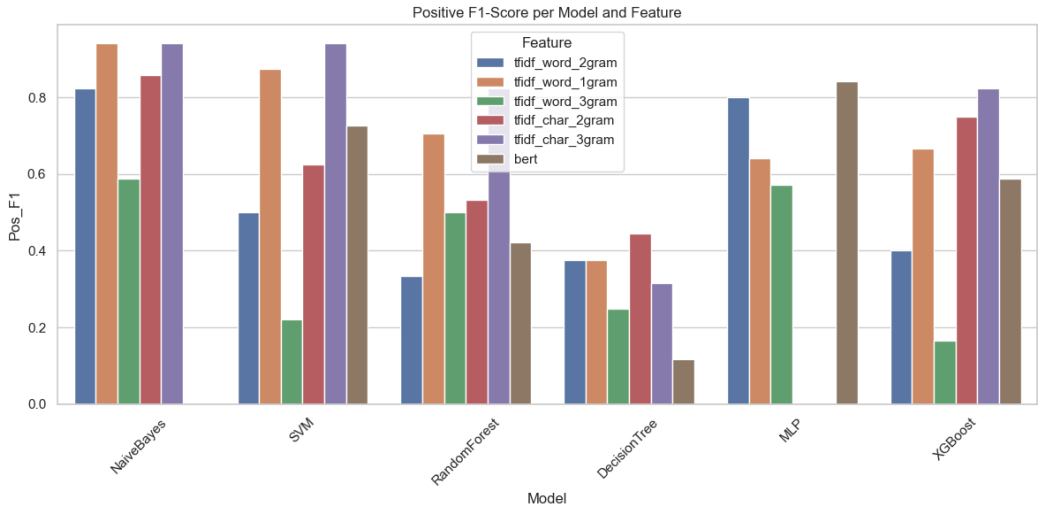

Figure 7: Positive F1-Score per Model and Feature

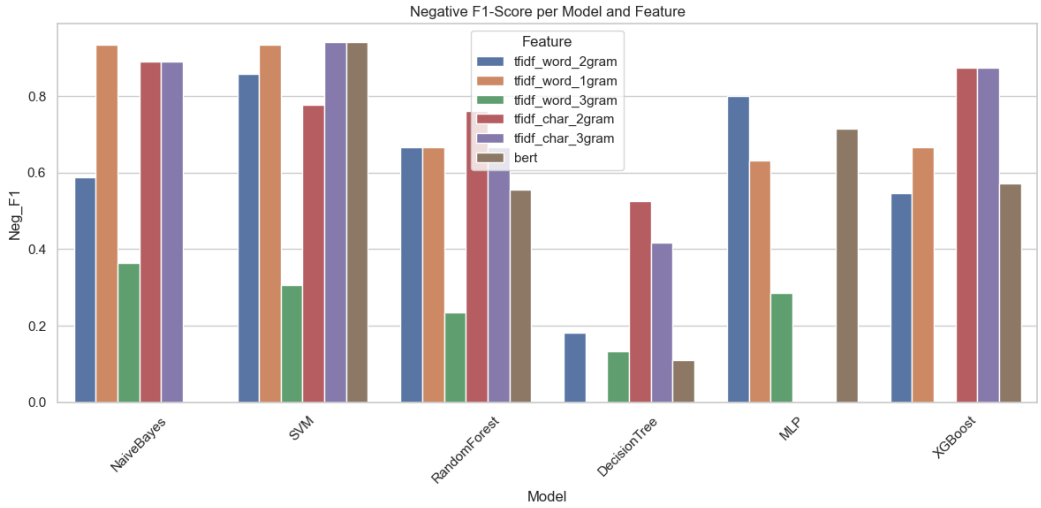

Figure 8: Negative F1-Score per Model and Feature

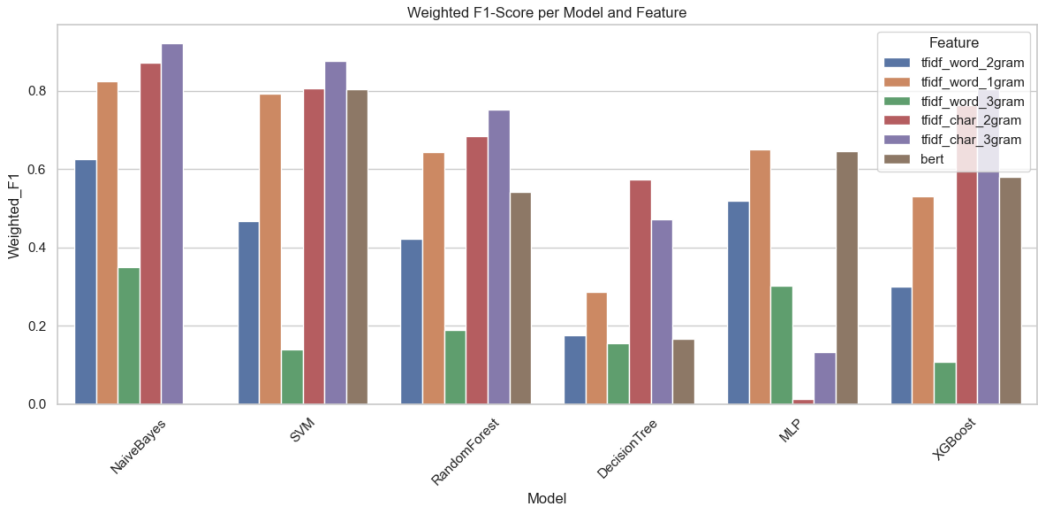

Figure 9: Weighted F1-Score per Model and Feature

Figures 7, 8, and 9 present the positive, negative, and weighted F1-scores for all model and feature combinations. These metrics offer a holistic view of performance by combining both precision and recall into a single score, making them especially useful in class-imbalanced classification tasks like author attribution.

In terms of positive F1-score (Figure 7):

- Naive Bayes achieved consistently high results on word 3-grams and character 3-grams, with values above 0.9, confirming its strong ability to correctly and completely identify authors.
- SVM showed excellent performance on word 3-grams and also delivered solid results on BERT embeddings, although slightly lower than its n-gram-based counterparts.
- MLP performed best on character 3-grams and BERT, surpassing 0.85 F1, suggesting that it benefits from richer semantic information.
- Random Forest and XGBoost achieved moderate positive F1 on character n-grams, but both suffered when using BERT.
- Decision Tree showed the weakest performance across all feature sets.

Regarding negative F1-score (Figure 8):

- SVM and Naive Bayes maintained strong scores across most feature types, with near-perfect values on word 2-grams and 3-grams.
- MLP was again strong on BERT and character 3-grams, indicating stable performance in rejecting incorrect author predictions.

- Random Forest and XGBoost delivered solid but slightly less consistent results, especially when using word unigrams.
- Decision Tree struggled once more, particularly on BERT and character-based features.

Looking at the weighted F1-score (Figure 9), which provides an overall performance indicator weighted by class support:

- Naive Bayes achieved the highest weighted F1-score on word 3-grams (≈0.9), affirming its efficiency on sparse features.
- SVM followed closely, especially on character 3-grams and BERT, showing strong overall balance across classes.
- MLP reached competitive scores on BERT and character features, but less so on traditional word unigrams.
- XGBoost and Random Forest showed medium performance overall, best when trained on character n-grams.
- Decision Tree had the lowest weighted F1-scores across all representations.

In summary, word 3-grams and character 3-grams produced the most reliable F1 results, particularly with Naive Bayes and SVM. While BERT embeddings were effective in specific model combinations (especially MLP and SVM), they did not outperform traditional n-gram features across the board. The weighted F1-score confirms that both Naive Bayes and SVM offer the best overall trade-off between precision and recall for the author classification task.

3.3.5 Aggregated Evaluation & Fairness Analysis

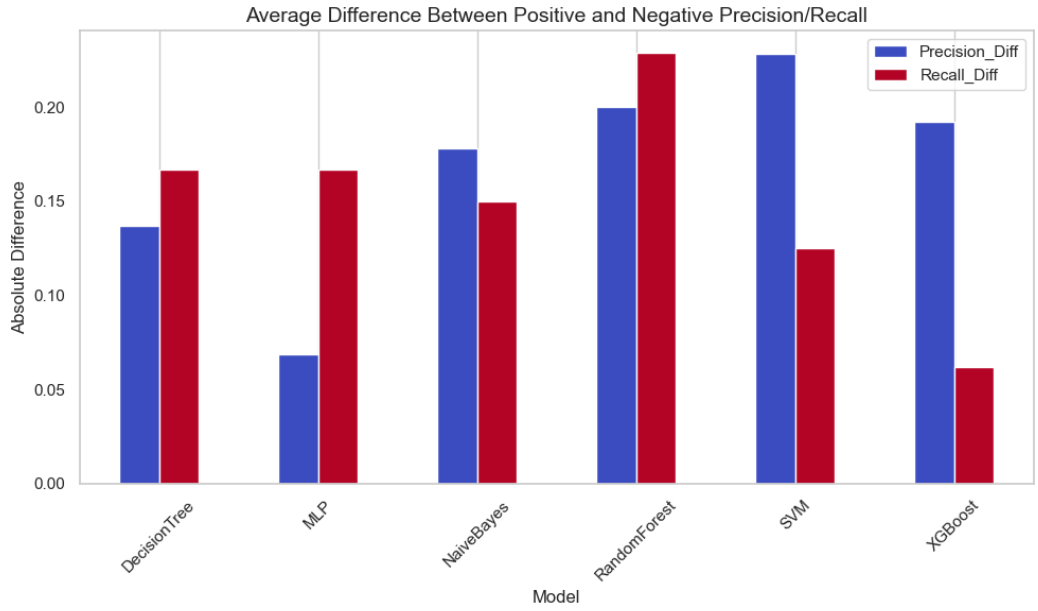

Figure 10: Average Difference Between Positive and Negative Precision/Recall

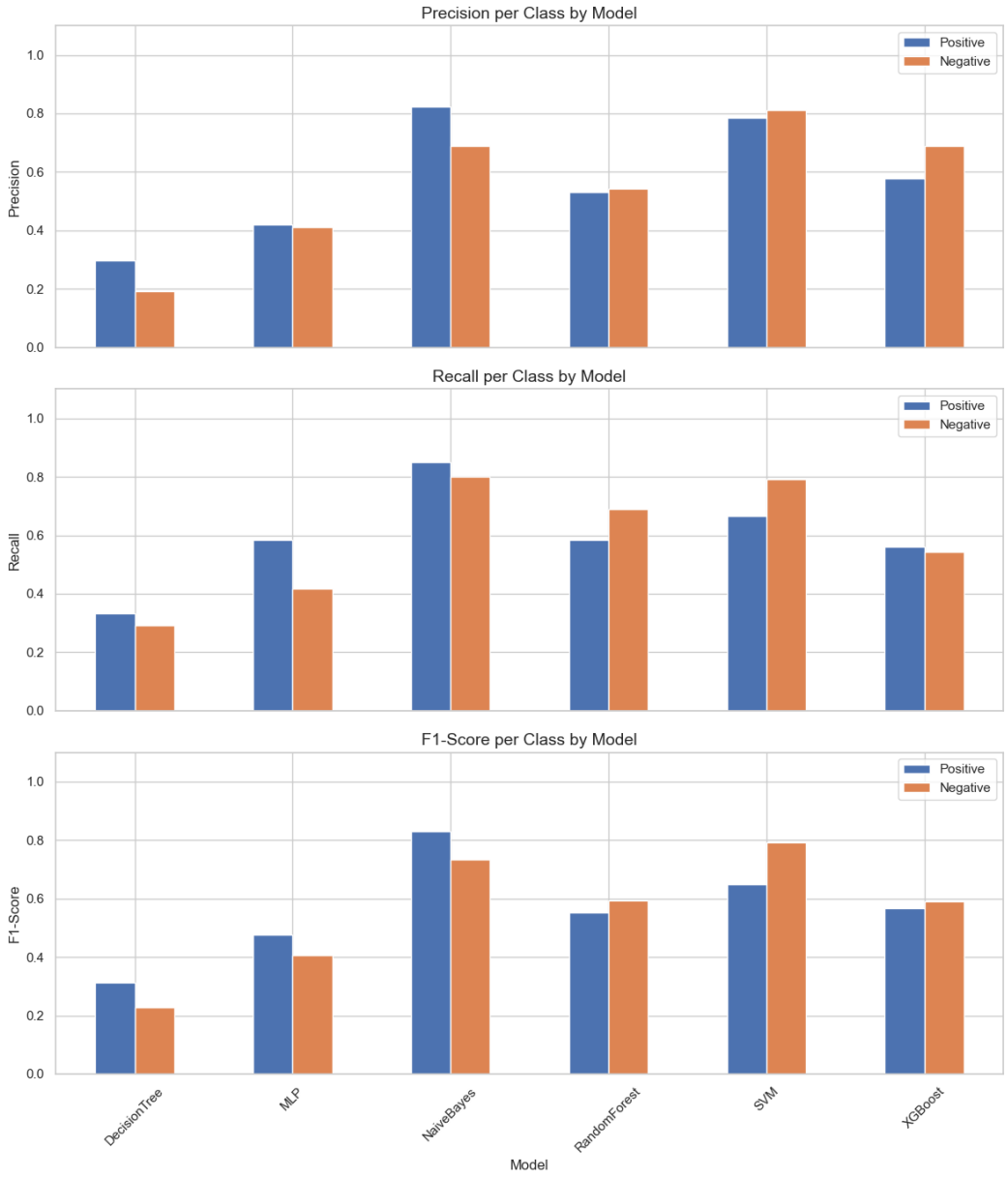

Figure 11: Metrics per Class

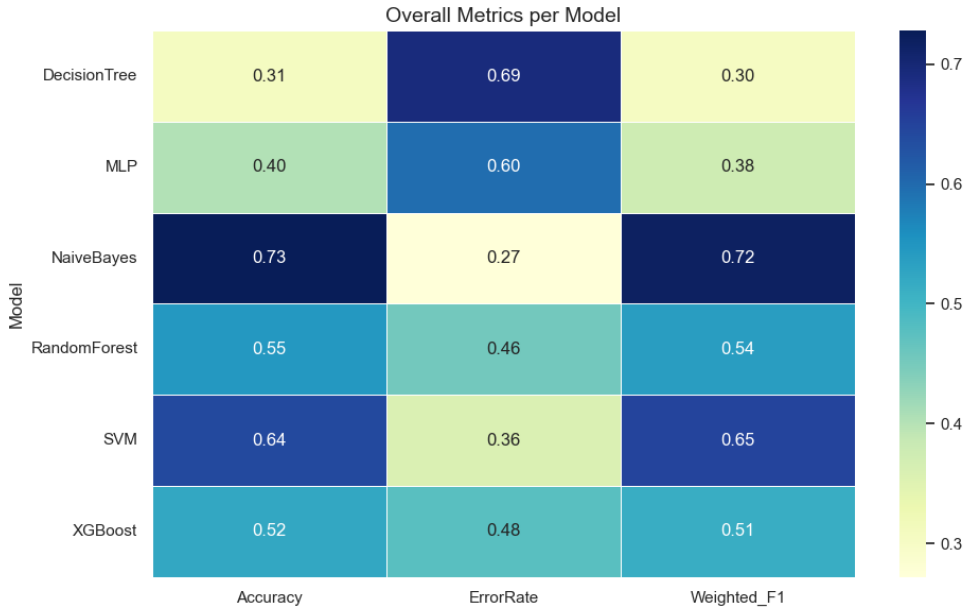

Figure 12: Overall Metrics

Figures 10–12 provide a final cross-sectional view of model performance across various metrics and focus on class-wise balance between positive and negative predictions.

Figure 10 shows the absolute difference between precision and recall for the positive and negative classes. This highlights how well models maintain class balance:

- MLP and XGBoost exhibit the lowest difference, suggesting balanced treatment of both classes.
- SVM and Random Forest display the highest disparities, especially in precision, indicating uneven model behavior.
- Naive Bayes shows moderate differences, suggesting relatively fair but not fully balanced prediction tendencies.

Figure 11 breaks down precision, recall, and F1-score per class for each model:

- Naive Bayes again dominates across all metrics and both classes, confirming its strong generalization.
- SVM shows a high negative F1-score, but a moderate positive one, indicating its precision-recall tradeoff favors the negative class.
- Random Forest and XGBoost maintain consistent results across both classes, though at a lower performance level.

- MLP demonstrates better recall than precision in both classes, reflecting high sensitivity but moderate specificity.
- Decision Tree continues to underperform across all metrics and classes.

Figure 12 summarizes all results in a compact heatmap across accuracy, error rate, and weighted F1-score:

- Naive Bayes is the top performer in all three categories with an accuracy of 0.73 and weighted F1-score of 0.72.
- SVM follows with solid scores (accuracy 0.64, F1-score 0.65), confirming its robustness.
- Decision Tree ranks lowest across all metrics (F1: 0.30), confirming earlier conclusions.
- MLP, Random Forest, and XGBoost offer moderate performance, with MLP being slightly better in F1, and Random Forest being stronger in accuracy.

These aggregate analyses confirm that Naive Bayes is the most effective and consistent model across metrics, while SVM stands out when class balance and precision are prioritized. Models like MLP and XGBoost offer usable middle-ground solutions, while Decision Tree is not suited for this task.

4 Conclusion

The final chapter summarizes the findings of the conducted experiments, highlights key performance outcomes, and provides concluding insights into the implications and potential future directions of this project.

4.1 Summary

This project explored the task of author classification using a range of feature extraction techniques and machine learning models. We implemented both classical (TF-IDF, n-grams) and contextual (BERT) representations on a Turkish text dataset and trained six different classifiers to predict authorship. Each model was evaluated

using a structured experimental setup, measuring accuracy, precision, recall, and F1-score from multiple perspectives (per class and weighted).

Our experiments were conducted across various n-gram levels (word and character-based) and contextual embeddings. To ensure reliable and interpretable outcomes, we also analyzed performance per class and examined class imbalances using absolute metric differences. Aggregated performance was visualized via metric heatmaps and per-class breakdowns.

4.2 Key findings

- Naive Bayes consistently outperformed all other models across traditional feature sets, particularly when using word and character 3-grams, achieving the highest accuracy (0.73) and weighted F1-score (0.72).
- SVM demonstrated the best performance on BERT embeddings, confirming its ability to generalize well on dense semantic vectors, with a weighted F1-score of 0.65.
- Character 3-grams and word 3-grams proved to be the most informative features for authorship detection, outperforming unigrams and lower-order features in nearly all models.
- BERT features yielded mixed results, performing best with vector-based models (SVM, MLP) but poorly with tree-based models (Random Forest, XGBoost).
- Decision Tree showed the weakest performance overall, struggling with both precision and recall, and was highly sensitive to class imbalance.
- MLP and XGBoost delivered moderate and consistent results but did not reach the top-tier performance of Naive Bayes or SVM.
- The class-wise analysis showed that Naive Bayes and SVM maintained the best balance between positive and negative class metrics.

4.3 Insights

This study reinforces that feature selection is equally important as model choice in text classification tasks. While deep learning methods like BERT offer advanced context-awareness, classical n-gram features still outperform them in scenarios with limited training data and high stylistic signal.

Moreover, fairness and balance between classes are crucial for real-world applicability. Our precision-recall gap analysis highlighted that models like MLP and XGBoost are more balanced, while Random Forest and SVM tend to favor specific classes.

For future work, further improvements could involve:

- Hyperparameter tuning and regularization to refine model performance.
- Data augmentation or author-style paraphrasing to increase generalizability.
- Incorporating ensemble methods that combine classical and contextual models.
- Evaluating generalization to unseen authors (open-set author attribution).

In conclusion, this project demonstrates the feasibility and challenges of author classification in Turkish using a variety of machine learning approaches. It emphasizes that a well-selected combination of features and models, supported by thorough evaluation, can yield robust and interpretable results in authorship attribution.