

# **DTI5125: Data Science Applications**



uOttawa

## **Assignment 1:**

### **A Tale of Text Classification**

**By:**

**Group 11**

Chandranshu Verma - 300372673

Omkar Harkar - 300359531

Jainil Rana - 300362571

Kishita Pakhrani - 300324851

## **Introduction**

In the realm of text classification, this report navigates through the intricate landscape of predicting authorship and genre of Gutenberg digital books. Six diverse literary works form our dataset, guiding us through the application of machine learning algorithms and feature engineering. Our journey unfolds with meticulous data preparation, preprocessing, and feature engineering, leading to an in-depth exploration of models like Naive Bayes, Logistic Regression, SVM, Decision Tree, Random Forest, Gradient Boosting, and k-Nearest Neighbors. The report concludes with insights gleaned from error analysis, visualizations, and a discerning selection of the champion model, offering a comprehensive understanding of the challenges in text classification.

## **Data and Data Preprocessing**

Let's dive into how we gathered and prepped our data for this exciting journey through the realms of literature and technology.

### **Data Selection**

For this classification task, six books are randomly chosen from the Gutenberg digital library. The aim is to ensure a diverse set of genres and authors. This random selection helps in capturing variations in writing styles, genres, and themes.

### **Data Partitioning**

Each selected book undergoes a partitioning process to create segments for training, validation, and testing. The book is split into 200 segments, and each segment contains 100 words. This ensures a sufficiently large dataset for training while maintaining randomness and avoiding biases in data allocation.

### **Data Shuffling**

Once the books were selected, the data was partitioned into unbiased random subsets for training, validation, and testing. This random partitioning strategy is crucial as it prevents the model from learning specific patterns related to the ordering of the data. Each partition was then labeled

alphabetically ('a', 'b', 'c', ...) based on the book it belonged to. This labeling system serves as a key identifier during training, enabling easy tracking and analysis of the model's performance on specific books.

The labeled partitions, consisting of book labels, author information, and text segments, are serialized into a CSV file. This file acts as the main source for subsequent processing steps. The `GutenbergPartitioner` class is responsible for managing this data serialization.

### Tokenization:



## **Stopword Removal**

Stopwords, common words like "the" and "and," are often not informative for text classification tasks. Removing these stopwords helps in focusing on more meaningful words, improving the efficiency of the analysis.

## **Feature Engineering**

Feature engineering is a critical aspect of text classification, as the choice of features significantly influences the model's ability to capture relevant patterns in the data. In this project, the primary feature extraction technique employed was Term Frequency-Inverse Document Frequency (TF-IDF). Here, we delve into an explanation of TF-IDF and its exclusive use in the feature engineering process.

## **Part-of-Speech (POS) Tagging**

POS tagging is employed to assign grammatical categories (such as nouns, verbs, adjectives) to each word in the text. This information is valuable for understanding the syntactic structure of sentences. NLTK's POS tagging is used to enhance the richness of features for subsequent analysis.

## **Stemming and Lemmatization**

Stemming and lemmatization are techniques applied to reduce words to their base or root form. Stemming involves removing suffixes to obtain the root form, while lemmatization involves transforming words to their dictionary form. This standardizes the text, reducing variations and aiding in feature extraction.

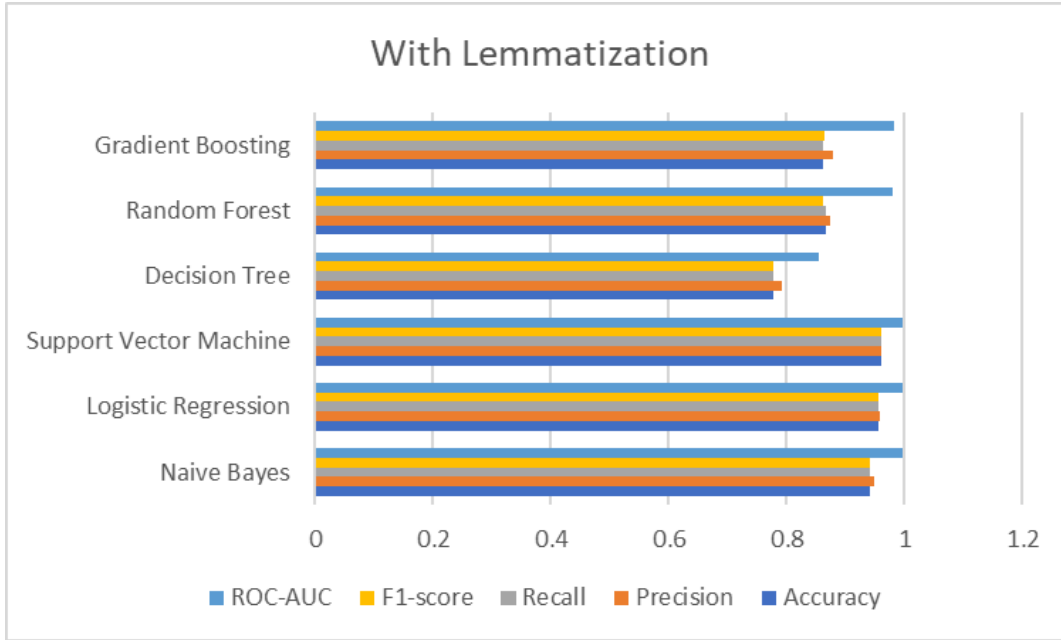


Fig 2. Model Accuracies with Lemmatized data

Stemming is performed using the Porter Stemmer, and lemmatization is conducted using WordNet Lemmatizer from the NLTK library.

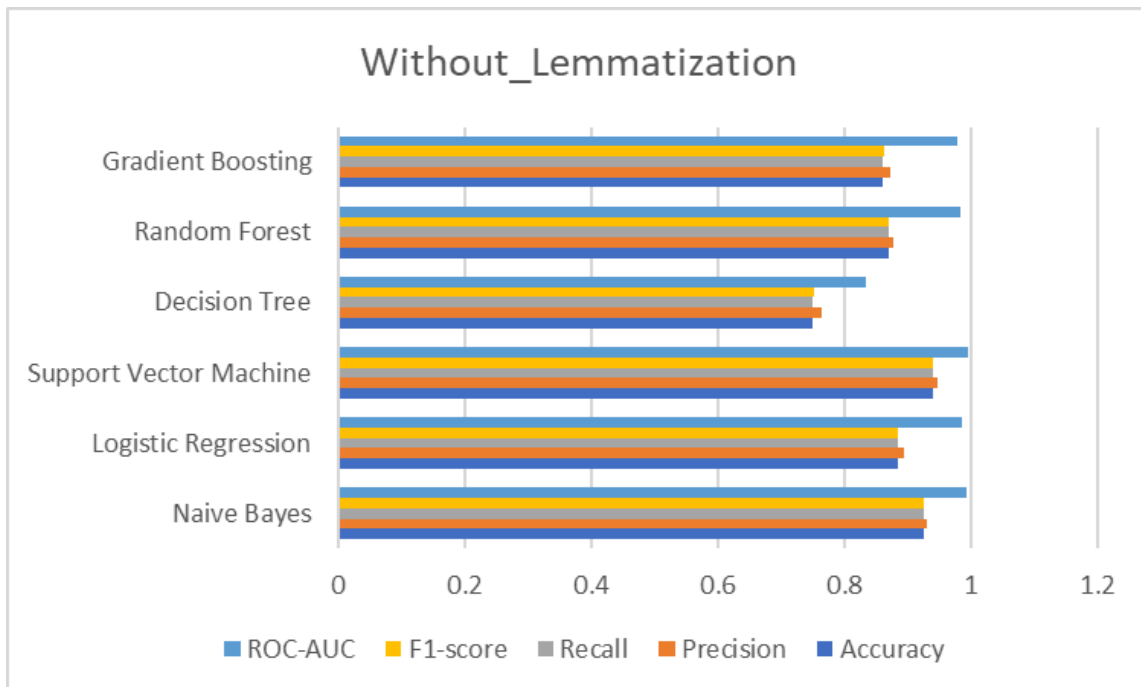


Fig 3. Model Accuracies without lemmatized data

As we can see in fig 2 and fig 3, using lemmatization specifically helps in increasing accuracy by ensuring that words are transformed to their canonical forms. This is important because words in different forms might convey the same meaning. For example, "running" and "ran" are lemmatized to "run," capturing the core semantics. Increased accuracy is achieved as the model generalizes better on the lemmatized representations.

Moreover, lemmatization contributes to managing variance in the dataset. Without lemmatization, the same word in different forms might be treated as distinct features, leading to increased variance. By lemmatizing, variations in word forms are reduced, providing a more stable and consistent dataset.

These preprocessing steps collectively contribute to cleaning and preparing the data for subsequent feature engineering and modeling stages. The processed data retains essential information while eliminating noise and irrelevant details.

### **TF-IDF Magic:**

#### **TF-IDF (Term Frequency-Inverse Document Frequency)**

**Term Frequency (TF):** This component of TF-IDF measures the frequency of a term (word) within a specific document. It reflects how often a word occurs in a particular text relative to the total number of words in that text. Higher TF values indicate that a term is more prevalent in a given document.

**Inverse Document Frequency (IDF):** IDF evaluates the uniqueness or rarity of a term across all documents in the dataset. Terms that appear frequently across multiple documents receive lower IDF scores, while terms appearing in a limited set of documents receive higher scores. This helps in distinguishing terms that carry distinctive information.

**TF-IDF:** The product of TF and IDF yields the TF-IDF score, a numerical representation of a term's importance in a document relative to the entire dataset. TF-IDF effectively highlights terms that are both frequent within a document and distinctive across the entire corpus.

## **Why TF-IDF:**

**Weighting Relevance:** TF-IDF assigns higher weights to terms that are both frequent and unique to specific documents, emphasizing their relevance in distinguishing between documents.

**Dimensionality Reduction:** By focusing on important terms and downplaying common ones, TF-IDF naturally reduces the dimensionality of the feature space. This is crucial for computational efficiency and preventing the model from being overly influenced by less informative terms.

**Language Agnosticism:** TF-IDF is language-agnostic, making it suitable for diverse datasets. It assesses the importance of terms based on their distribution across documents rather than relying on language-specific rules.

## **Experimentation and Rationale:**

The decision to exclusively use TF-IDF as the feature engineering technique was driven by its effectiveness in handling textual data, especially in scenarios with limited computational resources. The method provides a concise yet informative representation of documents, capturing both local and global term importance.

While other techniques such as word embeddings (e.g., Word2Vec, GloVe) and n-grams could have been explored, TF-IDF sufficed for this project's objectives. Word embeddings require extensive computational resources and large datasets for training, and n-grams might introduce high dimensionality and sparsity issues, making them less suitable for this specific context.

In conclusion, the feature engineering strategy centered on the TF-IDF technique, leveraging its ability to highlight relevant terms and reduce dimensionality effectively. This choice aligns with the project's goals of achieving accurate and interpretable text classification results.

## **Model Training and Evaluation:**

In the pursuit of effective text classification, the choice of algorithms assumes paramount importance. This section delves into the algorithms implemented for the project, evaluates their performances, and identifies the champion model.

### **Naive Bayes**

The Naive Bayes algorithm, known for its computational simplicity and ease of implementation, was initially considered for text classification. While it excels in processing large datasets, its fundamental assumption of feature independence might limit its effectiveness in capturing complex relationships within text data.

### **Logistic Regression**

Logistic Regression, offering probability outputs and adeptness in handling non-linear relationships, was another candidate. Its simplicity and interpretability make it a popular choice, though its performance may be overshadowed by more complex algorithms in certain scenarios.

### **Support Vector Machine (SVM)**

Support Vector Machine (SVM), acknowledged for its prowess in high-dimensional spaces and adaptability through various kernel functions, emerged as a strong contender. Its ability to capture intricate relationships within the data and perform well across diverse textual genres led to its selection as the champion model.

### **Decision Trees and Random Forests**

Decision Trees provide interpretability, breaking down decisions into simple if-else conditions. Random Forests, an ensemble of decision trees, enhance accuracy and mitigate overfitting. However, their interpretability diminishes compared to individual decision trees.



## **Gradient Boosting**

Gradient Boosting, recognized for high accuracy by sequentially training weak models, was also considered. Its capacity to adapt to errors and produce a robust final model makes it a powerful choice.

## **K-Nearest Neighbor (KNN)**

K-Nearest Neighbor (KNN), a non-parametric approach relying on the proximity of data points, provided an alternative methodology. However, its effectiveness can diminish with high-dimensional data and large datasets.

## **Evaluation Metrics**

In evaluating these models, a 10-fold cross-validation approach was employed, encompassing metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. The outcomes revealed nuanced variations in performance across the spectrum of algorithms.

## **Champion Model: Support Vector Machine (SVM)**

Upon rigorous evaluation, the Support Vector Machine (SVM) emerged as the champion model. Renowned for its effectiveness in high-dimensional spaces and versatility through kernel functions, SVM exhibited robust performance across diverse textual genres. The decision to anoint SVM as the champion model was anchored in its ability to capture intricate relationships within the data, coupled with its reliable performance.

## **Model Comparison**

Despite the advantages, SVM is not devoid of challenges. Its computational complexity, particularly with sizable datasets, warrants consideration. Additionally, sensitivity to noise necessitates meticulous preprocessing for optimal results. In conclusion, the selection of SVM as the champion model aligns seamlessly with the project's objectives of achieving heightened accuracy and generalizability in text classification. This choice underscores the significance of aligning algorithmic selections with the intrinsic characteristics of the dataset and the intricacies of the classification task.

## Error Analysis

A critical aspect of model evaluation is the meticulous analysis of errors, providing insights into the characteristics of data segments that challenge the classification model. This section delves into the error analysis conducted on the implemented text classification models, shedding light on the nuances that contributed to misclassifications.

### Confusion Matrix Visualization

To initiate the error analysis process, a confusion matrix was generated, visually presenting the model's performance across different genres. This matrix encapsulates true positive, true negative, false positive, and false negative predictions, offering a comprehensive overview of the model's efficacy in classifying each genre.

#### 1. Naive Bayes

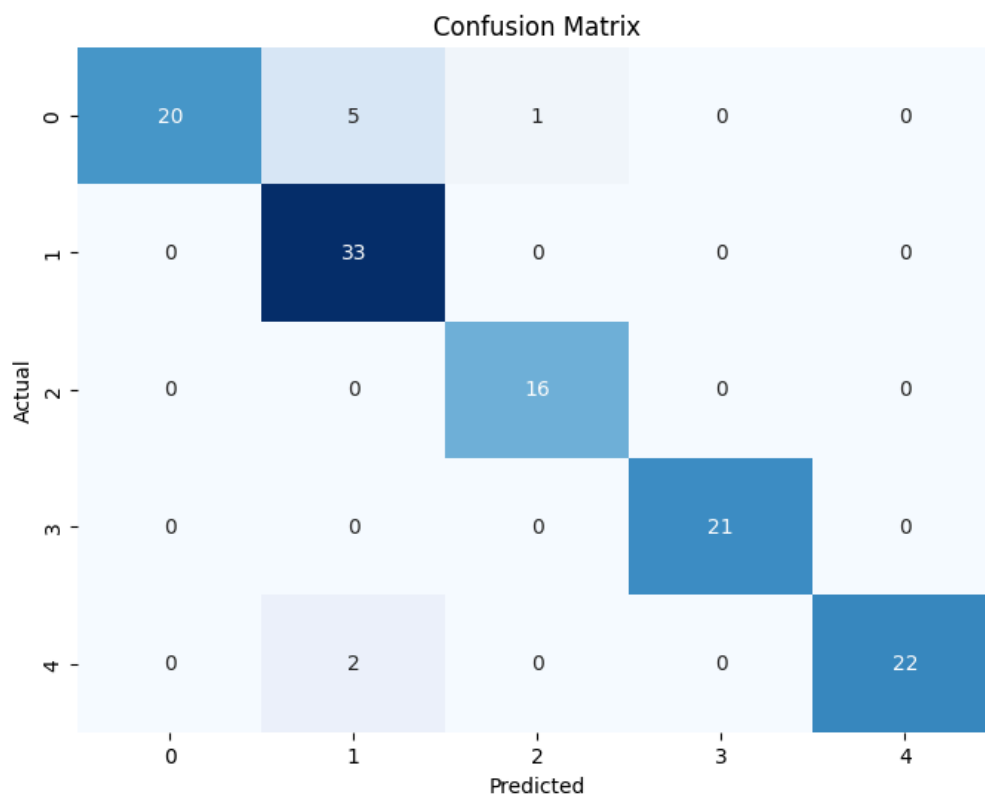


Fig 4. Confusion Matrix of Naive Bayes

```

Results for Naive Bayes:
Bias: 0.05833333333333324
Variance: 0.00015277777777778
Accuracy: 0.9416666666666668
Precision: 0.9471483036342333
Recall: 0.9416666666666668
F1-score: 0.9406009390808523
ROC-AUC: 0.9971655302180473

```

Fig 5. Results of Naive Bayes

## 2. Logistic Regression

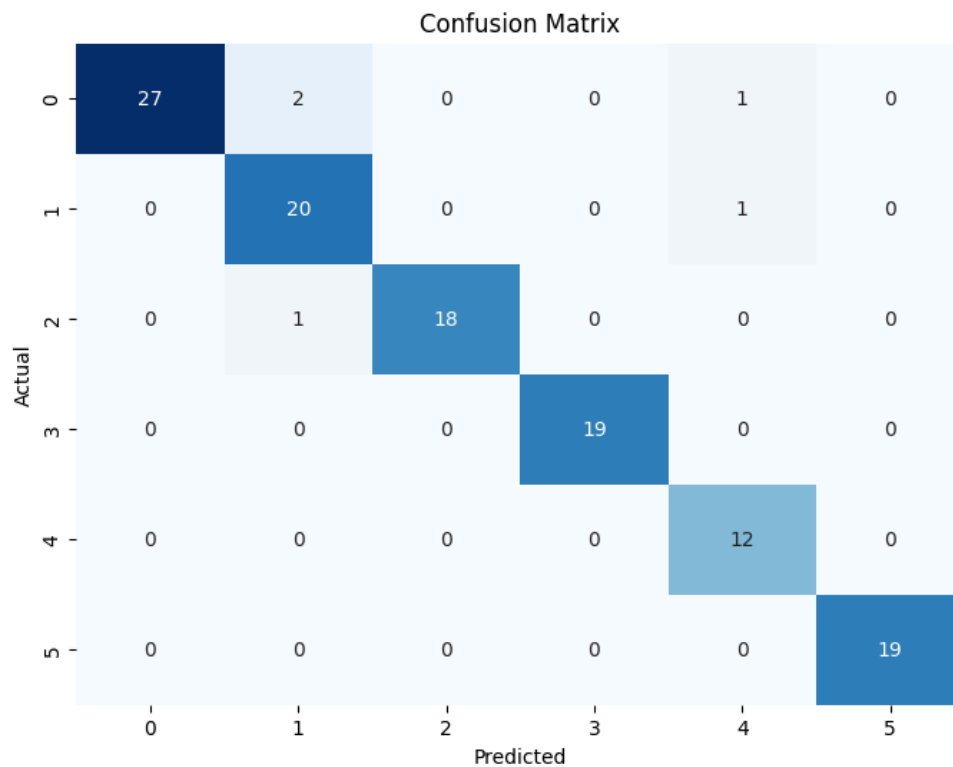


Fig 6. Confusion Matrix of Logistic Regression

```
Results for Logistic Regression:  
Bias: 0.054166666666666585  
Variance: 0.0004340277777777824  
Accuracy: 0.9458333333333334  
Precision: 0.9501056713685815  
Recall: 0.9458333333333334  
F1-score: 0.9459782288440254  
ROC-AUC: 0.9944071123641349
```

Fig 7. Results of Logistic Regression

### 3. SVM

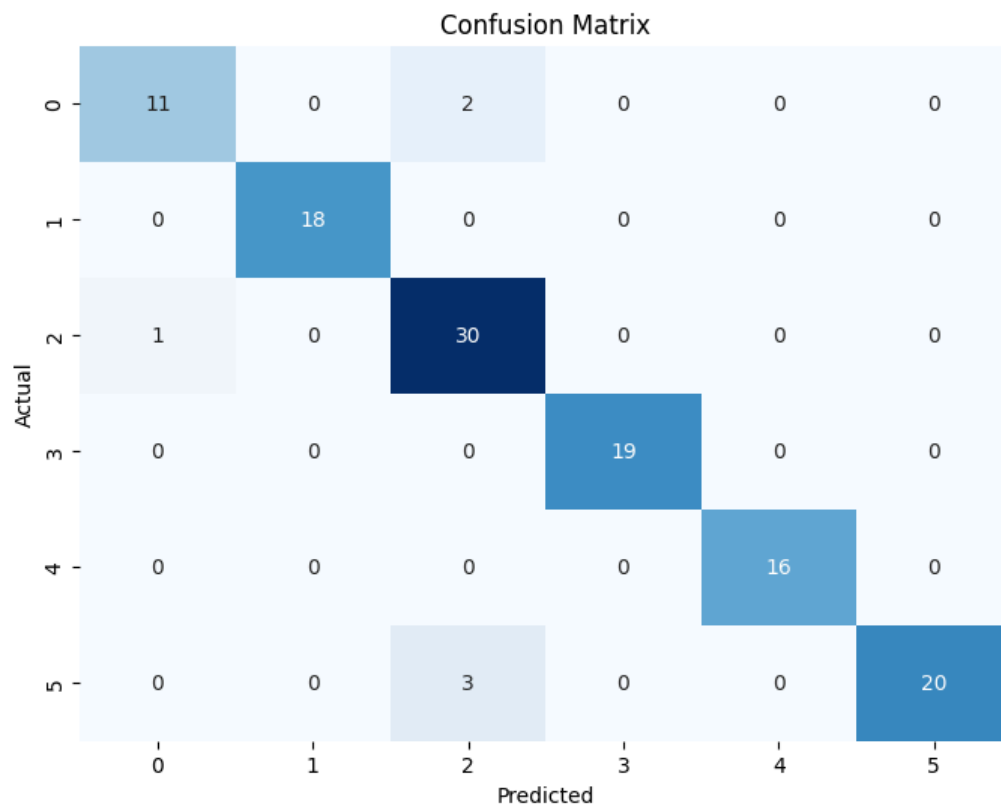


Fig 8. Confusion Matrix of SVM

```
Results for Support Vector Machine:  
Bias: 0.04333333333333356  
Variance: 0.00044166666666666643  
Accuracy: 0.9566666666666664  
Precision: 0.9606026861386558  
Recall: 0.9566666666666664  
F1-score: 0.9570997227623443  
ROC-AUC: 0.9975932974297175
```

Fig 9. Results of SVM

4. Decision Tree

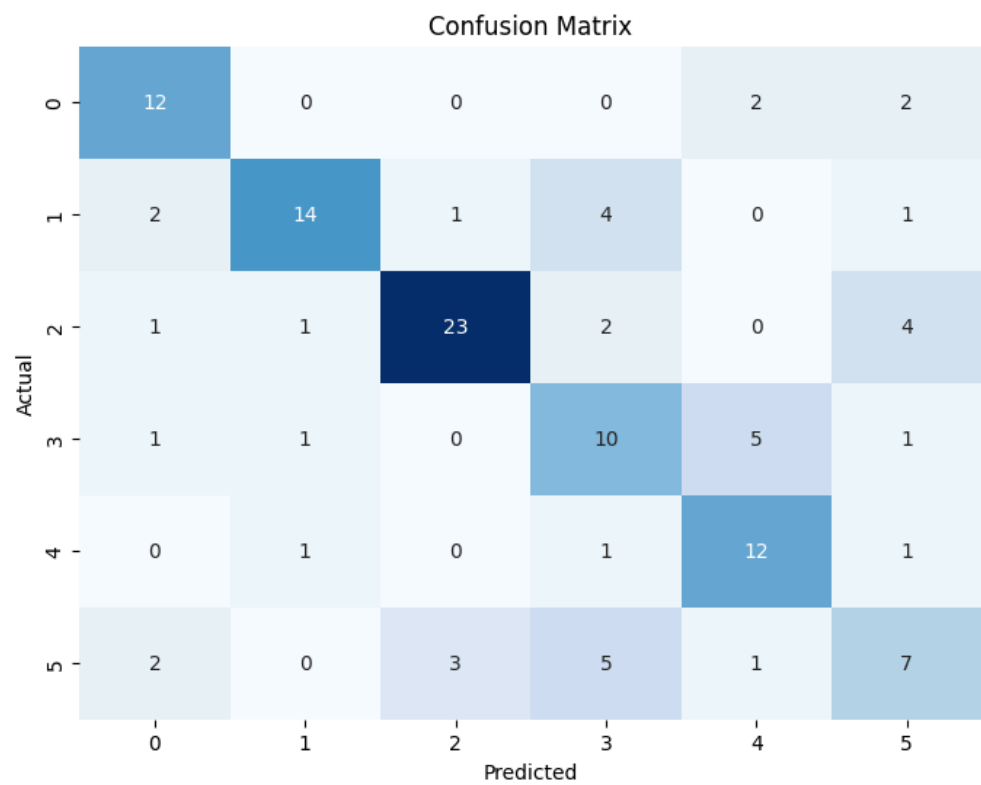


Fig 10. Confusion Matrix of Decision Tree

```

Results for Decision Tree:
Bias: 0.4116666666666674
Variance: 0.0022388888888888884
Accuracy: 0.5883333333333333
Precision: 0.6055504959760096
Recall: 0.5883333333333333
F1-score: 0.5888069262952988
ROC-AUC: 0.7553537097707959

```

Fig 11. Results of Decision Tree

## 5. Random Forest

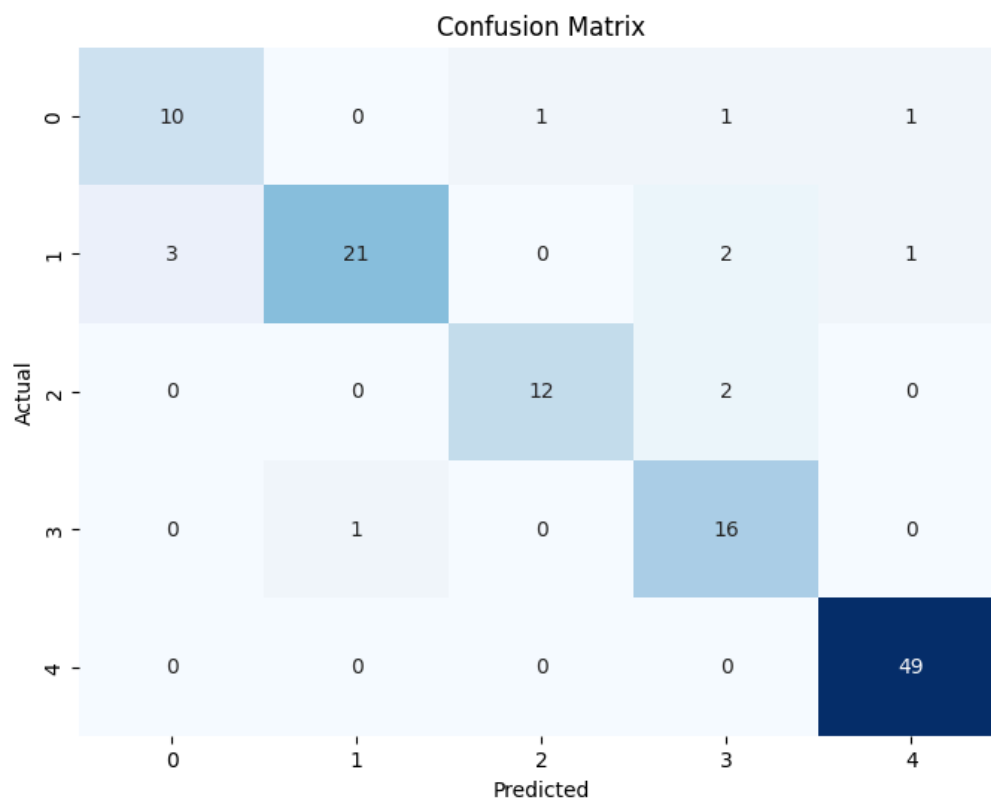


Fig 12. Confusion Matrix of Random Forest

```
Results for Random Forest:  
Bias: 0.09416666666666662  
Variance: 0.000806250000000003  
Accuracy: 0.9058333333333334  
Precision: 0.9098504643896212  
Recall: 0.9058333333333334  
F1-score: 0.9050774331169086  
ROC-AUC: 0.9877991787980873
```

Fig 13. Results of Random Forest

## 6. Gradient Boosting

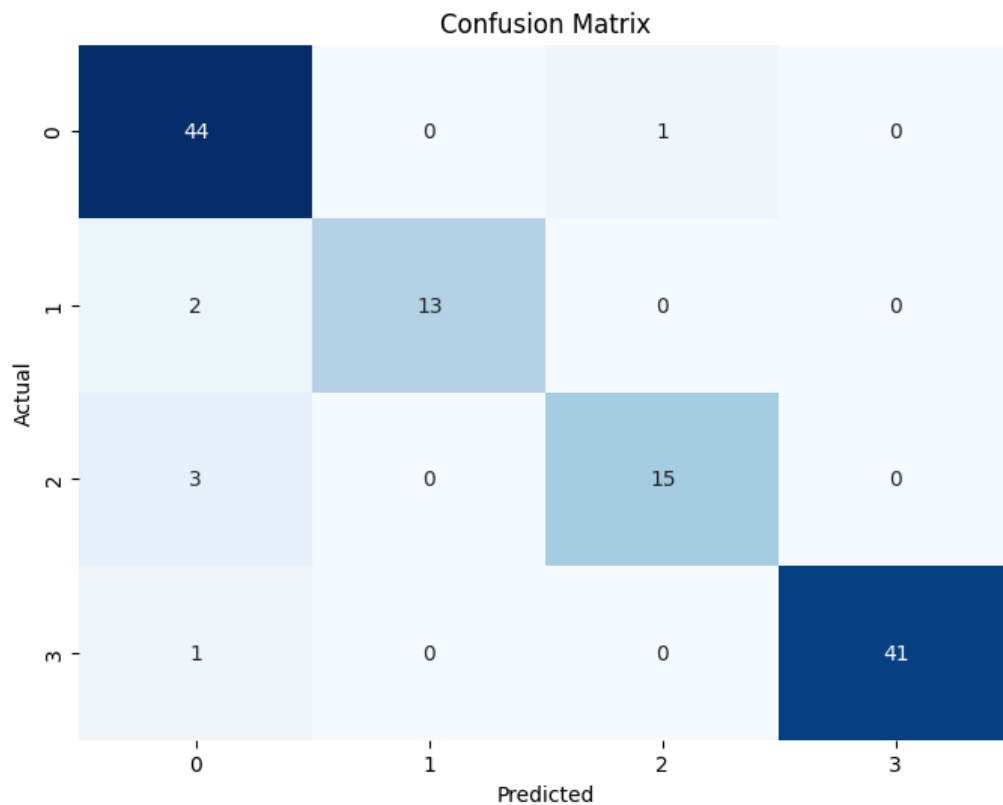


Fig 14. Confusion Matrix of Gradient Boosting

```
Results for Gradient Boosting:  
Bias: 0.09250000000000003  
Variance: 0.0004506944444444444  
Accuracy: 0.9075  
Precision: 0.9175106130461433  
Recall: 0.9075  
F1-score: 0.9072524470710125  
ROC-AUC: 0.9853533369926815
```

Fig 15. Results of Gradient Boosting

## 7. KNN

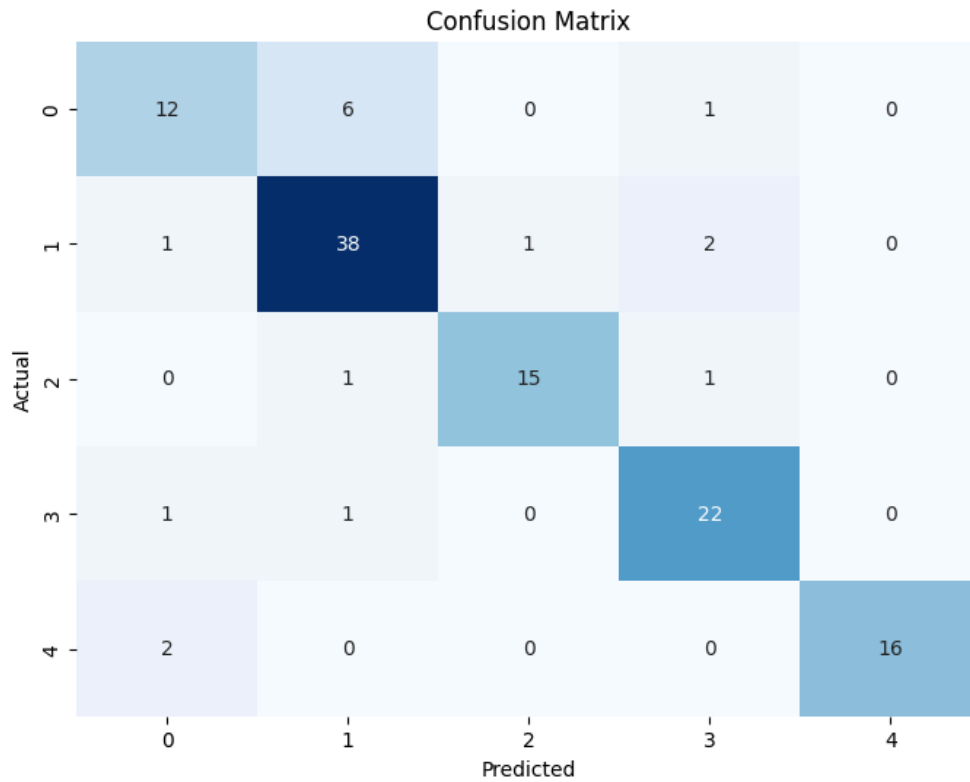


Fig 16. Confusion Matrix of KNN



```
Results for KNN:  
Bias: 0.10250000000000004  
Variance: 0.0007229166666666679  
Accuracy: 0.8975  
Precision: 0.9011266191582885  
Recall: 0.8975  
F1-score: 0.8973114802722313  
ROC-AUC: 0.9776782189606934
```

Fig 17. Results of KNN

### **Bias and Variability Analysis:**

The introduction of noise to the data landscape can have nuanced consequences on both bias and variance, influencing the model's capacity to discern underlying patterns. Understanding how noise interacts with the learning process is pivotal for comprehending the trade-offs involved.

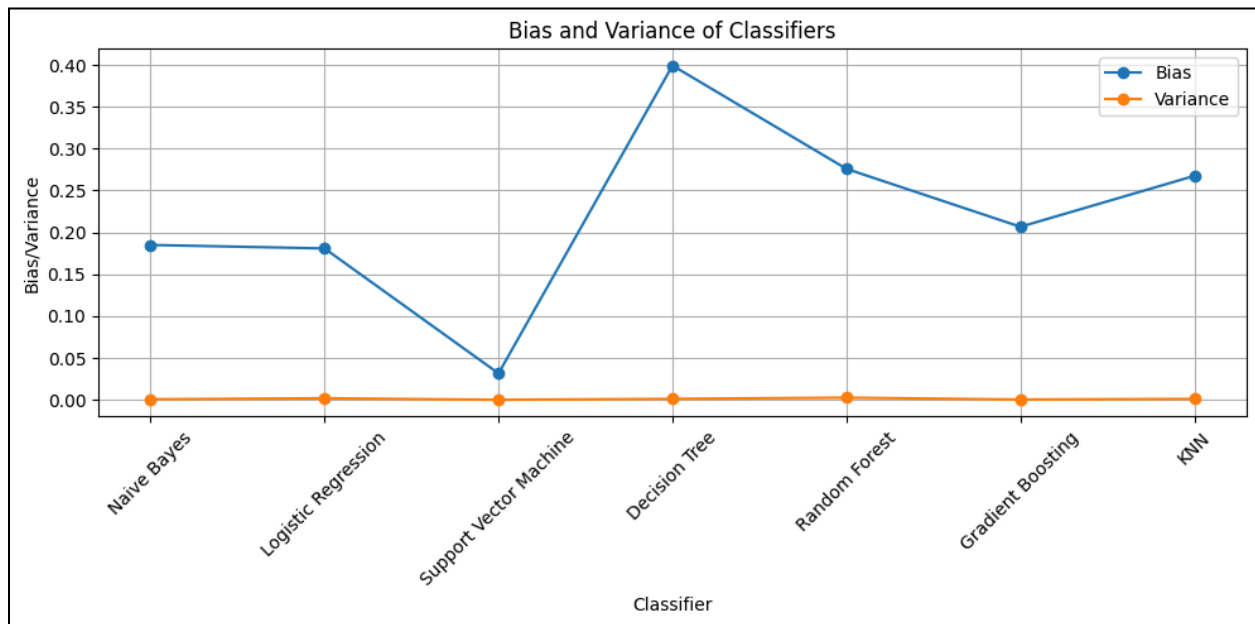


Fig 18. Bias and Variance with Noise

## **Bias Implications**

**Inconsistencies and Irrelevance:** Noise, by its nature, introduces inconsistencies and irrelevant information into the data. This presents a challenge for the model to differentiate genuine patterns from random fluctuations, potentially leading to a distorted representation of the true underlying structures.

**Generalization Challenges:** The presence of noise complicates the model's task of generalizing patterns from the training data to unseen instances. If the noise overwhelms the genuine signal, the model may struggle to capture the essential features, resulting in poor generalization. This phenomenon manifests as an increase in bias, as the model becomes less adept at fitting the training data.

## **Variance Considerations**

**Inherent Variability:** Unlike bias, variance is more attuned to the inherent variability in the data. Noise, especially if random, might not significantly alter this intrinsic variability. Variance primarily concerns the model's sensitivity to minor fluctuations in the training data.

**Limited Impact on Sensitivity:** Random noise, devoid of systematic errors, may not substantially impact the model's sensitivity to variations. Since variance encapsulates the model's responsiveness to diverse patterns in the data, random noise might not induce a systematic shift, leaving the overall variance relatively unaffected.

In summary, while noise introduces challenges for both bias and variance, its impact is more pronounced on bias due to the introduction of inconsistent and irrelevant elements. Variance, closely tied to the inherent variability, exhibits a more resilient stance against random noise, especially when the noise lacks systematicity.

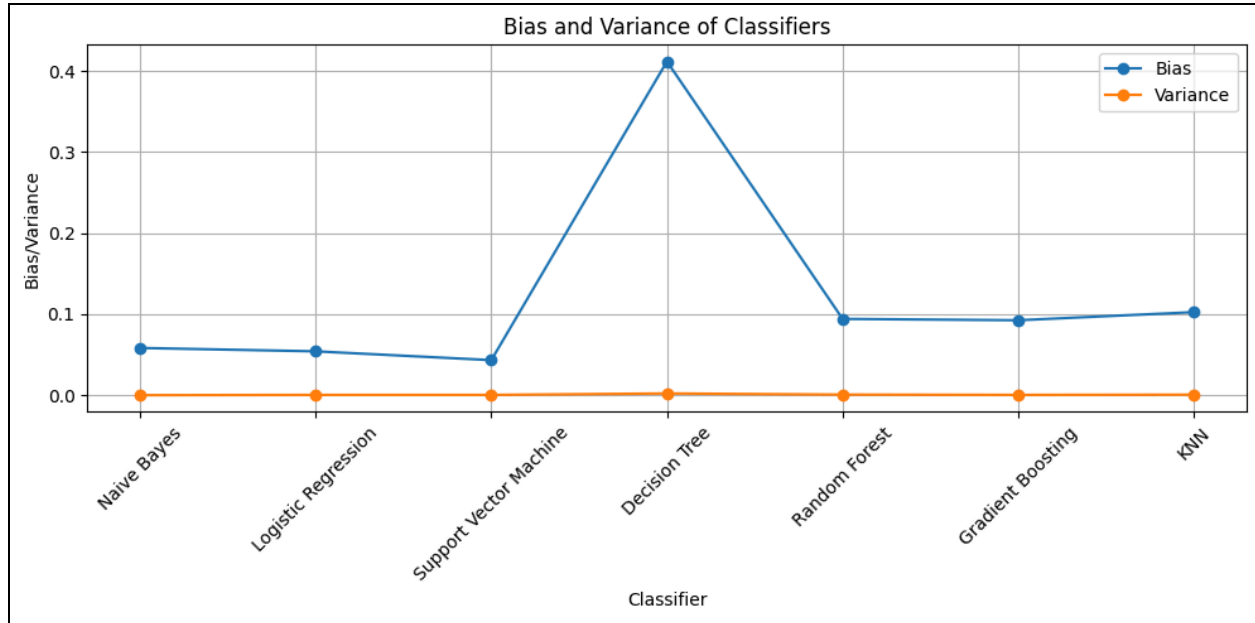


Fig 19. Bias and Variance without Noise

Support Vector Machines (SVMs) are renowned for their ability to effectively capture intricate patterns within data, contributing to their characteristic low bias. Several key factors underline this propensity for low bias in SVMs.

### 1. Flexibility of Decision Boundaries

SVMs exhibit a remarkable flexibility in defining decision boundaries. This adaptability allows them to intricately navigate complex patterns present in the data. The inherent capacity of SVMs to accommodate highly flexible decision boundaries contributes significantly to their ability to fit diverse and intricate relationships between features and labels, ultimately resulting in low bias.

### 2. Margin Maximization

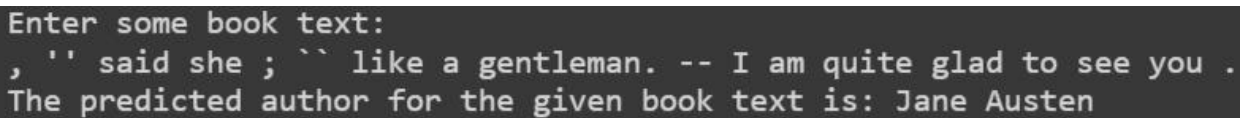
One of the fundamental principles guiding SVMs is the maximization of the margin between different classes. By seeking to maximize this separation, SVMs prioritize the identification of robust decision boundaries. This emphasis on a wide margin enhances the model's generalization capacity, facilitating the creation of decision boundaries that extend well to unseen data. The result is a low-bias model that captures the essential patterns inherent in the dataset.

### 3. Tendency to Overfit

While the tendency to overfit can be perceived as a potential drawback, especially in small datasets or when model complexity is inadequately controlled, it contributes to low bias during training. Overfitting implies that the model precisely tailors itself to the intricacies of the training data. While this might result in low bias for the training set, there's a trade-off as it may hinder generalization to new, unseen data, leading to an increase in variance.

## Conclusion

Our exploration into text classification using Gutenberg digital books has revealed nuanced insights into the complexities of predicting authorship. Through rigorous data preparation, preprocessing, and feature engineering, we laid the foundation for an extensive analysis of various machine learning algorithms.



```
Enter some book text:
, '' said she ; `` like a gentleman. -- I am quite glad to see you .
The predicted author for the given book text is: Jane Austen
```

Fig 20. Author Prediction using champion model

In our journey, SVM emerged as the champion model, showcasing its ability to handle complex patterns with low bias. The careful interplay of feature engineering, particularly TF-IDF, provided a robust framework for our models. The evaluation process, including ten-fold cross-validation, allowed us to scrutinize the performance of each algorithm comprehensively.

Error analysis shed light on the model's vulnerabilities, offering valuable cues for refinement. The addition of noise to the data revealed divergent impacts on bias and variance, emphasizing the delicate balance required for optimal model performance.

In conclusion, SVMs emerge as powerful tools with low bias due to their adaptability, margin maximization strategy, and capacity to navigate complex decision boundaries. However, a nuanced understanding of their behavior, coupled with thoughtful parameter tuning, is crucial to harness their potential effectively.