**TREATMENT PREDICTION FROM CLINICAL DATA**

by

**TEAM E**

AYUSHI BHUJADE (ZG28331)

NIMIT TOLIA (YT29650)

SRITEJA MADISHETTY (VT76695)

**FINAL PROJECT REPORT FOR**

For

DATA 606 CAPSTONE FOR DATA SCIENCE

**UNIVERSITY OF MARYLAND BALTIMORE COUNTY**

**2024**

Copyright ©2024

ALL RIGHTS RESERVED

**ABSTRACT**

This project aimed to create a predictive model for analyzing clinical notes to accurately identify treatment names. The impetus behind this work was to enhance clinical decision-making and improve patient care through advanced machine learning methods. The dataset used comprised 17,000 records, with 11 features and 100 treatment categories, sourced from the Hugging Face dataset 'AGBonnet/augmented-clinical-notes'.

The approach included several critical phases: initial preprocessing of the textual data to address inconsistencies and noise, feature engineering with the TF-IDF Vectorizer to transform text into numerical form, and training various machine learning models. Among these, the Clinical BERT model—a transformer-based language model fine-tuned for clinical text—was selected for its exceptional ability to grasp the subtleties of medical language.

The models were evaluated rigorously using metrics such as accuracy, precision, recall, F1-score, and area under the curve (AUC). The Clinical BERT model outperformed others, achieving a notable accuracy of 94%. This performance underscores its potential in aiding healthcare professionals with more precise treatment recommendations.

The success of Clinical BERT in this project illustrates the effectiveness of utilizing large language models in healthcare, particularly for handling complex and context-sensitive textual data. This work demonstrates the significant impact that cutting-edge machine learning techniques can have on clinical data analysis and highlights the potential for future advancements in healthcare.

In conclusion, this project successfully tackled the challenge of predicting treatment names from clinical notes with high accuracy, contributing valuable insights to the field of clinical data analysis and paving the way for future enhancements in patient care.

**ACKNOWLEDGEMENTS**

I acknowledge and thank my team members Ayushi Bhujade, Nimit Tolia, and Sriteja Madishetty for their significant contributions to this project, as well as many colleagues, friends, and guide Unal Sokuglu who gave feedback, helped with, and guided this research project.

TEAM MEMBERS’ CONTRIBUTIONS

Ayushi Bhujade conducted the exploratory data analysis (EDA), which was essential for laying the foundation of our project. She thoroughly examined the dataset, pinpointed significant patterns, and derived insights that were pivotal for the subsequent phases of the work. Her role focused on understanding the dataset’s structure and attributes, which she did with great accuracy and attention to detail.

Nimit Tolia handled the data preprocessing and the extraction of relevant and structured information from the extensive textual data. He ensured the data was clean, well-organized, and prepared for analysis. His proficiency in managing complex data structures was crucial for converting raw data into a usable format. While initially focused on preprocessing, Nimit also contributed significantly to feature engineering and data normalization as the project evolved.

Sriteja Madishetty ran the machine learning models and evaluated the results. He meticulously trained, fine-tuned, and tested various models to ensure the highest accuracy. Sriteja’s primary duty involved selecting the most suitable models, optimizing their performance, and interpreting the outcomes. His efforts in model evaluation and result interpretation were critical to achieving the project's high accuracy.

Below is a table detailing the team members’ primary responsibilities and achievements:

| **Name** | **Duties** | **Achievements** |
| --- | --- | --- |
| **Ayushi Bhujade** | Exploratory Data Analysis | Identified key patterns and provided crucial insights |
| **Nimit Tolia** | Data Preprocessing, Data Extraction | Transformed raw data into a usable format, supported feature engineering |
| **Sriteja Madishetty** | Running ML Models, Evaluating Results | Achieved high model accuracy through meticulous model tuning |

This acknowledgment section highlights the invaluable contributions of each team member and their dedication to the success of this project.

**Table of Contents**

[ABBREVIATIONS 5](#_Toc173771220)

[I. INTRODUCTION 6](#_Toc173771221)

[II. METHODS 8](#_Toc173771222)

[III. RESULTS 11](#_Toc173771223)

[IV. DISCUSSIONS AND CONCLUSIONS 12](#_Toc173771224)

[V. FUTURE WORK 13](#_Toc173771225)

[VI. REFERENCES 14](#_Toc173771226)

[VII. APPENDICES 15](#_Toc173771227)

# ABBREVIATIONS

**List of Abbreviations**

* **NLP**: Natural Language Processing
* **LLM**: Large Language Model
* **TF-IDF**: Term Frequency-Inverse Document Frequency
* **EDA**: Exploratory Data Analysis
* **LDA**: Latent Dirichlet Allocation
* **BERT**: Bidirectional Encoder Representations from Transformers
* **SMOTE**: Synthetic Minority Over-sampling Technique
* **SVM**: Support Vector Machine
* **RF**: Random Forest
* **LR**: Logistic Regression
* **KNN**: K-Nearest Neighbors
* **AUC**: Area Under the Curve

# I. INTRODUCTION

I.1 PROBLEM STATEMENT

The healthcare industry produces a large volume of clinical data daily, including details such as reasons for visits, age, gender, admission causes, diagnoses, and related conditions. While this data is vital, it often presents challenges due to its complexity and unstructured nature. Traditional analytical methods, which rely on manual processes or rule-based systems, often struggle with the sheer volume and intricacy of this data, leading to inefficiencies and potential inaccuracies in clinical decision-making [1].

This project aims to address the challenge of predicting treatment names from clinical data. The dataset contains various facets of patient information and clinical context, but its unstructured format complicates the extraction of meaningful insights. The goal is to develop a predictive model that can accurately identify and recommend treatments by analyzing this detailed clinical dataset.

To tackle this problem, the project employs advanced machine learning techniques and natural language processing (NLP) tools. By transforming raw, unstructured clinical data into structured, actionable insights, the model seeks to enhance the accuracy and efficiency of treatment predictions. Ultimately, the objective is to improve clinical decision-making, streamline treatment processes, and boost patient outcomes, thereby bridging the gap between the vast amount of generated data and its practical application in patient care [7].

I.2 ABOUT THE DATA

The dataset utilized for this project is obtained from the Hugging Face repository, specifically 'AGBonnet/augmented-clinical-notes.' It includes 17,000 records, each featuring 11 distinct columns that capture a wide range of clinical information. This data encompasses critical aspects such as visit motivation, patient age, gender, admission reasons, diagnoses, and related conditions, as well as details on 100 different treatment classes.

To ensure the dataset's quality and suitability for model training, a series of preprocessing steps were implemented. Missing values were systematically addressed to maintain data integrity, preventing gaps that could undermine model performance. The text data was normalized to standardize formats and ensure consistency across the dataset. Additionally, text data was converted into numerical representations using the TF-IDF Vectorizer, a crucial step that allows the model to interpret and process the unstructured textual information effectively. These preprocessing efforts were integral in refining the raw data into a structured and analyzable format, ultimately enhancing the model’s accuracy and efficiency in predicting appropriate treatment names based on the clinical information provided.

1.3 BACKGROUND AND LITERATURE SURVEY

Recent advancements in clinical data analysis leverage machine learning and natural language processing to improve various aspects of clinical documentation and prediction. The MediNote project enhances the efficiency and accuracy of clinical note generation using large language models like MediNote-7B and MediNote-13B [4]. MediTron builds on this by pre-training clinical LLMs on extensive biomedical literature and fine-tuning them with specific datasets to generate detailed clinical notes [1]. The Literature-Augmented Clinical Outcome Prediction study integrates clinical notes with relevant literature through sparse and dense retrieval models to enhance outcome prediction accuracy [2]. NoteChat generates synthetic dialogues using ChatGPT and GPT-4 to provide realistic training data for clinical interactions [3]. Finally, Structured Patient Information Extraction employs GPT-4 to extract structured data from unstructured clinical notes, refining predictive model training [1].

A summary of these studies is provided in the table below -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Project/Study** | **Objective** | **Techniques Used** | **Key Contributions** | **Citation** |
| *MediNote* | Generate clinical notes from dialogues | **Fine-tuning large language models (LLMs)** such as MediNote-7B and MediNote-13B | Automated clinical documentation to enhance efficiency and accuracy | [4] |
| *MediTron* | Clinical LLM development | **Pre-trained on PubMed articles and** clinical guidelines, fine-tuned with the dataset | Improved generation of detailed and structured clinical notes | [1] |
| *Literature-Augmented Clinical Outcome Prediction* | Enhance clinical outcome predictions | Sparse and dense retrieval models for integrating biomedical literature with clinical notes | Improved accuracy in outcome predictions by using both clinical notes and relevant literature | [2] |
| *NoteChat* | Extend PMC-Patients with synthetic dialogues | **Synthetic dialogues generated** using ChatGPT and GPT-4 | Realistic training data mimicking real-world clinical interactions | [3] |
| *Structured Patient Information Extraction* | Extract structured data from clinical notes | Using GPT-4 **to extract structured patient information** | Enhanced structured data extraction from unstructured text for better training of predictive models | [1] |

*Table 1. Literature Survey Summary*

# II. METHODS

II.1 DATA PREPROCESSSING AND FEATURE ENGINEERING

Meticulous data processing and robust feature engineering were fundamental in preparing the clinical data for predictive modeling, ensuring the dataset was clean, consistent, and analytically robust.

**Data Processing**

The data processing phase commenced with the handling of missing values. Records with missing data were deleted to maintain the dataset's integrity and quality. Although this approach reduced the dataset's size, it eliminated the potential biases and inaccuracies that could arise from imputing missing values. This decision ensured that the remaining data was complete, facilitating accurate analysis.

The next step involved standardizing gender categories. The dataset contained various representations for male and female, which could introduce inconsistencies and noise. By mapping all gender entries to standardized categories (male and female), the uniformity of the data was significantly improved. This standardization was essential for ensuring that the analysis and modeling treated gender consistently across all records, thereby improving the reliability of the results.

Addressing the age information provided in textual descriptions was another crucial step. Numerical values were extracted from these descriptions, converting them into a usable numerical format. This transformation facilitated precise age-related insights and allowed for meaningful statistical analysis of age distributions within the dataset. The conversion of textual descriptions into numerical values enabled a more accurate and nuanced understanding of the demographic variables.

Conducting exploratory data analysis (EDA) provided deeper insights into the dataset. Gender and age distributions were analyzed to understand the demographic composition of the data, revealing essential patterns and trends. Additionally, Latent Dirichlet Allocation (LDA) was applied to the visit motivation and diagnosis test columns. LDA helped identify underlying topics and patterns within these textual fields, enhancing the understanding of the data's content. This analysis provided a detailed overview of the key themes and issues within the clinical notes, offering valuable context for the predictive modeling.

The frequency and popularity of different treatments were also analyzed, identifying the most common treatments and providing a clearer picture of the dataset's landscape. This analysis helped prioritize the treatments for further modeling and evaluation.

**Feature Engineering**

Feature engineering was essential for converting raw data into meaningful inputs for the predictive model. A significant step in this process was the standardization of treatment names. The dataset contained multiple synonyms and terms for the same treatments, which could cause redundancy and confusion. By consolidating these terms into a standardized name (e.g., "antibiotics", "antibiotic therapy", and "antibiotic treatment" were all unified under "antibiotics"), we improved the clarity and consistency of the target variable. This standardization allowed the model to learn from and predict treatment outcomes more accurately.

Additionally, to prepare the categorical target variable (treatment name) for machine learning algorithms, label encoding was applied. This conversion to a numerical format was crucial for making the target variable compatible with the predictive models, thereby enabling effective processing and learning from the data.

For the textual data, comprehensive text preprocessing was performed using SpaCy, a powerful natural language processing (NLP) tool. This preprocessing included several steps:

* **Stop Words Removal:** Common, non-informative words were eliminated to reduce noise and improve the relevance of the features.
* **Punctuation Removal:** Punctuation marks were removed to clean the text data, ensuring that only meaningful words were retained.
* **Lemmatization:** Words were reduced to their base forms, ensuring uniformity and reducing the dimensionality of the feature space.
* **Tokenization:** Text was split into individual tokens, facilitating detailed analysis and feature extraction.

Finally, Term Frequency-Inverse Document Frequency (TF-IDF) vectorization was applied to convert the processed textual data into numerical features. TF-IDF effectively captured the importance of words relative to the entire corpus, enhancing the representational quality of the textual features. This vectorization was crucial for enabling the models to leverage the textual data effectively, allowing them to identify key terms and phrases that were predictive of the target variable.

By following these rigorous data processing and feature engineering steps, the dataset was transformed into a clean, consistent, and well-prepared form for predictive modeling. These preparations significantly contributed to the accuracy and reliability of the final predictive model, ensuring robust and insightful outcomes.

II.2 ANALYSIS AND MODEL DEVELOPMENT

Building on rigorous data processing and feature engineering efforts, the analysis and model development phase focused on creating a predictive model to accurately classify treatment names based on clinical data.

**Data**

The dataset consisted of 6,749 records and 10 columns, including 9 predictor variables (X variables) and 1 target variable (Y variable), specifically "Treatment." This dataset presented a challenging multi-class classification problem with 20 distinct treatment classes. The diverse nature of the data required a sophisticated approach to ensure each treatment class was adequately represented and predicted [1].

**Train-Test Split**

To facilitate effective model evaluation, the dataset was divided into training and testing subsets. Using stratified sampling ensured that the distribution of treatment classes in both subsets mirrored that of the original dataset. The training set comprised 80% of the data (5,616 records), while the test set constituted the remaining 20% (1,404 records). Stratification was crucial in maintaining class balance, thus providing a reliable metric for evaluating model performance.

**Cross Validation**

Cross-validation was employed to enhance the robustness of the model evaluation process. Specifically, stratified K-fold cross-validation was utilized. This method divided the data into K subsets, or folds, ensuring each fold had a representative distribution of the target classes. During the training process, each fold served as a validation set while the remaining folds were used for training. This approach mitigated overfitting and provided a comprehensive assessment of the model’s generalization capabilities across different data subsets.

**Oversampling Technique**

Dealing with class imbalance was a critical aspect of developing a successful predictive model. To address this, the Synthetic Minority Over-sampling Technique (SMOTE) was employed. SMOTE works by creating synthetic samples for underrepresented classes, thus balancing the class distribution without simply replicating existing records. This approach improved the model's ability to make accurate predictions across all classes by enhancing the representation of minority classes. The effectiveness of this method was assessed using the F1\_weighted scoring metric, which provides a balanced evaluation of precision and recall.[6]

**Machine Learning Models**

A range of machine learning models were evaluated to identify the most effective method for predicting treatment names:

* **K-Nearest Neighbors (KNN):** Chosen for its straightforwardness and effectiveness in specific situations.
* **Logistic Regression:** Applied due to its interpretability and efficiency in addressing the multi-class classification problem.
* **Support Vector Machine (SVM):** Used for its robustness and effectiveness in high-dimensional spaces.
* **Random Forest:** An ensemble method that aggregates multiple decision trees, selected for its ability to enhance predictive performance and manage complex data structures.
* **Clinical BERT:** This model, fine-tuned for clinical text, was utilized for its advanced natural language processing capabilities, adeptly handling and interpreting intricate textual data.

Each model was trained and evaluated using the prepared training and testing datasets. Their performances were compared to identify the most accurate and reliable model for this task. The evaluation process involved assessing various metrics, including accuracy, precision, recall, F1-score, and AUC, ensuring a comprehensive understanding of each model's strengths and weaknesses.

# III. RESULTS

The evaluation of various models for predicting treatment names was carried out systematically. The models compared include Support Vector Machine (SVM), Random Forest (RF), Logistic Regression (LR), K-Nearest Neighbors (KNN), and a fine-tuned ClinicalBERT model. Each model's performance was assessed based on metrics such as Accuracy, Precision, Recall, F1 score, and AUC score. Below are the detailed results:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision** | **Recall** | **F1 Score** | **Parameters** |
| **ClinicalBERT (Fine-tuned)** | 94% | 0.94 | 0.94 | 0.94 | Epochs: 5; Batch size: 16;  Warmup steps: 500; Weight decay: 0.01;  Evaluation strategy: per epoch |
| **Support Vector Machine** | 91% | 0.91 | 0.91 | 0.91 | C: 10;  class\_weight: 'balanced’;  gamma: 'scale’; kernel: 'linear' |
| **Random Forest** | 87% | 0.88 | 0.87 | 0.87 | class\_weight: 'balanced’; max\_depth: None; features: 'sqrt’; n\_estimators: 200 |
| **Logistic Regression** | 89% | 0.89 | 0.88 | 0.89 | folds = 5; max\_iter = 1000; random\_state = 42 |
| **K-Nearest Neighbors** | 78% | 0.82 | 0.78 | 0.78 | folds = 5;  n\_neighbors = 5 |

*Table 2. Results summary*

The fine-tuned ClinicalBERT model outperformed traditional machine learning models in predicting treatment names based on clinical data. It achieved the highest accuracy and balanced performance across all evaluation metrics, demonstrating the effectiveness of leveraging advanced natural language processing techniques and pretrained models for complex clinical data analysis.

# IV. DISCUSSIONS AND CONCLUSIONS

The results of the project highlighted the significant impact of advanced machine learning techniques, with Clinical BERT proving particularly effective in predicting treatment names from clinical data. Fine-tuned specifically for medical text, Clinical BERT achieved an impressive accuracy of 94%, surpassing other models such as SVM, Random Forest, Logistic Regression, and K-Nearest Neighbors. This exceptional performance demonstrates Clinical BERT’s ability to comprehend and process complex medical language, affirming its value as a powerful tool for enhancing clinical decision-making.

Our study confirmed that leveraging LLMs for tasks involving nuanced and context-rich textual data in healthcare can significantly enhance predictive accuracy. The preprocessing steps, including handling missing values, standardizing gender categories, and converting textual age descriptions to numerical formats, were crucial in ensuring the data's integrity and quality. The use of TF-IDF for text vectorization further refined the data, enabling more accurate model training.

Moreover, the implementation of SMOTE addressed the class imbalance, which was pivotal for the model's ability to make reliable predictions across all treatment classes. The stratified K-fold cross-validation ensured the robustness of our evaluation, mitigating overfitting and providing a comprehensive assessment of model performance.

In conclusion, this project demonstrates the significant potential of employing advanced machine learning models in the analysis of clinical data to enhance patient care. The high accuracy achieved by the Clinical BERT model indicates that such technologies can aid healthcare professionals in making more informed treatment decisions, thereby improving patient outcomes. This work sets the stage for future research and development in this field, with the potential for broader applications across the healthcare industry.

# V. FUTURE WORK

Although our project achieved significant results, several opportunities for future work could further refine and expand these findings. Increasing the dataset's size and diversity could enhance the model's robustness and its ability to generalize across different patient groups and medical conditions. Exploring additional advanced machine learning techniques, such as GPT-4 or hybrid models that integrate various algorithms, might also lead to improved predictive accuracy.

Another valuable direction for future research is the integration of multimodal data, including structured electronic health records (EHRs) and imaging data. This could provide a more comprehensive analysis of patient information and enhance treatment predictions by considering a wider range of clinical factors.

Additionally, developing an interactive tool based on our predictive model could aid healthcare professionals in real-time decision-making. Such a tool could offer treatment recommendations and insights directly at the point of care, improving patient management's efficiency and effectiveness.

Finally, conducting prospective studies to test the model’s performance in real-world clinical environments is essential. Collaborating with healthcare institutions to implement and evaluate the model would ensure its practical applicability and effectiveness in improving patient outcomes. These future steps will help advance the use of machine learning in healthcare, leading to better clinical decision-making and patient care.

# VI. REFERENCES

1. Bonnet, A. (n.d.). AGBonnet/augmented-clinical-notes · datasets at hugging face. AGBonnet/augmented-clinical-notes · Datasets at Hugging Face. <https://huggingface.co/datasets/AGBonnet/augmented-clinical-notes>.
2. Zhao, Z., Jin, Q., Chen, F., Peng, T., & Yu, S. (2023, April 19). PMC-patients: A large-scale dataset of patient summaries and relations for benchmarking retrieval-based clinical decision support systems. arXiv.org. <https://arxiv.org/abs/2202.13876>.
3. Wang, J., Yao, Z., Yang, Z., Zhou, H., Li, R., Wang, X., Xu, Y., & Yu, H. (2024, June 28). NoteChat: A dataset of synthetic doctor-patient conversations conditioned on clinical notes. arXiv.org. <https://arxiv.org/abs/2310.15959>.
4. <https://huggingface.co/datasets/AGBonnet/augmented-clinical-notes/blob/main/report.pdf>.
5. Centers for Disease Control and Prevention. (n.d.). Health Insurance Portability and accountability act of 1996 (HIPAA). Centers for Disease Control and Prevention. [https://www.cdc.gov/phlp/php/resources/health-insurance-portability-and-accountability-act-of-1996 hipaa.html?CDC\_AAref\_Val=https%3A%2F%2Fwww.cdc.gov%2Fphlp%2Fpublications%2Ftopic%2Fhipaa.html](https://www.cdc.gov/phlp/php/resources/health-insurance-portability-and-accountability-act-of-1996%20hipaa.html?CDC_AAref_Val=https%3A%2F%2Fwww.cdc.gov%2Fphlp%2Fpublications%2Ftopic%2Fhipaa.html)
6. Chawla, N. V., DeCoste, D. (2005). "Learning from Imbalanced Data." *ACM Computing Surveys (CSUR)*, 36(3), 321-367. doi:10.1145/1132952.1132962
7. Jurafsky, D., & Martin, J. H. (2021). "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition." *3rd Edition*. Pearson.

# VII. APPENDICES

1. IMPORTING LIBRARIES

from sklearn.model\_selection import train\_test\_split, StratifiedKFold, GridSearchCV, cross\_val\_score

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import (

accuracy\_score, f1\_score, precision\_score, recall\_score,

confusion\_matrix, classification\_report, make\_scorer

)

from sklearn.feature\_extraction.text import TfidfVectorizer

from imblearn.over\_sampling import SMOTE

from imblearn.under\_sampling import RandomUnderSampler

from imblearn.pipeline import make\_pipeline

import matplotlib.pyplot as plt

import seaborn as sns

1. LOADING DATA

# Load the dataset from a CSV file

file\_path = '/path/to/your/data.csv' # Update the path to your dataset

df = pd.read\_csv(file\_path)

# Display the first few rows of the dataset

print("First 3 records of the dataset:")

print(df.head(3))

1. DATA PREPROCESSING

def standardize\_sex(sex):

if pd.isna(sex):

return None

sex = sex.lower().strip()

if sex in ['male', 'm']:

return 'male'

elif sex in ['female', 'f']:

return 'female'

else:

return None

df['sex'] = df['sex'].apply(standardize\_sex)

print("Unique values in the 'sex' column after standardization:")

print(df['sex'].unique())

def extract\_age(age):

if pd.isna(age):

return None

age = re.findall(r'\d+', str(age))

return int(age[0]) if age else None

df['age'] = df['age'].apply(extract\_age)

# Fill missing age values with the median age

median\_age = df['age'].median()

df['age'].fillna(median\_age, inplace=True)

print("Unique values in the 'age' column after extraction and imputation:")

print(df['age'].unique())

1. EXPLORATORY DATA ANALYSIS

# Gender Distribution

plt.figure(figsize=(8, 6))

sns.countplot(x='sex', data=df, palette='pastel')

plt.title('Gender Distribution')

plt.xlabel('Gender')

plt.ylabel('Count')

plt.show()

# Age Distribution

plt.figure(figsize=(8, 6))

sns.histplot(df['age'], bins=20, kde=True, color='skyblue')

plt.title('Age Distribution')

plt.xlabel('Age')

plt.ylabel('Count')

plt.show()

# Popular Treatments

plt.figure(figsize=(12, 6))

top\_treatments = df['treatment\_name'].value\_counts().head(10)

sns.barplot(x=top\_treatments.index, y=top\_treatments.values, palette='viridis')

plt.title('Top 10 Most Common Treatments')

plt.xlabel('Treatment Name')

plt.ylabel('Count')

plt.xticks(rotation=45, ha='right')

plt.show()

# LDA on Visit Motivation

vectorizer = CountVectorizer(max\_df=0.95, min\_df=2, stop\_words='english')

X = vectorizer.fit\_transform(df['visit\_motivation'].dropna())

lda = LatentDirichletAllocation(n\_components=5, random\_state=42)

lda.fit(X)

# Displaying top words for each topic

def display\_topics(model, feature\_names, no\_top\_words):

for topic\_idx, topic in enumerate(model.components\_):

print(f"Topic {topic\_idx+1}:")

print(" ".join([feature\_names[i] for i in topic.argsort()[:-no\_top\_words - 1:-1]]))

no\_top\_words = 10

display\_topics(lda, vectorizer.get\_feature\_names\_out(), no\_top\_words)

1. FEATURE ENGINEERING

# Label encode the target variable 'treatment\_name'

from sklearn.preprocessing import LabelEncoder

label\_encoder = LabelEncoder()

df['treatment\_name\_encoded'] = label\_encoder.fit\_transform(df['treatment\_name'])

print("Mapping of encoded treatment names:")

for original, encoded in zip(label\_encoder.classes\_, label\_encoder.transform(label\_encoder.classes\_)):

print(f"{original}: {encoded}")

# Text preprocessing using SpaCy

nlp = spacy.load('en\_core\_web\_sm')

def preprocess\_text(text):

doc = nlp(text.lower())

tokens = [token.lemma\_ for token in doc if not token.is\_stop and not token.is\_punct]

return " ".join(tokens)

df['processed\_text'] = df['treatment\_name'].apply(preprocess\_text)

1. MODEL TRAINING

# Splitting the data into training and testing sets

X = df['processed\_text']

y = df['encoded\_treatment']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42, stratify=y)

# TF-IDF Vectorization

tfidf = TfidfVectorizer(max\_features=5000)

X\_train\_tfidf = tfidf.fit\_transform(X\_train)

X\_test\_tfidf = tfidf.transform(X\_test)

# Model Building and Evaluation

# Logistic Regression

lr\_model = LogisticRegression(max\_iter=1000, class\_weight='balanced')

lr\_model.fit(X\_train\_tfidf, y\_train)

y\_pred\_lr = lr\_model.predict(X\_test\_tfidf)

print("Logistic Regression")

print(classification\_report(y\_test, y\_pred\_lr))

# Random Forest Classifier

rf\_model = RandomForestClassifier(n\_estimators=100, class\_weight='balanced')

rf\_model.fit(X\_train\_tfidf, y\_train)

y\_pred\_rf = rf\_model.predict(X\_test\_tfidf)

print("Random Forest Classifier")

print(classification\_report(y\_test, y\_pred\_rf))

# Gradient Boosting Classifier

gb\_model = GradientBoostingClassifier(n\_estimators=100)

gb\_model.fit(X\_train\_tfidf, y\_train)

y\_pred\_gb = gb\_model.predict(X\_test\_tfidf)

print("Gradient Boosting Classifier")

print(classification\_report(y\_test, y\_pred\_gb))

# Support Vector Machine (SVM)

svm\_model = SVC(class\_weight='balanced')

svm\_model.fit(X\_train\_tfidf, y\_train)

y\_pred\_svm = svm\_model.predict(X\_test\_tfidf)

print("Support Vector Machine (SVM)")

print(classification\_report(y\_test, y\_pred\_svm))

1. MODEL EVALUATION

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score, confusion\_matrix

import seaborn as sns

# Predict on the test set

y\_pred = svm\_model.predict(X\_test)

# Calculate and print evaluation metrics

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average="weighted")

recall = recall\_score(y\_test, y\_pred, average="weighted")

f1 = f1\_score(y\_test, y\_pred, average="weighted")

roc\_auc = roc\_auc\_score(y\_test, svm\_model.predict\_proba(X\_test), multi\_class="ovr")

print(f"Accuracy: {accuracy:.4f}")

print(f"Precision: {precision:.4f}")

print(f"Recall: {recall:.4f}")

print(f"F1 Score: {f1:.4f}")

print(f"ROC AUC: {roc\_auc:.4f}")

# Confusion matrix

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(10, 7))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=label\_encoder.classes\_, yticklabels=label\_encoder.classes\_)

plt.xlabel('Predicted')

plt.ylabel('True')

plt.title('Confusion Matrix')

plt.show()

1. TRANSFORMERS AND LLM

# Compute class weights

unique\_labels = np.unique(y\_train)

class\_weights = compute\_class\_weight(class\_weight='balanced', classes=unique\_labels, y=y\_train)

class\_weights = torch.tensor(class\_weights, dtype=torch.float)

# Load the ClinicalBERT tokenizer and model from Hugging Face

model\_name = "medicalai/ClinicalBERT"

tokenizer = AutoTokenizer.from\_pretrained(model\_name)

model = AutoModelForSequenceClassification.from\_pretrained(model\_name, num\_labels=len(unique\_labels), ignore\_mismatched\_sizes=True)

# Define a custom dataset class

class ClinicalDataset(Dataset):

def \_\_init\_\_(self, texts, labels, tokenizer, max\_len):

self.texts = texts

self.labels = labels

self.tokenizer = tokenizer

self.max\_len = max\_len

def \_\_len\_\_(self):

return len(self.texts)

def \_\_getitem\_\_(self, idx):

text = self.texts[idx]

label = self.labels[idx]

encoding = self.tokenizer(

text,

truncation=True,

max\_length=self.max\_len,

padding='max\_length',

return\_tensors='pt'

)

return {

'input\_ids': encoding['input\_ids'].flatten(),

'attention\_mask': encoding['attention\_mask'].flatten(),

'labels': torch.tensor(label, dtype=torch.long)

}

# Prepare the datasets

max\_len = 128

train\_dataset = ClinicalDataset(X\_train\_texts.tolist(), y\_train.tolist(), tokenizer, max\_len)

test\_dataset = ClinicalDataset(X\_test\_texts.tolist(), y\_test.tolist(), tokenizer, max\_len)

# Define a compute\_metrics function

def compute\_metrics(p):

preds = p.predictions.argmax(-1)

accuracy = accuracy\_score(p.label\_ids, preds)

precision, recall, f1, \_ = precision\_recall\_fscore\_support(p.label\_ids, preds, average='weighted')

return {

'accuracy': accuracy,

'precision': precision,

'recall': recall,

'f1': f1

}

# Define training arguments

training\_args = TrainingArguments(

output\_dir='./results',

num\_train\_epochs=5,

per\_device\_train\_batch\_size=16,

per\_device\_eval\_batch\_size=16,

warmup\_steps=500,

weight\_decay=0.01,

logging\_dir='./logs',

logging\_steps=10,

eval\_strategy="epoch"

)

# Modify the trainer to include class weights

class WeightedTrainer(Trainer):

def compute\_loss(self, model, inputs, return\_outputs=False):

labels = inputs.get("labels")

outputs = model(\*\*inputs)

logits = outputs.get("logits")

loss\_fct = torch.nn.CrossEntropyLoss(weight=class\_weights.to(model.device))

loss = loss\_fct(logits, labels)

return (loss, outputs) if return\_outputs else loss

# Define the trainer

trainer = WeightedTrainer(

model=model,

args=training\_args,

train\_dataset=train\_dataset,

eval\_dataset=test\_dataset,

tokenizer=tokenizer,

compute\_metrics=compute\_metrics

)

# Train the model

trainer.train()

# Evaluate the model on the test set

test\_results = trainer.evaluate(test\_dataset)

print(f"Test Accuracy: {test\_results['eval\_accuracy']:.4f}")

print(f"Test Precision: {test\_results['eval\_precision']:.4f}")

print(f"Test Recall: {test\_results['eval\_recall']:.4f}")

print(f"Test F1 Score: {test\_results['eval\_f1']:.4f}")