Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Fall, Year:2023), B.Sc. in CSE (Day)

Comparative Analysis between Different Classification Techniques in Machine Learning

---

Course title: Machine Learning Lab

Course Code: CSE 412 **Section:** 203 D1

Students Details

Name	ID
A.K.M. Atiqur Rahman	202002035

Submission Date : 02-01-2024

Course Teacher's Name: Dr. Muhammad Abul Hasan

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Introduction

The project, titled "Comparative Analysis between Different Classification Techniques in Machine Learning" focuses on exploring and comparing various classification methods used in machine learning. This project specifically concentrating on the comparison between Naive Bayes, Logistic Regression and MLP using a news article dataset to classify news as spam or not spam. It designed to highlight the strengths and characteristics of different techniques, this project aims to evaluate and contrast multiple classification approaches. By analyzing how these techniques handle different types of data, our goal is to provide insights that assist practitioners in selecting the most suitable classification technique for their specific applications. Ultimately, this project seeks to simplify the decision-making process for individuals working with machine learning models.

1.2 Motivation

There are several motivations driving the exploration and comparison of diverse classification techniques in machine learning:

- This project aims to simplify the understanding and comparison of various classification methods, aiding practitioners in selecting the most suitable technique for specific data types and applications.
- By understanding the strengths and weaknesses of different classification techniques is vital for improving model accuracy across varied real-world scenarios.
- The project's overarching motivation is to contribute to the advancement of machine learning practices. Through practical insights and comparisons, it aims to facilitate the application of classification techniques across diverse industries and research domains.
- Ultimately, the project endeavors to serve as a practical guide, simplifying

technical complexities into actionable insights for practitioners engaged in machine learning applications.

The primary motivation behind this project is to streamline the selection process of classification techniques, empowering practitioners in achieving improved predictive accuracy while advancing the utilization of machine learning methods.

1.3 Problem Definition

1.3.1 Problem Statement

A problem statement is a brief description of a specific issue or challenge that needs to be addressed. In this "Comparative Analysis between Different Classification Techniques in Machine Learning" project, the problem statement is:

- How can we systematically evaluate and compare various classification techniques in machine learning to assist practitioners in selecting the most suitable method for their specific big datasets?
- What approach can be formulated to simplify the comprehension of classification methodologies, enabling practitioners to make informed decisions while implementing these techniques in real-world scenarios?
- How can we delineate the strengths, limitations, and behaviors of different classification techniques to enable practitioners in comprehending the trade-offs involved in model selection?

By framing these problem statements, the project aims to focus on the specific challenges of comprehensively evaluating and comparing classification techniques, simplifying understanding for practical application, and contributing to the continual advancement of machine learning practices.

1.3.2 Complex Engineering Problem

The comparative analysis between different classification techniques in machine learning involves several complex engineering challenges that need to be addressed for a successful implementation. The complexity of this problem is summarized in the table below:

Attribute	How to Address and Relevance to Project
Algorithmic Expertise:	Attain expertise in various classification methodologies through continuous learning and collaborative knowledge sharing within the team. (Highly relevant to the project's success.)
Performance Optimization:	Implement efficient algorithms and parallel computing strategies to enhance model performance, ensuring scalability and real-time analysis. (Directly applies to the project's objectives.)
Data Processing Depth:	Employ robust data preprocessing techniques and scalable infrastructure for comprehensive data analysis and model training. (Highly pertinent to achieving project goals.)
Stakeholder Engagement:	Foster continuous engagement with stakeholders, aligning model choices with their expectations for effective utilization. (Critical for project success.)
System Integration:	Ensure seamless integration of diverse classification techniques into a cohesive system, enhancing usability and practical application. (Directly aligned with project objectives.)

Table 1.1: Summary of Attributes and How to Address Them

Solving these complex engineering challenges requires a multidisciplinary approach encompassing algorithmic expertise, performance optimization, comprehensive data processing, stakeholder engagement, and seamless system integration. The successful implementation of this project aims to empower practitioners in making informed decisions while selecting classification techniques and advancing the field of machine learning.

1.4 Design Goals/Objectives

In this project, the objectives are:

- To comprehensively evaluate and compare classification techniques in machine learning for optimized predictive accuracy.
- To simplify understanding of these techniques for practical real-world application.
- To outline the strengths and limitations of different methods for informed model selection.

- To contribute to advancing machine learning practices through comprehensive analyses.

1.5 Application

The project on comparative analysis between different classification techniques in machine learning holds significant applications across various domains:

- Guiding researchers and data scientists in selecting optimal classification techniques for varied research projects.
- Assisting industries like finance, healthcare, marketing, and retail to enhance predictive accuracy and decision-making.
- Academic institutions offering machine learning courses can utilize this project's insights to enrich their curriculum and provide students with a practical understanding of classification techniques.
- Empowering startups and innovators to optimize their product's performance through informed classification method selection.

The practical implications of the "Comparative Analysis between Different Classification Techniques in Machine Learning" project extend across diverse sectors, empowering various stakeholders with valuable insights to enhance their decision-making and application of classification techniques in their respective domains.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

In machine learning, there are various ways to classify data. These methods have different strengths and weaknesses. This analysis compares these methods to see which ones work best in different situations. We're looking at things like Logistic Regression, Naive Bayes, and Multilayer Perceptron (MLP). Each method has its own way of working and its pros and cons. We're checking how well they predict things and how fast they are. This comparison helps us understand when to use each method. It's like finding the right tool for the job. By comparing these methods, we learn more about them and how they fit into different problems in real life.

2.2 Project Details

2.2.1 Key Components

In the Machine Learning realm, for comparing different classification methods, a few important parts are crucial:

- **Classification Methods:** These methods are the heart of the project, helping sort and predict data based on patterns. They're the main tools for comparison.
- **Data Setup and Handling:** Getting, cleaning, and organizing data is essential. This ensures the algorithms have good data for learning and accurate predictions.
- **Performance Measurement:** To compare methods, we need to measure their performance. Metrics like accuracy, confusion metrics help understand how well they work.

- **Visual Tools:** Making charts or graphs helps us see and explain differences and similarities between different classification methods.
- **Testing and Changes:** Trying, testing, and making small improvements to each method needs a clear plan. We need to test fairly and compare results.

These are the main parts of this project, helping us explore and understand different Machine Learning classification techniques better.

2.3 Tools and libraries

Here are some tools and libraries used in the project:

- **Software Tools:**
  - Desktop/Laptop
  - Google Colab
  - Microsoft Edge
- **Python Libraries & Techniques :**
  - Classification Techniques: Logistic Regression, Naive Bayes, and MLP
  - Pandas: For data manipulation and analysis
  - Scikit-learn: For machine learning algorithms
  - Matplotlib: For data visualization
  - Ipywidgets: For interactive widgets in Jupyter Notebooks
  - IPython.display: For displaying content in Jupyter Notebooks

Overall, this are main tools and libraries that can be used to build a "comparative Analysis between Different Classification Techniques" project in Machine Learning.

2.4 Algorithms

Here's an algorithm summarizing the steps involved in the project on Comparative Analysis between Different Classification Techniques in Machine Learning:

Algorithm 1: Comparative Analysis between Different Classification Techniques

Input: SMS Spam Collection dataset
Output: Accuracy scores for models and their comparison
Data: Test size slider, learning rate dropdown, activation function dropdown, max iterations dropdown
Load the dataset
Map labels to numeric values
Create widgets for interaction
Initialize accuracy scores and iteration count
Function
`update models(`*test size, learning rate, activation function, max iterations*`):`
    foreach *Model* **do**
        Preprocess data: Split into train-test, convert to bag-of-words
        Train the model with selected parameters
        Predict on test set
        Compute and update accuracy scores
        Print accuracies and confusion matrices
Function `plot`$_c$*omparison*$_c$*urve*`():`
    `Initialize plot`
    `Plot accuracy scores for each model`
    `Display the comparison curve`
`Create a button for updating models and plotting`
while *Button not clicked* **do**
    if *Button clicked* **then**
        `Retrieve selected values from widgets`
        `Update models with the selected parameters`
        `Plot comparison curve`
`Display the widgets and button for updating and plotting;`

2.5 Implementation

2.5.1 Importing Libraries:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import
   CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score,
   confusion_matrix
```

```
import ipywidgets as widgets
from IPython.display import display
import matplotlib.pyplot as plt
```

2.5.2 Loading Data-set and Setting Up Widgets:

```
# Load the SMS Spam Collection dataset
df = pd.read_csv('/content/newsspam.csv', sep='\t',
    names=['label', 'text'])

# Map labels to numeric values
df['label'] = df['label'].map({'ham': 0, 'spam': 1})

# Create a slider to control test size
test_size_slider = widgets.FloatSlider(value=0.2, min
    =0.1, max=0.5, step=0.05, description='Test Size:')

# Define learning rates
learning_rates = [0.01, 0.001, 0.1, 0.9, 0.00000001]

# Create a dropdown menu for learning rate
learning_rate_dropdown = widgets.Dropdown(
    options=learning_rates,
    value=learning_rates[0],
    description='Learning Rate:'
)
# Define activation functions
activation_functions = ['logistic', 'tanh', 'relu']

# Create a dropdown menu for activation function
activation_function_dropdown = widgets.Dropdown(
    options=activation_functions,
    value=activation_functions[0],
    description='Activation Function:'
)
# Define max_iter options
max_iter_values = [100, 500, 1000, 5000, 10000]

# Create a dropdown menu for max_iter
max_iter_dropdown = widgets.Dropdown(
    options=max_iter_values,
    value=max_iter_values[2],
    description='Max Iterations:'
)
```

2.5.3 Storing Accuracy Scores and Iteration Count:

```
# Store accuracy scores for each model and parameter setting
accuracy_scores = {'Logistic Regression': [], 'Naive Bayes': [], 'MLP': []}
iteration_count = 0  # Starting iteration count
```

2.5.4 Function to Update Models:

```
# Function to update models with new parameters
def update_models(test_size, learning_rate, activation_function, max_iter):
    global iteration_count
    # Drop rows with missing values in the label column
    df_cleaned = df.dropna(subset=['label'])

    # Split dataset into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(df_cleaned['text'].astype(str), df_cleaned['label'], test_size=test_size, random_state=42)

    # Convert text data into bag-of-words representation
    count_vector = CountVectorizer()
    train = count_vector.fit_transform(X_train)
    test = count_vector.transform(X_test)

    # Train Logistic Regression classifier
    logistic_regression = LogisticRegression()
    logistic_regression.fit(train, y_train)
    predictions_logistic_regression = logistic_regression.predict(test)
    accuracy_logistic_regression = accuracy_score(y_test, predictions_logistic_regression)
    accuracy_scores['Logistic Regression'].append(accuracy_logistic_regression)

    # Train Naive Bayes classifier
    naive_bayes = MultinomialNB()
    naive_bayes.fit(train, y_train)
    predictions_naive_bayes = naive_bayes.predict(test)
    accuracy_naive_bayes = accuracy_score(y_test, predictions_naive_bayes)
    accuracy_scores['Naive Bayes'].append(accuracy_naive_bayes)

```

```
    # Train MLP classifier with selected parameters
    mlp = MLPClassifier(activation=activation_function,
        max_iter=max_iter, learning_rate_init=
        learning_rate, hidden_layer_sizes=(3, 3))
    mlp.fit(train, y_train)
    predictions_mlp = mlp.predict(test)
    accuracy_mlp = accuracy_score(y_test,
        predictions_mlp)
    accuracy_scores['MLP'].append(accuracy_mlp)

    iteration_count += 1  # Increment iteration count
        after updating models

    # Print accuracies and confusion matrices
    print(f"\nAccuracy for Logistic Regression = {
        accuracy_logistic_regression:.6f}%")
    print(f"Accuracy for Naive Bayes = {
        accuracy_naive_bayes:.6f}%")
    print(f"Accuracy for MLP with {activation_function}
        activation function = {accuracy_mlp:.6f}%")

    print("\nConfusion Matrix for Logistic Regression:")
    print(confusion_matrix(y_test,
        predictions_logistic_regression))
    print("\nConfusion Matrix for Naive Bayes:")
    print(confusion_matrix(y_test,
        predictions_naive_bayes))
    print("\nConfusion Matrix for MLP:")
    print(confusion_matrix(y_test, predictions_mlp))
```

2.5.5 Function to Plot Comparison Curve:

```
# Function to plot comparison curve
def plot_comparison_curve():
    plt.figure(figsize=(10, 6))
    for model, scores in accuracy_scores.items():
        plt.plot(range(iteration_count), scores, label=
            model)
    plt.xlabel('Parameter Setting')
    plt.ylabel('Accuracy')
    plt.title('Model Comparison')
    plt.legend()
    plt.grid(True)
    plt.show()
```

2.5.6 Button Click Function and Display Widgets:

```
# Create a button to trigger the update and plot the comparison curve
update_and_plot_button = widgets.Button(description='Update Models and Plot')

# Function to handle button click event
def on_button_click_update_and_plot(b):
    test_size = test_size_slider.value
    learning_rate = learning_rate_dropdown.value
    activation_function = activation_function_dropdown.value
    max_iter = max_iter_dropdown.value
    update_models(test_size, learning_rate, activation_function, max_iter)
    plot_comparison_curve()

update_and_plot_button.on_click(on_button_click_update_and_plot)

# Display the widgets and button for updating and plotting
display(test_size_slider)
display(learning_rate_dropdown)
display(activation_function_dropdown)
display(max_iter_dropdown)
display(update_and_plot_button)
```

Chapter 3

Performance Evaluation

3.1 Results Analysis/Testing

3.1.1 UI/UX

After running the code it will look like this:

Figure 3.1: Interactive widgets

3.1.2 Final Testing

Here are some output screenshots from our project:

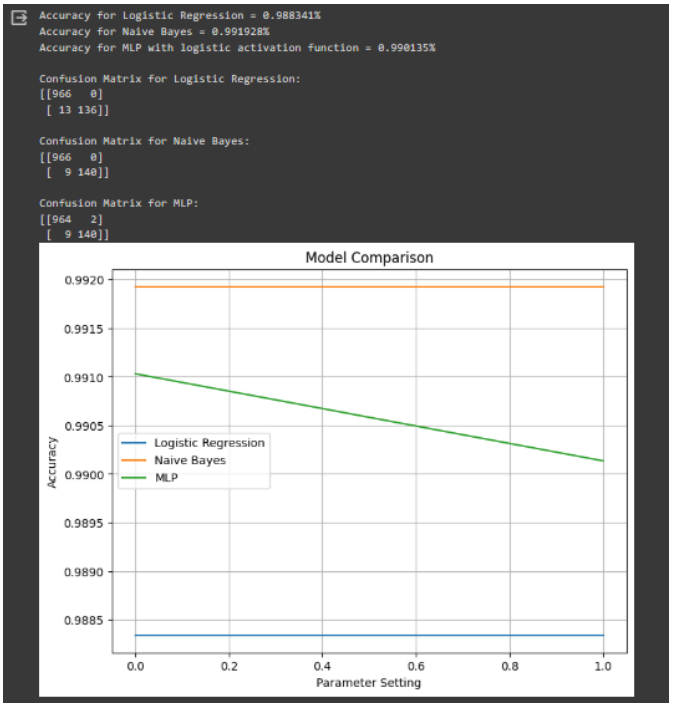

Figure 3.2: The first click updates the models with the initial set of parameter values(Learning Rate=0.01, Activation. Function=Logistic, Max iteration=1000), and its x-axis value would be 1.

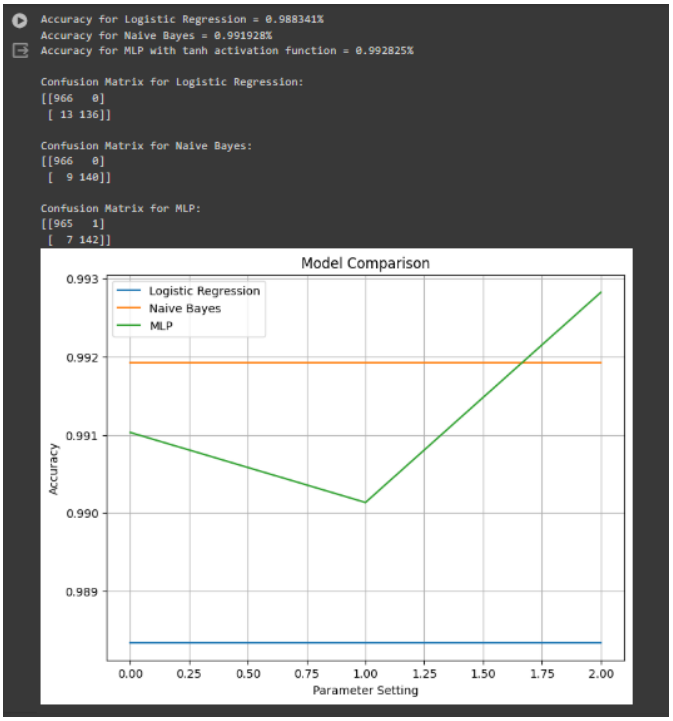

Figure 3.3: The second click updates the models with the new set of parameter values(Learning Rate=0.01, Activation. Function=tanh, Max iteration=1000), and its x-axis value would be 2.

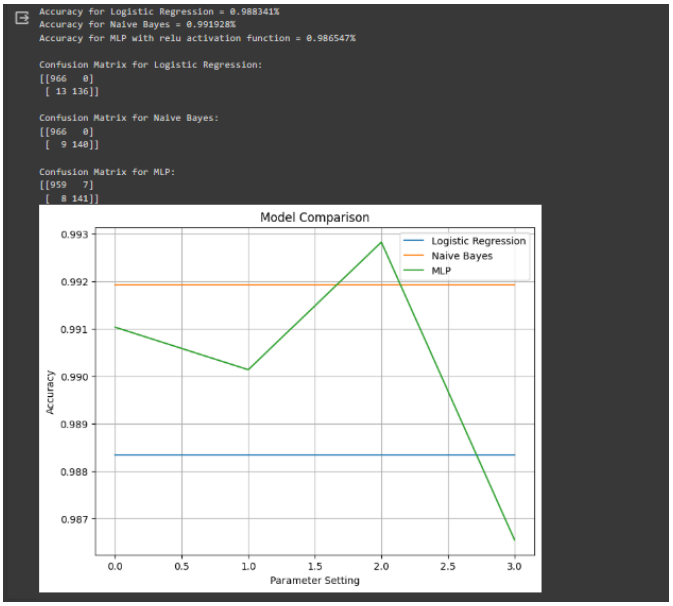

Figure 3.4: The third click updates the models with the new set of parameter values(Learning Rate=0.1, Activation. Function=relu, Max iteration=1000), and its x-axis value would be 3.

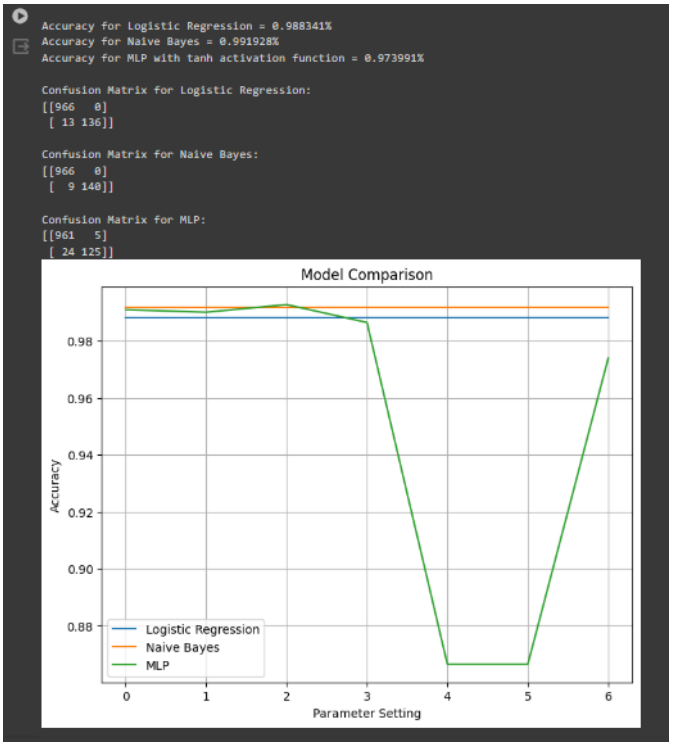

Figure 3.5: The fifth click updates the models with the new set of parameter values(Learning Rate=0.9, Activation. Function=tanh, Max iteration=10000), and its x-axis value would be 5.

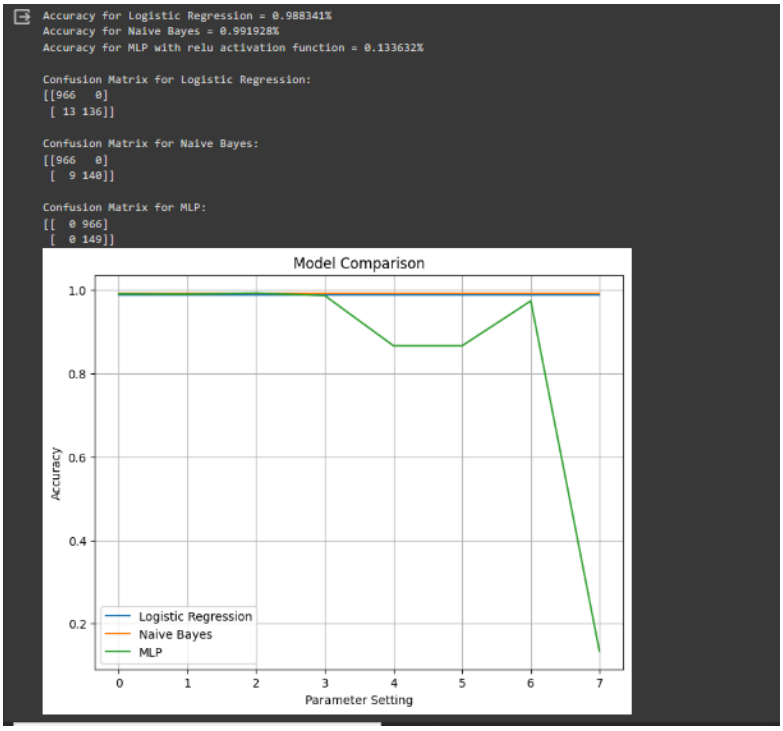

Figure 3.6: The seventh click updates the models with the new set of parameter values(Learning Rate=1e-8, Activation. Function=relu, Max iteration=1000), and its x-axis value would be 7.

3.2 Results Overall Discussion

3.3 Overview of Results

The analysis involved three classification algorithms - Logistic Regression, Naive Bayes, and Multilayer Perceptron (MLP) with different parameter settings like learning rates and activation functions. The dataset used was for classifying news as spam or not spam.

3.3.1 Key Findings and Observations

1. **Algorithmic Performance::**
    - **Naive Bayes:** Surprisingly, Naive Bayes consistently outperformed both Logistic Regression and MLP in classifying news articles across various parameters and scenarios. classifying news articles.
    - **MLP:**The performance of the MLP algorithm exhibited sensitivity to parameter changes, particularly noticeable with the learning rate. A low learning rate, notably 1e-8, resulted in significantly reduced accuracy.

- **Logistic Regression:** While generally trailing behind Naive Bayes and occasionally MLP, Logistic Regression showcased competitive performance in some instances.

2. **Impact of Activation Functions:**

   - **ReLU Activation Function:** Among the tested activation functions, ReLU demonstrated superior performance compared to Tanh and Logistic. This trend was notably consistent, particularly within the MLP algorithm.

Overall, the result showcased Naive Bayes as a robust classification technique for news articles. However, it emphasized the necessity of precise parameter tuning for MLP, particularly in conjunction with ReLU activation. Future endeavors could involve further exploration of parameter optimization strategies for MLP, experimenting with ensemble techniques incorporating Naive Bayes, and investigating alternative activation functions for enhanced classification accuracy.

Chapter 4

Conclusion

4.1 Discussion

The comparison of diverse machine learning classification techniques provides a critical view of their performance differences in categorizing news articles. By studying Logistic Regression, Naive Bayes, and Multilayer Perceptron (MLP), we observed distinct strengths in these algorithms. Particularly, Naive Bayes consistently displayed superior accuracy in distinguishing between genuine news and spam, establishing itself as a dependable classification method. In contrast, MLP's performance was notably sensitive to parameter adjustments, especially the learning rate, highlighting the importance of precise parameter configuration. Additionally, the examination emphasized the prevalence of the ReLU activation function in MLP, significantly impacting its overall classification effectiveness. Overall, this comparative investigation underscores the importance of customized algorithm selection tailored to specific task demands, delivering valuable insights into the diverse performance distinctions among classification approaches.

4.2 Limitations

There are several limitations to consider when developing this project. Some of the main limitations include:

- **Large Dataset Complexity:** The project encountered challenges due to the dataset's scale, leading to extensive computation time.
- **Hyperparameter Sensitivity:** The project faced limitations in effectively handling hyperparameters' sensitivity, particularly in models like MLP. The intricate interplay of these parameters with the dataset's complexity might have restricted achieving optimal performance.
- **Algorithmic Adaptability:** The fixed architecture of certain algorithms, like the fixed hidden layer in MLP, might not sufficiently adapt to varying complexities within the dataset. This lack of adaptability could limit the model's ability to capture intricate patterns.

4.3 Scope of Future Work

There are several potential avenues for future development within this project. Some possible directions include:

- **Optimized Hyperparameter Tuning:** Implement advanced techniques like grid or randomized search to better handle parameter sensitivity in MLP, improving performance.
- **Dynamic Model Architectures:** Develop adaptable model structures adjusting to dataset complexity for enhanced adaptability and performance.
- **Classifier Diversity:**Broaden the range of classifiers assessed to encompass different advanced classification methods. This expansion enables insights into their adaptability to diverse data structures and distributions across different datasets.

These future steps aim to address limitations, enhancing performance, scalability, and adaptability with larger datasets.

References

[1] Analytics Vidhya. (2021, August). Conceptual Understanding of Logistic Regression for Data Science Beginners. Retrieved from https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/?fbclid=IwAR2CUKb2GZiIuysMDOm6hVOe_6l8aucsafZxQO5i0Idp3SdsC3E-iS3PWRQ

[2] Analytics Vidhya. (2017, September). Naive Bayes Explained. Retrieved from https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/?fbclid=IwAR2rbtLku35rhsXHHtQtLk84Co_ztdkTRwrkg5-ZskYpVk27qT_Vdm7Lduw

[3] IBM. Naive Bayes. Retrieved from https://www.ibm.com/topics/naive-bayes?fbclid=IwAR265W7zbW_PqZ2kkH_TNhpWic-4nneQjArMimp5aGJLV7ta1ca9Vg6Xf3I#:~:text=The%20Na%C3%AFve%20Bayes%20classifier%20is,a%20given%20class%20or%20category.

[4] Towards Data Science. (n.d.). Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis. Retrieved from https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentir

[5] Analytics Vidhya. (2020, December). MLP (Multilayer Perceptron) – A Simple Overview. Retrieved from https://www.analyticsvidhya.com/blog/2020/12/mlp-multilayer-perceptron-simple-overview/?fbclid=IwAR3mfxBK252-QLoqRrVp1O04tOTBpy4kf7wCttmmnfeAXa_bFl-Ub_A3LXc