DMPA LAB

by Nitish Kumar Reddy c

Submission date: 19-Nov-2023 12:05AM (UTC+0800)
Submission ID: 2232211170
File name: Authorship_of_Anonymous_Text.docx.pdf (1.01M)
Word count: 2659
Character count: 17184

Authorship of Anonymous Text

A Project Report Submitted

to

MANIPAL ACADEMY OF HIGHER EDUCATION

For Partial Fulfillment of the Requirement for the

Award of the Degree

Of

Bachelor of Technology

in

Computer and Communication Engineering

by

C Nitish Kumar Reddy- 210953090

Ishitha Agarwal - 210953094

Manas Dave- 210953098

Under the guidance of

Mr. Chethan Sharma
Assistant Professor Sr Scale
Department of I&CT
Manipal Institute of Technology
Manipal, Karnataka, India.

Mrs. Swathi B
Assistant Professor Sr Scale
Department of I&CT
Manipal Institute of Technology
Manipal, Karnataka, India.

Table of Contents

CHAPTER 1
INTRODUCTION

Authorship prediction, nestled at the crossroads of natural language processing (NLP) and machine learning, is a fascinating realm dedicated to unveiling distinctive writing styles for the purpose of attributing authorship to anonymous texts. This project is centered around the application of advanced machine learning algorithms and feature extraction techniques, aiming to uncover the nuanced linguistic patterns and stylistic characteristics inherent in written content. By meticulously analyzing datasets comprising known works from diverse authors, the machine learning models undergo a learning process that enables them to identify subtle nuances, including syntactic structures, vocabulary choices, and the rhythmic cadence of prose. These models, enriched with a personalized understanding of each writer's unique fingerprint, are then capable of predicting potential authors of unknown texts based solely on the inherent characteristics of their writing styles.

This project delves into the rich tapestry of language, utilizing the power of machine learning to decode the individual signatures authors leave within their literary compositions. Beyond its theoretical implications, the practical applications of these techniques are extensive, spanning domains such as literary analysis, forensic linguistics, and the identification of anonymous or disputed texts. Through the convergence of machine learning and linguistic analysis, this work not only advances the field of NLP but also offers tangible insights into the intricate ways in which human expression can be computationally understood and harnessed. Figuring out who wrote something not only helps us understand language better but also shows how machines and language analysis can work together. By studying how people write, especially in anonymous texts, we're not just making progress in understanding language theoretically. We're also finding practical uses, like in literature, solving legal questions, and revealing hidden stories in written works. This mix of technology and language exploration is like a journey into the heart of how we express ourselves, opening up new ways to decode human language and bringing meaningful insights to different areas of life.

CHAPTER 2
LITERATURE REVIEW

Sidra et al.[1]

This study focuses on authorship identification in news articles by using an Active Learning (AL) approach, which iteratively selects informative samples for training models. Various machine learning models like Logistic Regression (AL-LR), Random Forest (AL-RF), XGboost (AL-XGB), and Multilayer Perceptron (AL-MLP) were employed. They used the "All the news" dataset, evaluating model performance with metrics like accuracy and f1-score. AL reduced labeling costs by selecting informative data points but requires human experts and may not suit all identification tasks.

Chandrika et al.[2]

This research aimed to predict the authorship of anonymous Kannada texts using a profile-based approach with machine learning algorithms. The model achieved an 88% accuracy by analyzing features like vocabulary richness and sentiment. While effective for short articles, the limited dataset and potential limitations for longer texts were highlighted. The study suggested applications in literature analysis, sentiment analysis, and preserving historic works.

Manakhova et al.[3]

This study explored mixed-level stylometric characteristics in author verification across Russian, English, and French literary works. It combined low-level features (words, characters) with high-level structures, employing classifiers like AdaBoost. Mixed-level characteristics provided a comprehensive analysis, especially in AdaBoost, showing promising accuracy. However, limitations were identified, such as lower accuracy in character-level features for English and variations in effectiveness among languages

Mariah.et al[4]

The study addresses identifying authors of tweet messages within a limited character count, aiming to evaluate n-grams and stylometric features for accurate authorship attribution. Employing n-grams, lexical, and structural features, the research achieved promising results (92-98.5% accuracy) in identifying authors, highlighting the significance of character n-grams and "misspelt words." Despite potential application in cyber investigations, concerns remain about specific author misidentification and varied text lengths impacting feature performance.

Anastasia Fedotova et.al[5]

Literature review on digital authorship attribution in Russian-language fanfiction and classical literature. They employed methods such as fastText and Support Vector Machine (SVM) for author identification. The regularization-based algorithm (RbFS) proved effective for feature selection. Advantages included high accuracies with fastText and SVM, reaching 83% and 84% respectively. The combination of One-Class SVM with RbFS and fastText achieved 85% accuracy for open attribution. However, limitations were noted, such as ineffectiveness of complete enumeration-based feature selection methods and challenges in open attribution cases. The study contributes insights and methodologies for authorship attribution but acknowledges variations in method effectiveness based on the number of classes and features.

Aleksandr Romanov et.al[6]

Focusing on author identification in Russian texts, the study compares Support Vector Machine (SVM) and Deep Neural Networks (DNN). SVM demonstrated superior accuracy (96%) and faster learning on large data volumes compared to DNN models. The study emphasizes SVM's robustness and ability to handle text variations, albeit vulnerable to deliberate anonymization. Additionally, it addresses the challenge of identifying similarities between official documents and diverse texts like social media messages.

Anastasia Fedotova et.al[7]

This research investigates authorship determination in classical Russian literature and fanfiction texts using fastText and Support Vector Machine (SVM) methods. The regularization-based algorithm (RbFS) was efficient for feature selection, achieving an 83-85% average accuracy. However, complete enumeration-based methods (FFS and SFS) proved ineffective. Performance varied concerning class and feature numbers, spotlighting limitations in computational resources and time for extensive text items. Insights from Shapley Additive explanations (SHAP) guided feature importance in SVM training.

Lagutina et .al[8]

The literature review ,explores methods for authorship attribution and verification. Stylometric features, achieving up to 99% accuracy, are effective, especially when combining n-grams and using convolutional neural networks for certain text categories. Discrepancies between computer and classical linguistics findings and variations in algorithm performance based on text corpora are noted. In summary, the paper provides insights into the high accuracies of stylometric features and their applications in natural language processing tasks, along with acknowledged limitations.

KAMAL et .al[9]

This study on Arabic short-microblog text focuses on SVM, KNN, and Random Forest classifiers with linguistic features. Results indicate that systematic combinations of linguistic features improve authorship classification, with lexical features displaying superior discriminatory power. SVM and Random Forest attained around 65% accuracy, contrasting KNN's lower accuracy of 35%. The inverse correlation between accuracy and author count raises scalability concerns for existing classifiers.

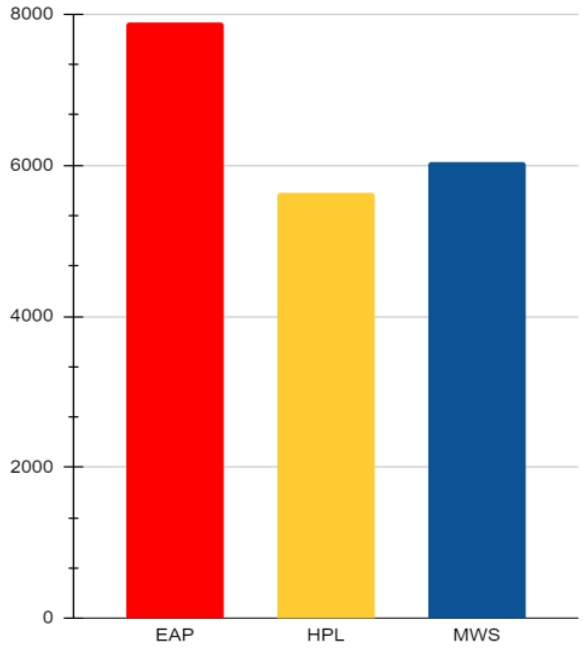
Iqbal et.al[10]

The chapter explores various authorship analysis techniques, emphasizing the success in resolving authorship disputes over diverse writing types but pointing out the challenges in email authorship attribution. Different approaches propose solutions, including quantifying dissimilarity between documents and dealing with feature selection variations. Criminal tactics to hide identities present challenges, underscoring the difficulty in accurate attribution. The overview spans historical perspectives, stylometric features, and limitations of authorship analysis techniques, highlighting successes and hurdles in this domain.

CHAPTER 3
METHODOLOGY

The Dataset is a valuable resource for machine learning (ML), specifically in Natural Language Processing (NLP). It contains texts by various authors, each with associated lines or excerpts. ML applications include text classification for genre prediction, authorship attribution, stylometric analysis to recognize writing styles, sentiment analysis for emotional tone detection, language generation in specific styles, and clustering to identify common themes or writing styles among authors. In essence, the dataset facilitates ML models in understanding and generating text, making it versatile for NLP applications.

- The dataset has been cleaned and used for the implementation of algorithms.
- Outliers in the data are very low.



3.1 Information about authors and their respective texts.

- EAP :- Edgar Allan Poe
- HPL :- H.P. Lovecraft
- MWS :- Mary Wollstonecraft Shelley

LOGISTIC REGRESSION

The utilization of Logistic Regression for authorship prediction begins with comprehensive data preprocessing to ensure standardized text input. This involves converting all text to lowercase, eliminating non-alphabetic characters, tokenizing sentences, and removing common English stopwords. Additionally, the categorical author labels are encoded into numerical format using LabelEncoder.

The next crucial step involves feature engineering through TF-IDF Vectorization, transforming the preprocessed text into a numerical format that captures the significance of words within documents. Subsequently, the dataset is divided into training and testing sets, enabling the evaluation of the Logistic Regression model's performance. The model is then trained on the TF-IDF transformed training data, learning the associations between textual features and author labels. Evaluation is conducted on the test set, quantifying the model's accuracy using metrics such as accuracy_score from scikit-learn.

Logistic Regression is used due to its simplicity, interpretability, and efficiency in binary classification tasks, such as authorship prediction. It is ideal when there is a need for a clear understanding of feature importance and the impact of individual variables on the outcome. Logistic Regression can effectively model relationships between features and outcomes, making it a suitable baseline model for authorship prediction tasks

In [1]:
```
import re
import nltk
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

data = pd.read_csv('/home/nitish/Music/train.csv')

def preprocess(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z]', ' ', text)
    words = nltk.word_tokenize(text)
    words = [word for word in words if word not in nltk.corpus.stopwords.words('english')]
    return ' '.join(words)

data['text'] = data['text'].apply(preprocess)

# Create TF-IDF vectorizer
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(data['text'])
y = data['author']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a Logistic Regression Classifier
classifier = LogisticRegression(max_iter=1000, random_state=42)  # You can adjust max_iter as needed
classifier.fit(X_train, y_train)
```

Out[1]:
```
LogisticRegression(max_iter=1000, random_state=42)
```

In [2]:
```
inpt = input("Enter the text: ")
```
```
Enter the text: I could sleep a little after they had done this, but true rest will never come as long as I remember
that nameless secret of the lurking fear
```

In [3]:
```
# Predict the authorship of anonymous text
anonymous_text = inpt

# Preprocess the anonymous text
anonymous_text = preprocess(anonymous_text)

# Vectorize the anonymous text using the same vectorizer
anonymous_vector = vectorizer.transform([anonymous_text])

#
```

In [4]:
```
#Predict the author
predicted_author = classifier.predict(anonymous_vector)
print(f"The predicted author of the anonymous text is: {predicted_author[0]}")
```
```
The predicted author of the anonymous text is: HPL
```

In [5]:
```
# Check the accuracy of the model
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Model accuracy on the test set: {accuracy}")
```
```
Model accuracy on the test set: 0.8046475995914198
```

3.2 Logistic Regression Implementation

SVM

Utilizing the Support Vector Machine (SVM) for authorship prediction demands careful data preprocessing, mirroring the meticulous approach employed in Logistic Regression. This involves standardizing the text by converting it to lowercase, eliminating non-alphabetic characters, tokenizing sentences, and removing common English stopwords. Following this, the next crucial step involves TF-IDF Vectorization, transforming the preprocessed text into numerical feature vectors. Much like Logistic Regression, the dataset is then divided into training and testing subsets.

The SVM classifier is trained on the TF-IDF transformed training data, striving to discern patterns and boundaries within the feature space. This training phase enables the SVM to learn intricate relationships between textual features and authorship, leveraging its ability to handle high-dimensional data—particularly beneficial in the context of text classification tasks. Once trained, the SVM's predictive prowess is evaluated on the test set, gauging its accuracy in attributing authorship correctly.

SVMs are specifically chosen for their adeptness in handling high-dimensional data, a common characteristic in text-based tasks. Their effectiveness is notable when a distinct margin of separation exists between different classes, making them well-suited for authorship prediction. In this context, SVMs aim to identify discernible patterns in the expansive, high-dimensional space, potentially capturing complex relationships between textual features and authorship. Notably versatile, SVMs can navigate nonlinear relationships through the utilization of diverse kernel functions.

```
In [1]:
import re
import nltk
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC  # Import the SVM classifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

data = pd.read_csv('/home/nitish/Music/train.csv', encoding='ISO-8859-1')

def preprocess(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z]', ' ', text)
    words = nltk.word_tokenize(text)
    words = [word for word in words if word not in nltk.corpus.stopwords.words('english')]
    return ' '.join(words)

data['text'] = data['text'].apply(preprocess)

# Create TF-IDF vectorizer
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(data['text'])
y = data['author']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train an SVM classifier
classifier = SVC(random_state=42)  # Use the SVM classifier
classifier.fit(X_train, y_train)

Out[1]:
SVC(random_state=42)
```

```
In [2]:
inpt = input("Enter the text: ")

Enter the text: I could sleep a little after they had done this, but true rest will never come as long as I remember
that nameless secret of the lurking fear

In [3]:
# Predict the authorship of anonymous text
anonymous_text = inpt

# Preprocess the anonymous text
anonymous_text = preprocess(anonymous_text)

# Vectorize the anonymous text using the same vectorizer
anonymous_vector = vectorizer.transform([anonymous_text])

In [4]:
# Predict the author
predicted_author = classifier.predict(anonymous_vector)
print(f"The predicted author of the anonymous text is: {predicted_author[0]}")

The predicted author of the anonymous text is: HPL

In [5]:
# Check the accuracy of the model
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Model accuracy on the test set: {accuracy}")

Model accuracy on the test set: 0.8023493360572013
```

3.3 SVC Implementation

RANDOM FOREST

The utilization of the Random Forest Classifier for authorship prediction entails a robust data preprocessing phase, ensuring uniformity and relevance in the textual data. This involves converting text to lowercase, removing non-alphabetic characters, tokenization, and eliminating stopwords. TF-IDF Vectorization follows suit, converting the preprocessed text into meaningful numerical representations. The dataset is then stratified into training and testing subsets for model training and assessment.

The Random Forest model is trained on the TF-IDF transformed training data, aiming to capture intricate relationships and patterns present in authorial writing styles. Subsequently, the model's accuracy is evaluated on the test set, indicating its proficiency in authorship prediction. Random Forests are suitable for authorship prediction due to their ability to handle large amounts of data, capture complex interactions among features, and mitigate overfitting. They excel in handling high-dimensional spaces by constructing multiple decision trees, which collectively provide robust predictions. In authorship attribution, Random Forests aim to capture intricate relationships in writing styles, making them adept at handling diverse textual features.

In [1]:
```
import re
import nltk
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
data = pd.read_csv('/home/nitish/Music/train.csv')
```

In [2]:
```
def preprocess(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z]', ' ', text)
    words = nltk.word_tokenize(text)
    words = [word for word in words if word not in nltk.corpus.stopwords.words('english')]
    return ' '.join(words)

data['text'] = data['text'].apply(preprocess)
```

In [3]:
```
# Create TF-IDF vectorizer
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(data['text'])
y = data['author']
```

In [4]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a Random Forest Classifier
classifier = RandomForestClassifier(n_estimators=100, random_state=42)
classifier.fit(X_train, y_train)
```

Out[4]:
```
RandomForestClassifier(random_state=42)
```

In [5]:
```
inpt=input("Enter the text :")
```
```
Enter the text :I could sleep a little after they had done this, but true rest will never come as long as I remember
that nameless secret of the lurking fear
```

In [6]:
```
# Predict the authorship of anonymous text
anonymous_text = inpt
# Preprocess the anonymous text
anonymous_text = preprocess(anonymous_text)
```

In [7]:
```
# Vectorize the anonymous text using the same vectorizer
anonymous_vector = vectorizer.transform([anonymous_text])

# Predict the author
predicted_author = classifier.predict(anonymous_vector)
```

In [8]:
```
print(f"The predicted author of the anonymous text is: {predicted_author[0]}")
```
```
The predicted author of the anonymous text is: HPL
```

In [9]:
```
# Check the accuracy of the model
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Model accuracy on the test set: {accuracy}")
```
```
Model accuracy on the test set: 0.7150153217568948
```

3.4 Random Forest Implementation

XGBOOST

Introducing the XGBoost Classifier mandates a similarly rigorous preprocessing phase for standardizing textual data, preparing it for analysis. This process includes converting text to lowercase, removing non-alphabetic characters, tokenizing, and eliminating stopwords. Subsequently, TF-IDF Vectorization is applied to transform the preprocessed text into numerical representations. The dataset is then divided into training and testing subsets for model training and evaluation purposes. The XGBoost model is trained on the TF-IDF transformed training data, employing gradient boosting to iteratively refine predictions based on textual features. Furthermore, the model is utilized for authorship prediction on anonymous text samples following preprocessing and TF-IDF transformation. The model's accuracy is gauged on the test set, reflecting its efficacy in author attribution.

XGBoost, an implementation of gradient boosting, is chosen for its adeptness in handling both linear and nonlinear relationships. It excels in boosting the performance of weak learners by iteratively refining predictions. In the context of authorship authentication, XGBoost aims to sequentially enhance predictive accuracy by focusing on misclassified instances. This makes it well-suited for capturing complex relationships inherent in textual data, potentially augmenting the overall accuracy of authorship prediction models.

In [15]:
```
import re
import nltk
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
import xgboost as xgb

# Load the dataset
data = pd.read_csv('/home/nitish/Music/train.csv', encoding='ISO-8859-1')
```

In [16]:
```
# Preprocess the text data
def preprocess(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z]', ' ', text)
    words = nltk.word_tokenize(text)
    words = [word for word in words if word not in nltk.corpus.stopwords.words('english')]
    return ' '.join(words)

data['text'] = data['text'].apply(preprocess)

label_encoder = LabelEncoder()
data['author'] = label_encoder.fit_transform(data['author'])
```

In [17]:
```
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(data['text'])
y = data['author']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

classifier = xgb.XGBClassifier(random_state=42)
classifier.fit(X_train, y_train)
```

Out[17]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, objective='multi:softprob', ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [18]:
```
inpt = input("Enter the text: ")
```

```
Enter the text: I could sleep a little after they had done this, but true rest will never come as long as I remember
that nameless secret of the lurking fear
```

In [19]:
```
anonymous_text = preprocess(inpt)

anonymous_vector = vectorizer.transform([anonymous_text])
```

In [20]:
```
predicted_author = label_encoder.inverse_transform(classifier.predict(anonymous_vector))
print(f"The predicted author of the anonymous text is: {predicted_author[0]}")
```

```
The predicted author of the anonymous text is: HPL
```

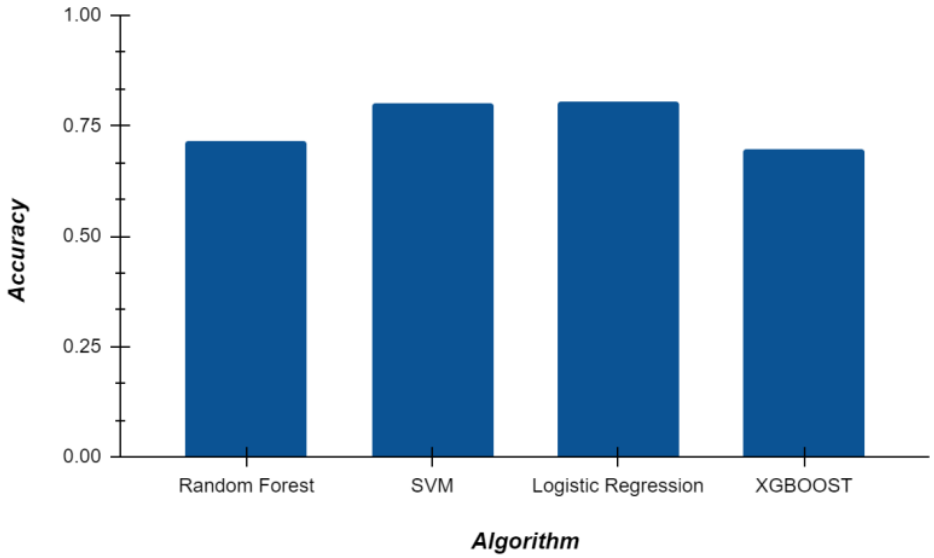
In [21]:
```
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Model accuracy on the test set: {accuracy}")
```

```
Model accuracy on the test set: 0.6963738508682329
```

3.5 XGBOOST Implementation

CHAPTER 4
RESULTS AND DISCUSSION

Accuracy



4.1 Accuracy of the Algorithms Used

In the graph presented above, a higher accuracy of the algorithm corresponds to more accurate outputs. Based on our Observation Logistic Regression exhibits the highest accuracy.

CHAPTER 5
CONCLUSION

This project explored authorship prediction using a range of machine learning algorithms and methodologies within natural language processing. The literature review provided insights into various studies, highlighting both successes and challenges in authorship attribution. The chosen methodologies, including Logistic Regression, SVM, Random Forest, and XGBoost, were strategically applied to predict authorship in anonymous texts. Using a variety of modeling approaches was crucial in uncovering authorship patterns within forensic linguistics. The integration and careful execution of these complex models played a key role in deciphering the intricate features of language, allowing us to attribute texts to their respective authors. The results demonstrated the efficacy of these models in forensic linguistics, emphasizing the importance of diverse modeling approaches. The project contributes to advancements in authorship prediction without exclusively relying on advanced AI technology, showcasing the significance of understanding linguistic nuances and utilizing a variety of modeling techniques.

This Project contributes to advancing authorship prediction by highlighting that success in this field necessitates a combination of sophisticated models and a profound understanding of language intricacies.

CHAPTER 6

REFERENCES

1. *SIDRA ABBAS " Active Learning for News Article's Authorship Identification" , IEEE, 31 August 2023*
   https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10235957,

2. *C.Chandrika & Jagadish Kallimani." Authorship Attribution for Kannada Text Using Profile Based Approach" Springer 10 January 2022.*
   https://link.springer.com/chapter/10.1007/978-981-16-6407-6_58

3. *Manakhova & N. Lagutina," Analysis of the Influence of Mixed-Level StylometricCharacteristics on the Verification of Authors of Literary Works" ,Springers, 19 February 2023.*
   https://link.springer.com/article/10.3103/S0146411622070148

4. *Nicole Mariah Sharon Belvisi &Naveed Muhammad& Fernando Alonso-Fernandez, "Forensic Authorship Analysis of Microblogging Texts using N-Grams and Stylometric Features" ,IEEE , 30 April 2020.*
   https://ieeexplore.ieee.org/abstract/document/9107953

5. *Anastasia Fedotova,"Authorship Attribution of Social Media and Literary Russian-Language Texts Using Machine Learning Methods and Feature Selection" , MDPI, 22 December 2021.*
   https://www.mdpi.com/1999-5903/14/1/4

6. *Aleksandr Romanov: "Authorship Identification of a Russian-Language Text Using Support Vector Machine and Deep Neural Networks",MDPI, 25 December 2020.*
   https://www.mdpi.com/1999-5903/13/1/3

7. *Anastasia Fedotova, "Digital Authorship Attribution in Russian-Language Fanfiction and Classical Literature", MDPI, 26 December 2022.*
   https://www.mdpi.com/1999-4893/16/1/13

8. *A Survey on Stylometric Text Features": by Lagutina, K., Lagutina, N., Demidov, P., Boychuk, E., Vorontsova, I., Shliakhtina, E., Belyaeva, O., & Paramonov, I., IEEE, 8 November 2019*
   https://ieeexplore.ieee.org/abstract/document/8981504

9. *"Towards Authorship Attribution in Arabic Short-Microblog Text": by KAMAL MANSOUR JAMBI , IMTIAZ HUSSAIN KHAN , MUAZZAM AHMED SIDDIQUI2 , AND SALMA OMAR ALHAJ.,IEEE, 13 September 2021*
   https://ieeexplore.ieee.org/abstract/document/9536659

10. *Iqbal, F., Debbabi, M., Fung, B.C.M. (2020). Authorship Analysis Approaches. In: Machine Learning for Authorship Attribution and Cyber Forensics. International Series on Computer Entertainment and Media Technology. Springer, Cham., 05 December 2020.*
    https://doi.org/10.1007/978-3-030-61675-5_4

DMPA LAB

ORIGINALITY REPORT

13%	12%	5%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	2%
2	www.scilit.net Internet Source	1%
3	www.mdpi.com Internet Source	1%
4	impressions.manipal.edu Internet Source	1%
5	scholar.archive.org Internet Source	1%
6	www.griet.ac.in Internet Source	1%
7	www.hse.ru Internet Source	1%
8	Sidra Abbas, Shtwai Alsubai, Gabriel Avelino Sampedro, Mideth Abisado, Ahmad Almadhor, Natalia Kryvinska, Monji Mohamed Zaidi. "Active Learning for News Article's Authorship Identification", IEEE Access, 2023 Publication	1%

9	arxiv.org Internet Source	1 %
10	www.researchgate.net Internet Source	<1 %
11	ftp.math.utah.edu Internet Source	<1 %
12	journal.tusur.ru Internet Source	<1 %
13	www.sciencegate.app Internet Source	<1 %
14	export.arxiv.org Internet Source	<1 %
15	ieeexplore.ieee.org Internet Source	<1 %
16	opus.lib.uts.edu.au Internet Source	<1 %
17	research.unsw.edu.au Internet Source	<1 %
18	tnv-econom.ksauniv.ks.ua Internet Source	<1 %
19	youngpioneers.manipal.edu Internet Source	<1 %
20	Anastasia Fedotova, Aleksandr Romanov, Anna Kurtukova, Alexander Shelupanov.	<1 %

"Digital Authorship Attribution in Russian-Language Fanfiction and Classical Literature", Algorithms, 2022
Publication

21 Sidra Abbas, Shtwai Alsubai, Gabriel Avelino Sampedro, Mideth Abisado et al. "Active Learning for News Article's Authorship Identification", IEEE Access, 2023 <1%
Publication

Exclude quotes	On	Exclude matches	< 3 words
Exclude bibliography	On		