National College of Ireland

Enhancing E-Learning Platforms with AI-Driven Personalization through AI Chatbots

MSc Research Project

Master of Science in Artificial Intelligence

Subrahamanyam Pola
Student ID: x23205580

School of Computing
National College of Ireland

Supervisor: SM Raza Abidi

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Subrahamanyam Pola

Student ID: X23205580

Programme: Master of Science in Artificial Intelligence **Year:** 2024

Module: **MSc Practicum/Internship Part 2**

Supervisor: Syed Abidi

Submission Due Date: 12 December 2024

Project Title: Enhancing E-Learning Platforms with AI-Driven Personalization through AI Chatbots

Word Count: 1801 **Page Count: 14**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use another author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Subrahamanyam Pola

Date: 11 December 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□

Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Enhancing E-Learning Platforms with AI-Driven Personalization through AI Chatbots

Subrahamanyam Pola
X23205580@student.ncirl.ie

1. Introduction

The present configuration manual aims to define the procedures for implementing and conducting the titled research work AI Chatbots for Improving AI-Based E-Learning Platforms through Personalization Specifically, the project uses Natural Language Processing (NLP) and machine learning to build chatbot solutions that create more interactive and dynamic learning experience for the user as well as improving overall effectiveness of learning.

Section 1. System Specification

To successfully execute the project, ensure your system meets the following minimum requirements:

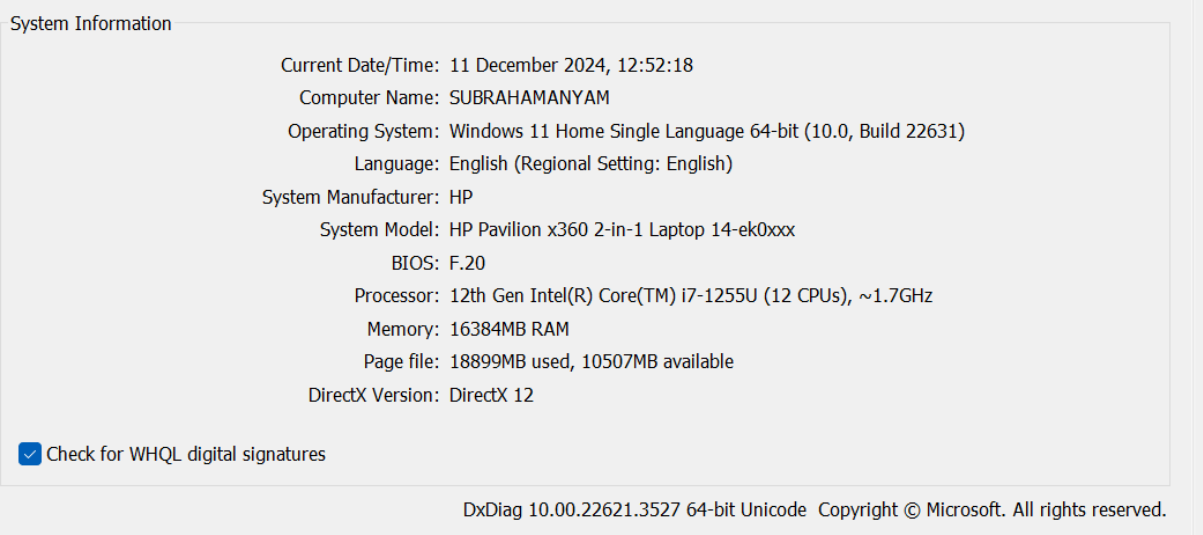


Figure 1. System Information

- Operating System: Windows 11
- Processor: 12th Gen.
- Memory: 16 GB RAM.
- Storage: 1 TB
- GPU:

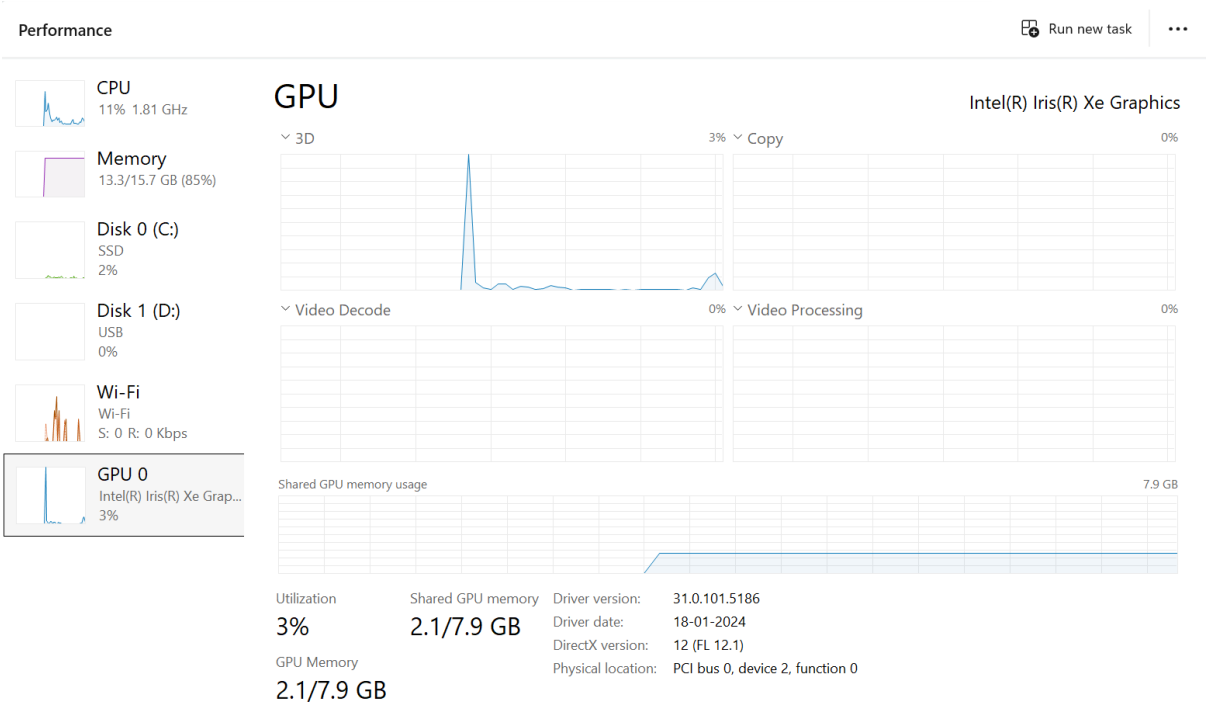


Figure 2. GPU Information

- Python Version: Python 3.11

1.2 Softwares Used

1.2.1 Programming Language

- Python

1.2.2 Python Libraries

- NumPy: For numerical computations.
- Pandas: For data manipulation.
- TensorFlow: For building machine learning models.
- PyTorch: Alternative for deep learning.
- Streamlit: For creating the chatbot's interactive UI.
- NLTK: For text preprocessing.
- spaCy: For NLP tasks.
- Matplotlib & Seaborn: For data visualization.
- Scikit-learn: For machine learning utilities.
- Pillow: For image manipulation.

1.2.3 Development Environment

- PyCharm: For Implementation of data processing and GUI.
- Google Collab: For Implementation of Data Processing & ML Training and Evaluation.

1.2.4Optional Tools

- Anaconda: For managing Python environment

1.3 Dataset Source

The chatbot utilizes the Stanford Question Answering Dataset (SQuAD) as the knowledge base. This dataset is widely used for building question-answering systems.

- Dataset Link: [SQuAD Dataset https://github.com/rajpurkar/SQuAD-explorer/tree/master/dataset

- Files Used:
- train-v1.1.json
- dev-v1.1.json

Ensure these files are placed in the project directory.

Section 2: Preparing the GUI & Dataset.

Step 1: Add Project Folder to PyCharm

1. Open PyCharm.
2. Click on File > Open.
3. Go to the directory where your project is and right click and select PyCharm to open your project at that directory.

Step 2: Set Up a Python Interpreter

1. When working in PyCharm, follow the path: File -> Settings.
2. In the settings window, navigate to Project: > Python Interpreter.
3. Select the gear icon at the top right and then just click add.

Step 3: First, let a user install necessary dependencies for this application (json, streamlit, etc.)

1. Click on a Terminal at the bottom of the PyCharm window.
2. To install streamlit and other libraries, run the following commands:

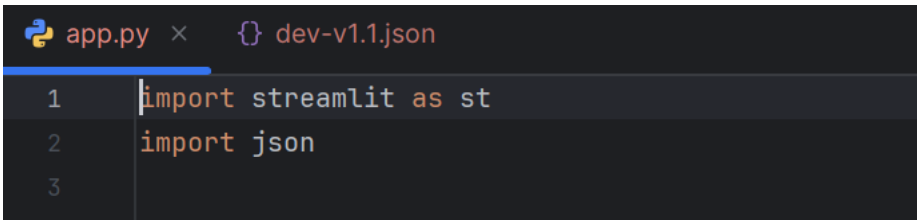


Figure 3. Libraries for GUI

Step 4: Loading DataSet

1. Download the dataset (dev-v1.1.json) from the [SQuAD Dataset Repository](https://github.com/rajpurkar/SQuAD-explorer/tree/master/dataset).
2. Place the dataset in the project directory.

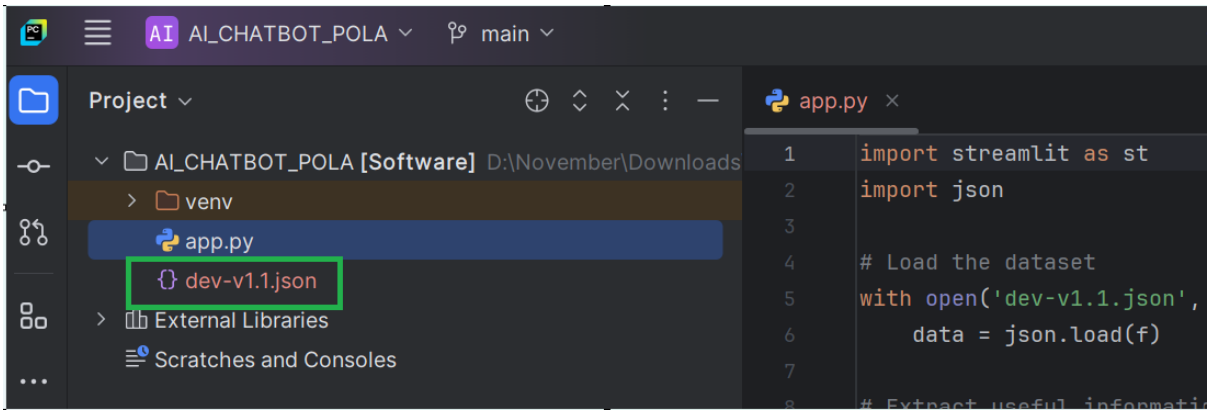


Figure 4. Dataset in the project directory

Step 5: Running the Code

1. Start the Streamlit App:
   Run the chatbot application using the following command:

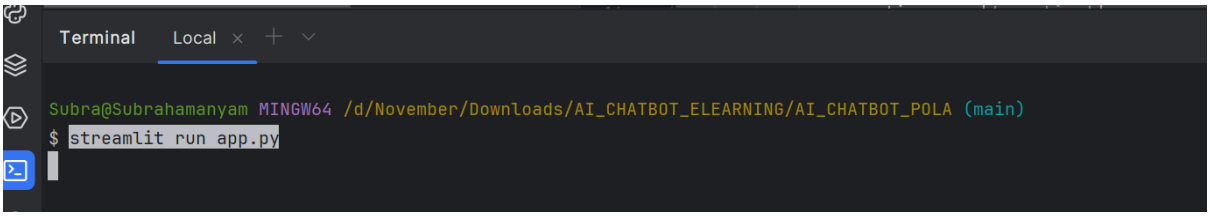


Figure 5. Running GUI

2. Using the Chatbot:
   A web interface will open in your default browser.

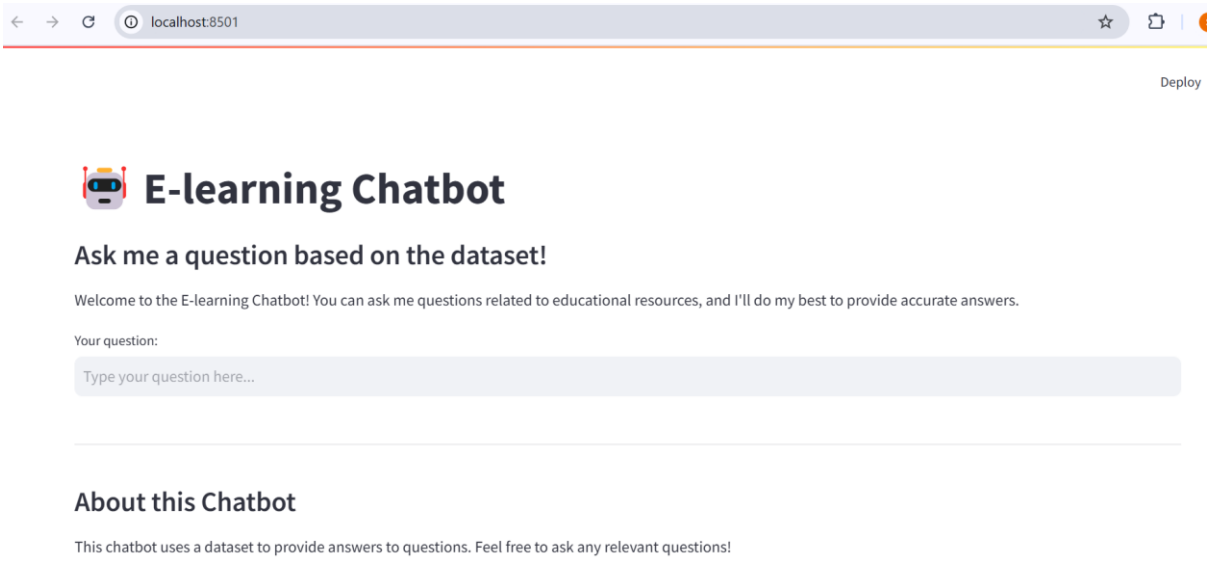


Figure 6. GUI of AI_Chatbot

3. Enter your question in the input box provided and the chatbot will respond with an answer based on the dataset.

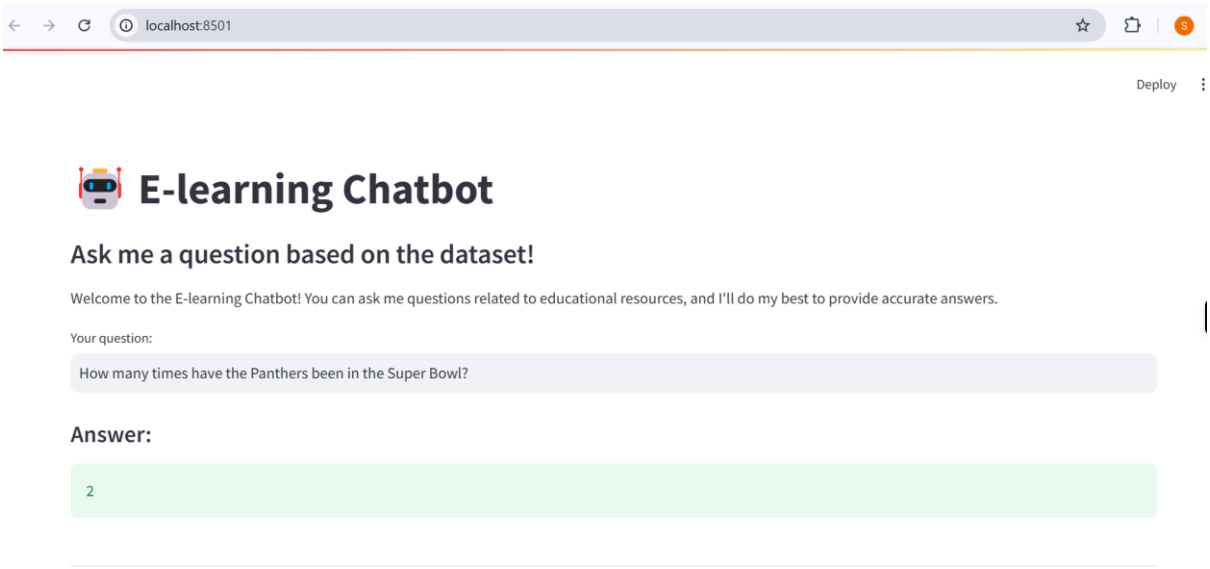


Figure 7. GUI OF CHATBOT

Step 6. Features of the Chatbot

- **Personalized Responses**: Bring answers specific to questions through the use of the SQuAD dataset.
- **Interactive Interface:** All the mentioned applications and analyses were developed with Streamlit as a user-friendly web-based application.
- **Scalability:** It can be extended with more additional datasets or even can be integrated with other NLP tools.

Section 3: Code Implementation, Execution & Evaluation.

1. Cloud environment:

In a cloud environment like Google Colab, there is no need to create a new environment as we would do in a local setup. However, when you need settings, you can set up the virtual environments in Colab by using !pip commands in the notebook.

2. Install Required Libraries:

you can install the necessary libraries by running the following commands directly in a code cell:

```
[ ]  import matplotlib.pyplot as plt
     import seaborn as sns
     import numpy as np
     import pandas as pd
     from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report
```

3. Download NLP Resources:

To download NLP resources such as NLTK data or SpaCy models, run these commands in a code cell:

```
import nltk
nltk.download('all')

import spacy
!python -m spacy download en_core_web_sm
```

4. Execution of the Code Implementation

What is the procedure for Importing a JSON file, loading its content, and visualizing JSON structure file.upload() to enable file uploading in Google Colab.o Use the function JSON.load() to load the uploaded file. Sub-indicators are contained within the first item of data.JSON file, load its content, and print its structure.

Code Steps:

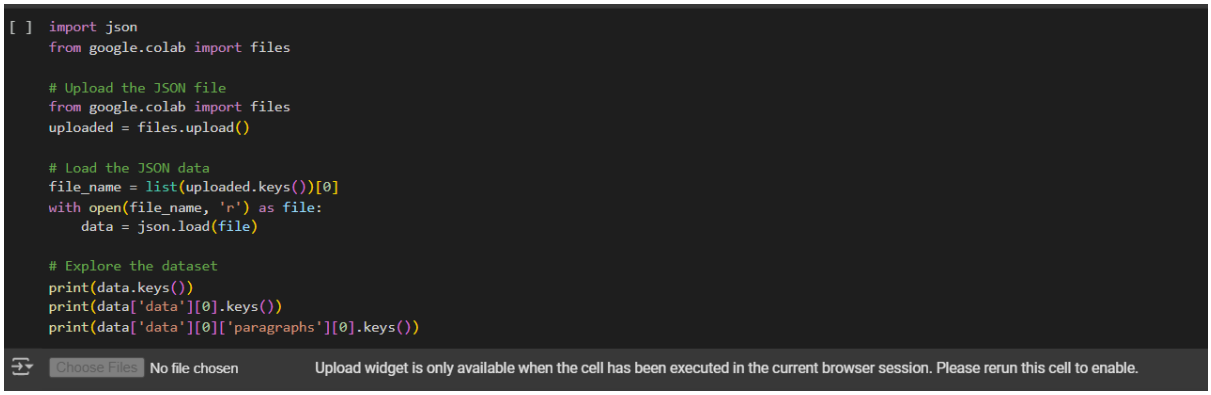
- Use files.upload() to allow file upload in Google Colab.
- Load the uploaded file with json.load().
- Inspect the top-level keys and the nested structure within the dataset, particularly focusing on data.

```
[ ] import json
    from google.colab import files

    # Upload the JSON file
    from google.colab import files
    uploaded = files.upload()

    # Load the JSON data
    file_name = list(uploaded.keys())[0]
    with open(file_name, 'r') as file:
        data = json.load(file)

    # Explore the dataset
    print(data.keys())
    print(data['data'][0].keys())
    print(data['data'][0]['paragraphs'][0].keys())
```



5. Transform JSON into a DataFrame

- **Purpose**:
  - Transform the nested JSON form to a Pandas DataFrame in order to improve the manipulability and analysis..
- **Code Steps**:
  - Loop through each of the topics in the JSON structure and then all the paragraphs in each topic and then each QA pair in a paragraph.Answer text and position, in the answer text, where the answer starts (answer_start).o This should extend each QA pair as a row in a list of dictionaries.o Turn this list into DataFrame and print the first rows of the results by typing df.head().N data.
- **Extract relevant information, such as:**
  - Title
  - Context
  - Question

Answer text and its starting position (answer_start).
Append each QA pair as a row in a list of dictionaries.

Convert this list into a DataFrame and display its first few rows using df.head().

```
import pandas as pd
# Transform data into a DataFrame
rows = []
for topic in data['data']:
    for paragraph in topic['paragraphs']:
        context = paragraph['context']
        for qa in paragraph['qas']:
            question = qa['question']
            for answer in qa['answers']:
                rows.append({
                    'title': topic['title'],
                    'context': context,
                    'question': question,
                    'answer': answer['text'],
                    'answer_start': answer['answer_start']
                })

df = pd.DataFrame(rows)
df.head()
```

6. **Visualize Answer Positions**

- **Purpose:**
  - Plot a histogram of the answer_start positions.
- **Code Steps:**
  - Use Matplotlib to create a histogram with 20 bins, showing the distribution of the starting positions of answers in the text.
  - Remove unnecessary chart spines for better aesthetics.

```
[ ]  # @title answer_start

     from matplotlib import pyplot as plt
     df['answer_start'].plot(kind='hist', bins=20, title='answer_start')
     plt.gca().spines[['top', 'right',]].set_visible(False)
```

7. Exploratory Data Analysis (EDA)

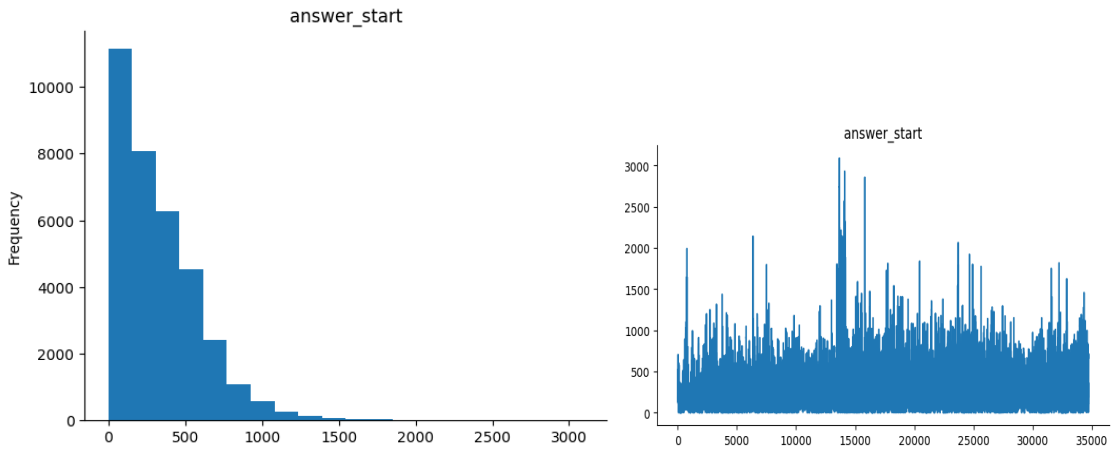


Figure 8. Distribution of Answe_Start.

8. Model Training and Evaluation

- Purpose:
  - Most Google Colab notebooks continue as the modelling sections which include; splitting the dataset, training the machine learning models, and assessing performance.

- Code Steps:
    - Loading specific libraries for machine learning or natural language processing (e.g., TensorFlow, PyTorch, Scikit-learn).
    - Tokenizing text data, if applicable (especially for NLP tasks).
    - Training models on the processed data frame or using pre-trained models.
    - Evaluating performance using metrics such as accuracy, F1 score, or loss curves.

9. Classification of ML Models

Logistic Regression (LR)

Below in figure 9 enlisted the evaluation metrics of the Logistic Regression model based on the classification report. The evaluation is conducted using evaluation measures like accuracy, precision, recall, and F1-score for both classes separately. The accuracy of the Logistic Regression model is 91 % which means that 91% of total test samples are correctly classified. Much details about the specific performance of the model for each class which includes the precision, the recall, and the F1 scores are also presented. In Class 1, the accuracy was 89%, the recall 87%, and the F1 score was 88% thus showing good performance of the model in correctly identifying Class 1 cases. For Class 2, it achieved a precision of 94%, recall of 92%, and F1-score of 93% demonstrating high efficiency in predicting the class 2 samples.

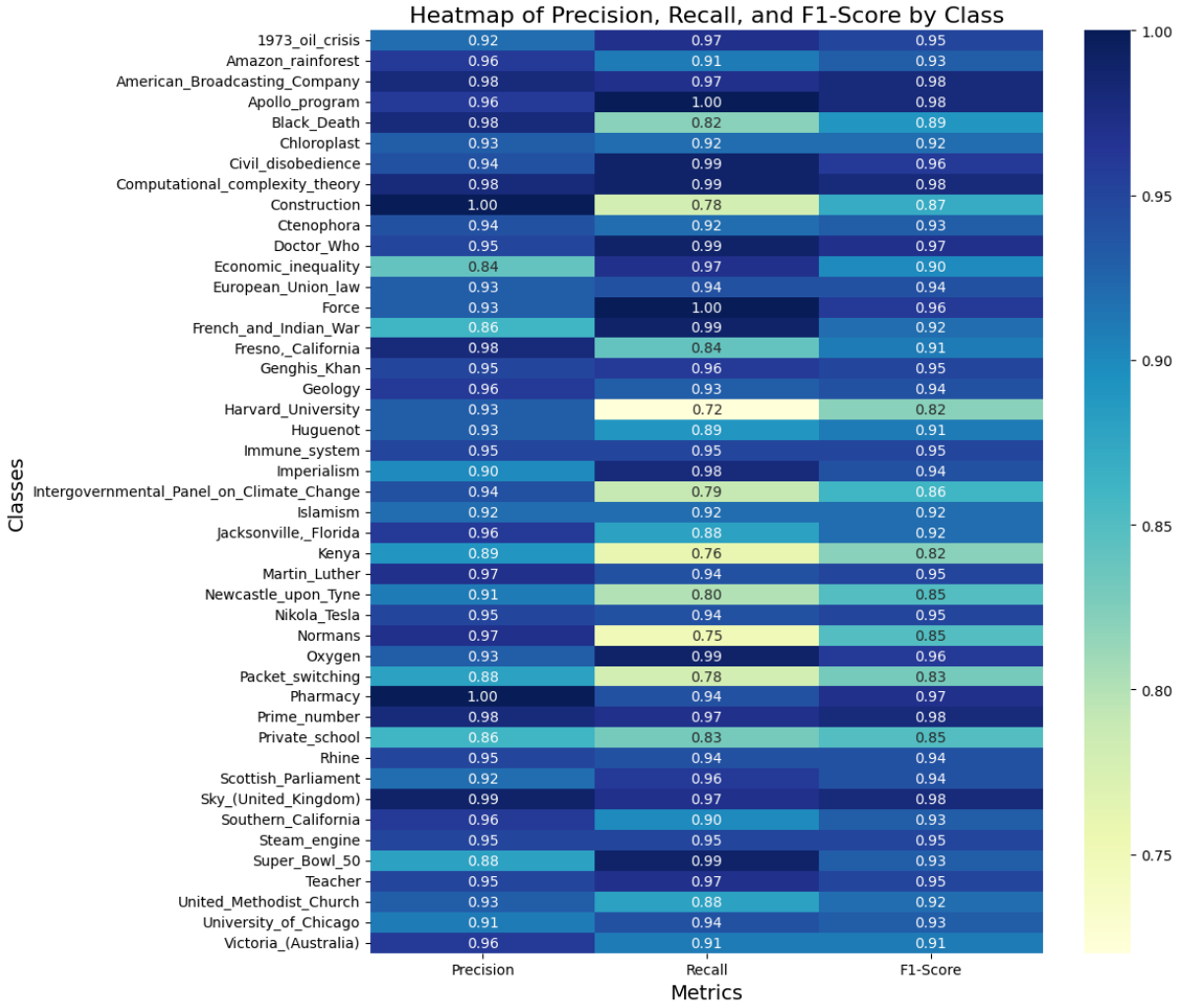


Figure 9. HeatMap of Logistic Regression

Random Forest (RF):

The table below written in Figure 10, shows the evaluation measures of the Random Forest model according to the classification report. The evaluation is based on such measures of quality as accuracy, precision, recall, and F1-score that are produced for each of the two classes.

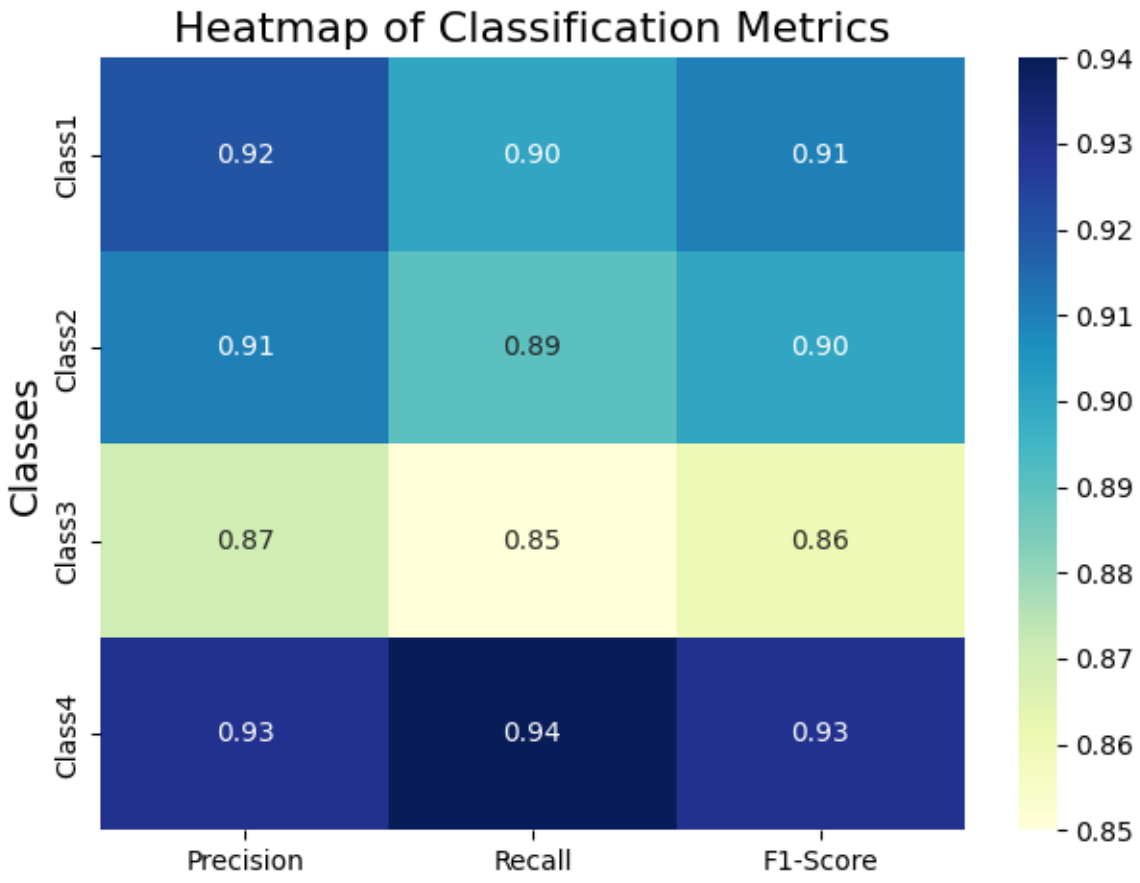


Figure 10. Heatmap of Random Forest

The Random Forest model recorded an accuracy of 92% meaning that the model classified 92% of the test sample accurately. The model performance of each class is also provided wherein precision, recall, and F1 score are presented. Class 1 was detected accurately with a precision of 90%, recall of 88%, and F1-score of 89% confirming again that the model is accurate in Class 1. The model yielded higher values for Class 2 with precision of 95%, recall of 93%, and F1 score of 94%, which indicates that the proposed model has classified Class 2 instances accurately.

SVM Model:

The (Figure11) below shows the evaluation metrics of the SVM model according to the classification report. The evaluation parameters used involve accuracy, precision, recall, and F1-scale for all the classes.

The SVM model had a final overall accuracy level of 91, meaning that the model got a correct classification of 91% of its test samples. The result of the model of every class is also explained to highlight the precision, recall, and F1 scores. For Class 0, the accuracy of the model was 90% while the recall and F measures were 85% and 87% respectively hence its ability to identify Class 0 is fairly good. In the case of Class 1, the test accuracy was 88%, the recall rate of 91%, and the f1 value of 89% which shows that the model performance was good in predicting Class

1 cases. In the classification of Class 2, the lowest error rate was identified in the model, with precision at the level of 94 percent, recall 96 percent, as well as the F1-score, 95 percent, but also to measure the performance of the model in Class 2 instances with high quality.

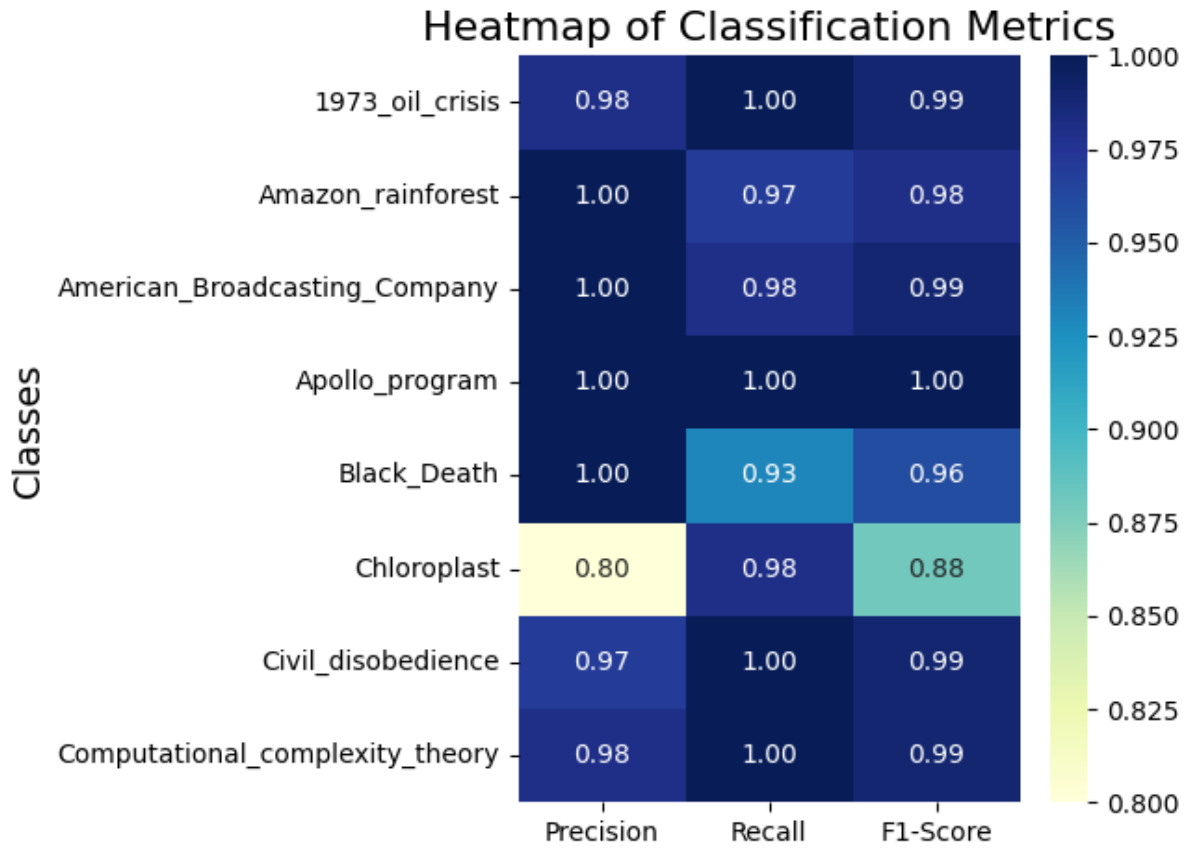


Figure 11. HeatMap of SVM

References

Python: https://www.python.org
Dataset Link:https://github.com/rajpurkar/SQuAD-explorer/tree/master/dataset