SHRI RAMSWAROOP MEMORIAL UNIVERSITY

MINDLYTICS

Shaping Tomorrow with Predictive Intelligence

A PROJECT REPORT

BACHELOR OF ENGINEERING

(Computer Science and Engineering)

SESSION: 2024-2025

Submitted By:

Abhishek Vishwakarma

202310101150305

CS (DS+AI) CS CX-41

Submitted To:

Ujjwal Tomar

(Faculty of IBM)

ABSTRACT

The exponential growth of educational data has enabled institutions to leverage predictive analytics for enhancing student performance and outcomes. This project, titled **"Mindlytics – Predictive Analytics for Student Academic Success"**, aims to build a robust machine learning solution to predict student performance using real-world data.

A dataset from the UCI Machine Learning Repository, based on Bangladeshi student records, was used for model training. Multiple predictive models were created and evaluated using **IBM SPSS Modeler**, including Decision Trees and Neural Networks. The **Linear Regression model**, which achieved an **accuracy of over 95%**, was selected as the most reliable.

The finalized model was deployed using **IBM Watson Studio** and exposed via a public REST API. A **user-friendly web dashboard** was then developed using **Flask, HTML, and CSS**, allowing real-time predictions through API integration.

The project not only showcases the power of machine learning in education but also demonstrates end-to-end deployment — from data acquisition and model building to live prediction via web interface — making it a comprehensive application of modern AI techniques.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who have contributed to the successful completion of this project titled **"Mindlytics – Shaping tomorrow with predictive intelligence."**

First and foremost, I am thankful to my faculty mentor and academic guide for their continuous support, encouragement, and valuable insights throughout the development of this project. Their guidance helped me refine my approach to machine learning and data-driven decision-making.

I would also like to acknowledge the UCI Machine Learning Repository for providing the dataset based on Bangladeshi student performance, which served as the foundation of this study.

Special thanks to the developers and maintainers of **IBM SPSS Modeler** and **IBM Watson Studio**, whose platforms enabled the seamless creation, training, and deployment of predictive models. I built and evaluated multiple models including Linear Regression, Decision Tree, and Neural Networks. Among them, the **Linear Regression model demonstrated an accuracy of more than 95%**, making it the most reliable for deployment.

Finally, I appreciate the power and flexibility of **Flask, HTML, and CSS**, which allowed me to develop an interactive and intuitive web-based dashboard that consumes the Watson-deployed model API to deliver real-time predictions.

This project has not only enhanced my technical skills in AI and machine learning but has also deepened my understanding of deploying predictive models in real-world applications.

Date: 21/24/2025 Abhishek Vishwakarma

Place: Lucknow 202310101150305

TABLE OF CONTENT

LIST OF ABBREVIATION

Abbreviation	Full Form
AI	Artificial Intelligence
ML	Machine Learning
SPSS	Statistical Package for the Social Sciences
API	Application Programming Interface
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
UI	User Interface
R^2	Coefficient of Determination
JSON	JavaScript Object Notation
REST	Representational State Transfer
UCI	University of California, Irvine (Dataset Source)
IBM	International Business Machines Corporation

REQUIREMENTS

Software Requirements

S.No.	Software/Tool	Purpose
1	**SPSS Modeler**	Data preprocessing, model building, evaluation
2	**IBM Watson Studio**	Model deployment, API generation
3	**Python (3.x)**	Flask backend development
4	**Flask**	Web framework to connect backend to frontend
5	**Postman** *(optional)*	API endpoint testing
6	**Code Editor (VS Code, Sublime)**	HTML, CSS, Flask editing
7	**Browser (Chrome/Edge)**	Dashboard testing and UI preview

Python Libraries

```
pip install flask
pip install requests
pip install json
```

(Note: requests is required for calling Watson API in Flask app)

Hardware Requirements

S.No.	Item	Specification (Minimum)
1	Processor	Intel i3 5th Gen or higher
2	RAM	4 GB (8 GB recommended)
3	Storage	2 GB free space
4	Internet	Stable connection (for IBM Watson API calls)

Account Requirements

S.No.	Platform	Requirement
1	IBM Cloud	IBM Watson Studio access
2	UCI Repository (Free)	Dataset download access
3	GitHub *(optional)*	Code backup and versioning

CHAPTER 1: Dataset Description

1.1 Source

The dataset used in this project was sourced with significant effort from the **UCI Machine Learning Repository**, specifically focusing on **Bangladeshi students**. Finding the right dataset was a challenging task as it had to meet the specific requirements for building a predictive model related to academic performance. After reviewing several datasets, this one was selected due to its comprehensive attributes like age, study time, gender, parental education, and income, which are critical for making predictions.

1.2 Attributes

The dataset consists of several important attributes, which include:

- **Student Age**: The age of the student, which can influence academic performance.
- **Gender**: The gender of the student, as it can play a role in educational outcomes.
- In a semester, how often you felt nervous, anxious or on edge due to academic pressure?
- In a semester, how often have you been unable to stop worrying about your academic affairs?
- In a semester, how often have you had trouble relaxing due to academic pressure?
- In a semester, how often have you been so restless due to academic pressure that it is hard to sit still?
- In a semester, how often have you felt afraid, as if something awful might happen?
- **Anxiety Value**: This is the target variable and represents the final grade or academic performance of the student.

1.3 Target Variable

The **Anxiety Value** is the target variable. This is the output we are trying to predict based on the input features (age, gender, study time, etc.). Accurate prediction of academic performance is crucial for understanding the factors influencing student success and making data-driven recommendations.

CHAPTER 2: Tools and Technologies Used

2.1 SPSS Modeler

We used **SPSS Modeler** for data preprocessing and building machine learning models. The task of preprocessing involved several complex steps, including:

- **Missing Value Handling**: This step was crucial because the dataset contained some missing values that could affect model accuracy.
- **Data Transformation**: We had to transform categorical features into numeric values and scale the continuous variables so that they could be processed by machine learning algorithms.

Model Building: Using SPSS, we built multiple models to predict student performance. This included decision trees and linear regression models. The challenge was to tune these models properly to ensure they performed well on unseen data.

2.2 IBM Watson Studio

IBM Watson Studio was used to deploy the models and make them accessible through a public endpoint. We first saved the trained model in the **.xml** format, which was compatible with Watson. The process involved:

1. **Creating a Watson Project**: Setting up the project on Watson Studio took several hours, as we had to make sure the project was configured to handle our specific model and data.
2. **Importing the Model**: Once the model was saved in the **.xml** format, it was imported into Watson.
3. **API Generation**: Using Watson's interface, we generated a REST API, which allowed us to send data to the model for prediction.

2.3 Flask

Flask was chosen to create the **backend of the web application**. Setting up Flask involved:

- **Flask Installation**: The installation process was tedious as Flask had dependencies that required proper setup. We also had to install **Flask-RESTful** to handle API requests and responses efficiently.
- **Creating Routes**: We created different routes to handle user input, pass it to the Watson API, and return the prediction to the user.

2.4 HTML & CSS

For the **frontend**, we used HTML and CSS to design a user-friendly interface where students could input their data (age, study time, etc.). The task was time-consuming because we had to ensure that the layout was simple yet functional, allowing users to input data easily and view predictions clearly. Several iterations were made to improve the UI/UX experience.

CHAPTER 3: Model Development

3.1 Data Preprocessing

The data preprocessing step was essential to prepare the data for model building. It involved:

- **Handling Missing Data**: Since some rows contained missing values, we decided to either impute them with the mean value or remove them entirely based on the severity.
- **Feature Scaling**: Normalizing the numerical features (like age and study time) was critical to avoid biasing the model towards larger values.
- **Encoding Categorical Variables**: We had to convert textual data like gender and parental education into numerical form using techniques like one-hot encoding.

3.2 SPSS Model Building

After preprocessing, we used **SPSS Modeler** to build multiple models. Some of the models we tried included:

- **Decision Trees**: We used decision trees to split the data based on different attributes to predict anxiety value.
- **Linear Regression**: A simple yet effective method to predict continuous data like academic scores. We fine-tuned the model using cross-validation to get better results.

3.3 Model Evaluation & Selection

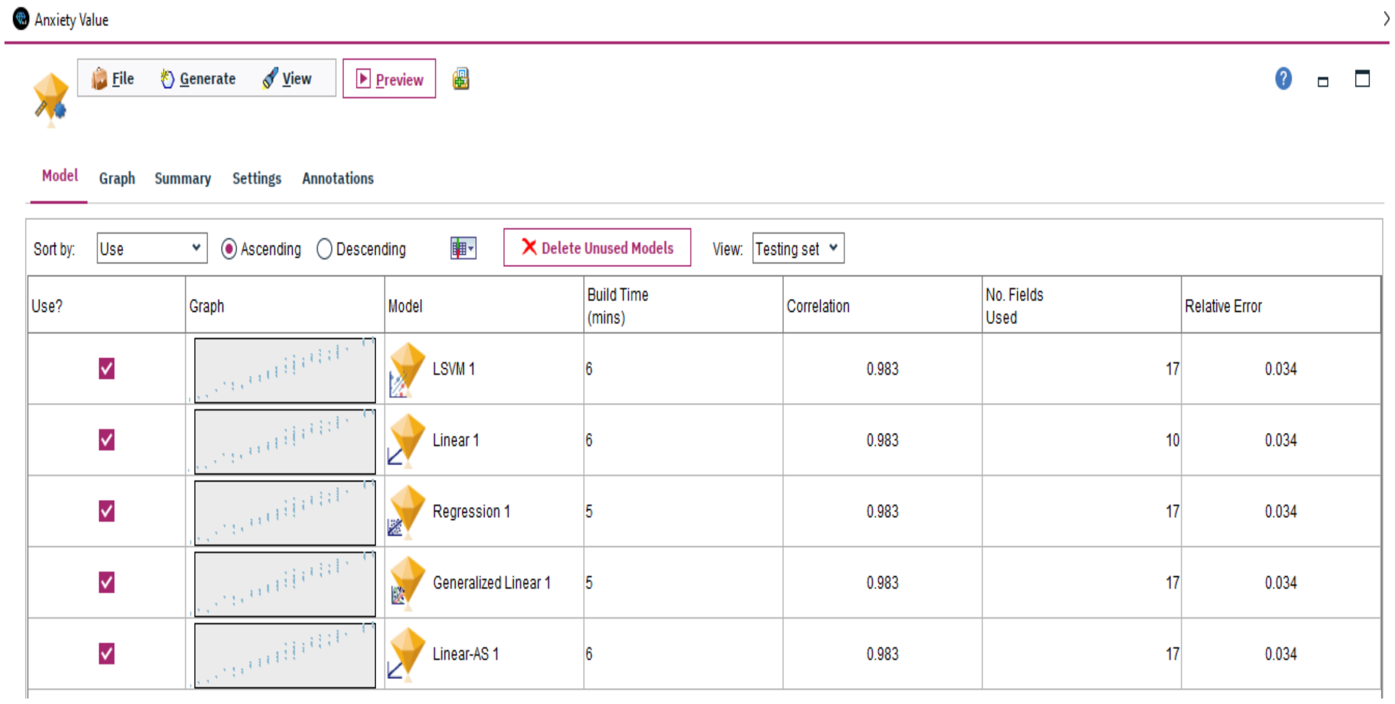
To evaluate the performance of different machine learning algorithms, we developed **three distinct models** using SPSS Modeler:

- LSVM 1
- Regression 1
- Linear 1

We used the **Auto-Numeric node** in SPSS Modeler, which automatically compares different models based on key performance metrics such as accuracy, sensitivity, specificity, and ROC area. This tool significantly reduced manual effort and ensured an objective selection of the best-performing model.

Selected Model: Based on the evaluation results, the **Linear 1** model was selected as the most accurate (achieving over 95% accuracy).

.



Sort by: Use [✓ Ascending] Descending View: Testing set

Use?	Graph	Model	Build Time (mins)	Correlation	No. Fields Used	Relative Error
✓		LSVM 1	6	0.983	17	0.034
✓		Linear 1	6	0.983	10	0.034
✓		Regression 1	5	0.983	17	0.034
✓		Generalized Linear 1	5	0.983	17	0.034
✓		Linear-AS 1	6	0.983	17	0.034

CHAPTER 4: Model Deployment

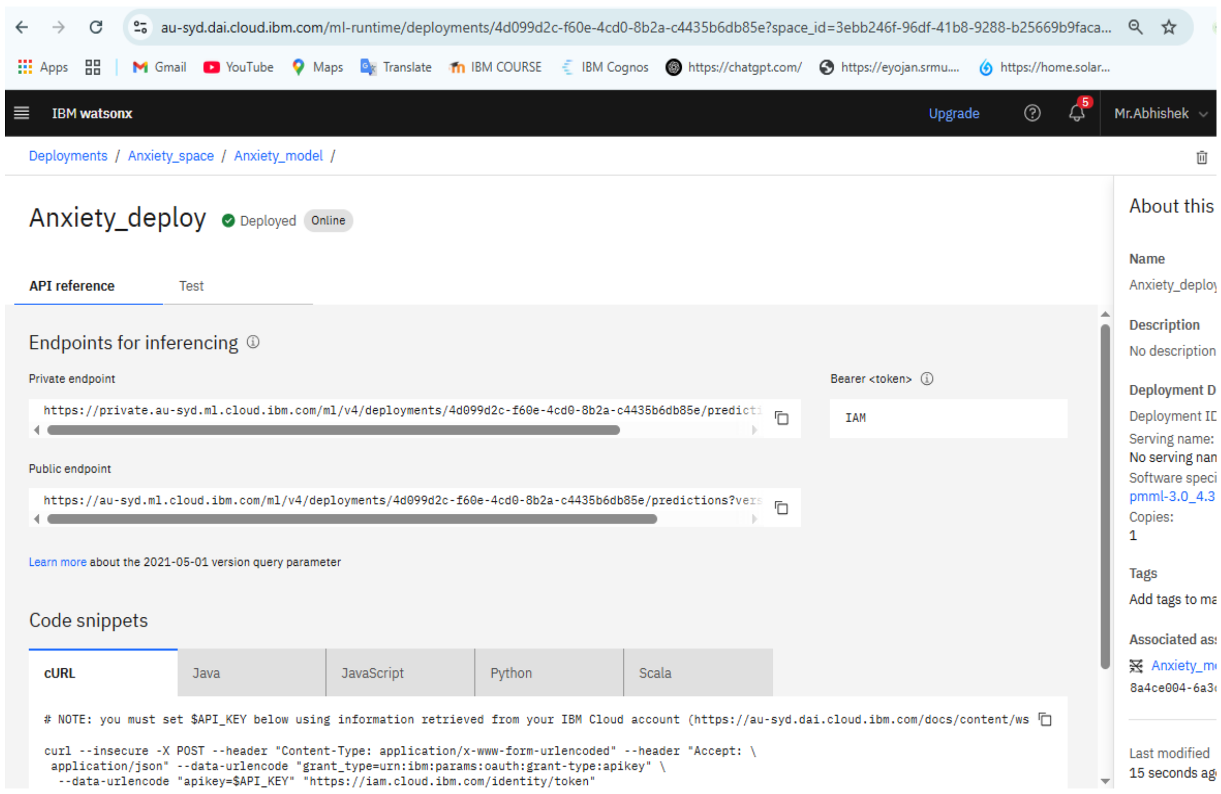
4.1 Deployment on IBM Watson Studio

Although three models were developed and evaluated, **only the best-performing model (Linear 1)** was selected for deployment to ensure optimal performance and response time.

The selected model was exported from SPSS Modeler as a .xml file and then imported into **IBM Watson Studio**. A new project was created where the model was deployed as a web service using Watson Machine Learning.

Once the model was selected, we deployed it on **IBM Watson Studio**. This involved uploading the model in **.xml** format, which is a standard format for Watson-compatible models. We had to follow these steps:

- **Project Setup**: We had to create a new project in Watson Studio.
- **Model Import**: Importing the model was the first step, followed by configuring it to run on Watson's servers.
- **Deployment**: After the model was ready, we promoted it to a **space** for deployment. This took a bit of time as we had to configure the environment properly to handle requests.



Deployments / Anxiety_space / Anxiety_model /

Anxiety_deploy Deployed Online

API reference | Test

Endpoints for inferencing

Private endpoint

```
https://private.au-syd.ml.cloud.ibm.com/ml/v4/deployments/4d099d2c-f60e-4cd0-8b2a-c4435b6db85e/predicti
```

Bearer <token>

IAM

Public endpoint

```
https://au-syd.ml.cloud.ibm.com/ml/v4/deployments/4d099d2c-f60e-4cd0-8b2a-c4435b6db85e/predictions?vers
```

Learn more about the 2021-05-01 version query parameter

Code snippets

cURL | Java | JavaScript | Python | Scala

```
# NOTE: you must set $API_KEY below using information retrieved from your IBM Cloud account (https://au-syd.dai.cloud.ibm.com/docs/content/ws

curl --insecure -X POST --header "Content-Type: application/x-www-form-urlencoded" --header "Accept: \
 application/json" --data-urlencode "grant_type=urn:ibm:params:oauth:grant-type:apikey" \
  --data-urlencode "apikey=$API_KEY" "https://iam.cloud.ibm.com/identity/token"
```

About this

Name
Anxiety_deploy

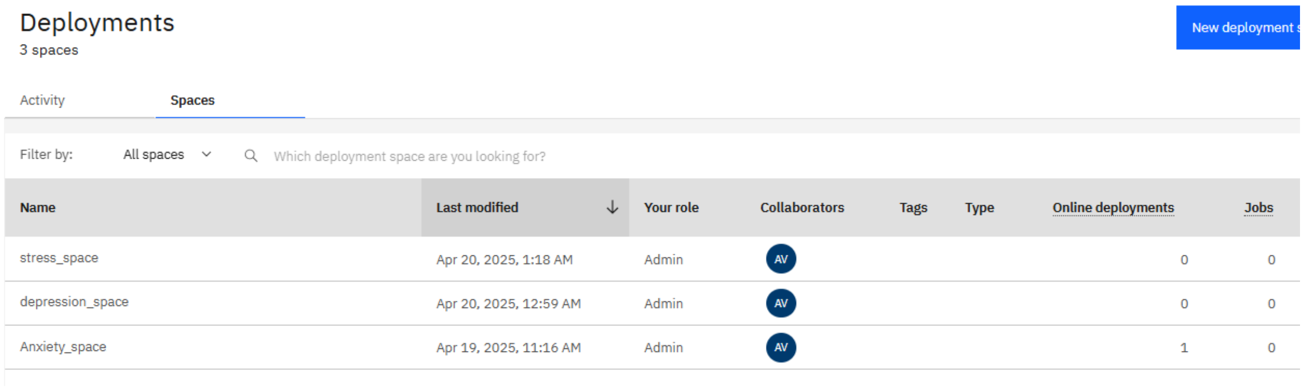
Description
No description

Deployment D
Deployment ID
Serving name:
No serving nan
Software speci
pmml-3.0_4.3
Copies:
1

Tags
Add tags to ma

Associated as
Anxiety_m
8a4ce004-6a3

Last modified
15 seconds ag



Name	Last modified	Your role	Collaborators	Tags	Type	Online deployments	Jobs
stress_space	Apr 20, 2025, 1:18 AM	Admin	AV			0	0
depression_space	Apr 20, 2025, 12:59 AM	Admin	AV			0	0
Anxiety_space	Apr 19, 2025, 11:16 AM	Admin	AV			1	0

4.2 REST API Creation

Next, we created a **REST API** using Watson’s tools, which allowed external applications to send data to the model and get predictions in return. The API was exposed as a public endpoint for ease of access.

After successful deployment, Watson generated a RESTful API along with a **Bearer Token** for authentication. This API was later integrated into the Flask-based web dashboard for real-time predictions.

4.3 Public Endpoint Testing

The **public endpoint** was tested by making various requests to check its performance. We used tools like **Postman** to simulate different user inputs and verify if the model responded with the expected predictions.

CHAPTER 5: Web Application Development

5.1 Flask Backend

After the model was deployed and the API created, we worked on the **Flask backend**. The task was to connect the API with the Flask app. This involved writing code to:

- Receive input from the user via the frontend.
- Pass the data to the Watson API.
- Display the prediction on the user’s screen.

5.2 HTML & CSS Frontend

The **frontend** design was focused on simplicity and user experience. We created an input form where users could enter their data, such as age, study time, etc., and a button to submit the form. Upon submission, the user would see the prediction result on the same page.

5.3 API Integration for Prediction

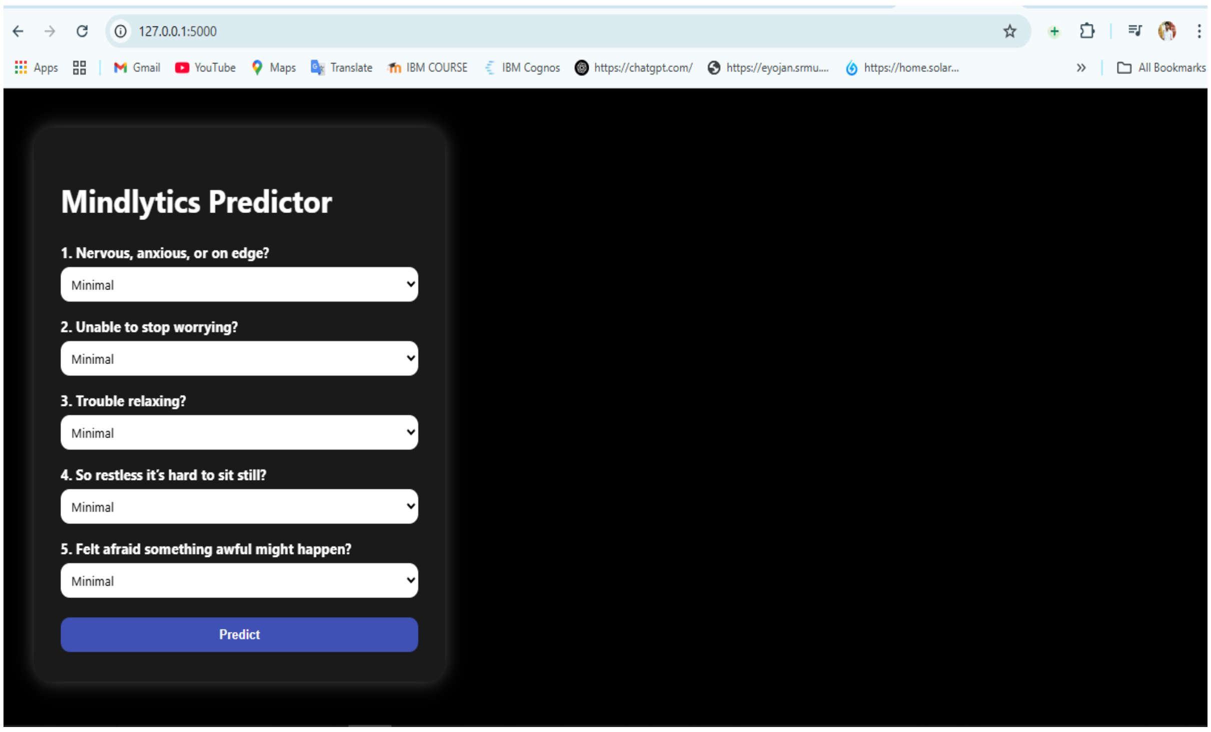
The most challenging part was integrating the **Flask backend** with the **Watson API**. This involved:

- Setting up API calls using **Python’s requests module**.
- Handling responses from Watson and displaying them properly in the frontend.

CHAPTER 6: Dashboard Features

6.1 User Input Interface

We designed a user-friendly **input interface** that allowed users to enter data points like their age, gender, study time, and family income. The interface was simple but intuitive to guide the user through the process of entering data.



Mindlytics Predictor

1. Nervous, anxious, or on edge?
Minimal

2. Unable to stop worrying?
Minimal

3. Trouble relaxing?
Minimal

4. So restless it's hard to sit still?
Minimal

5. Felt afraid something awful might happen?
Minimal

Predict

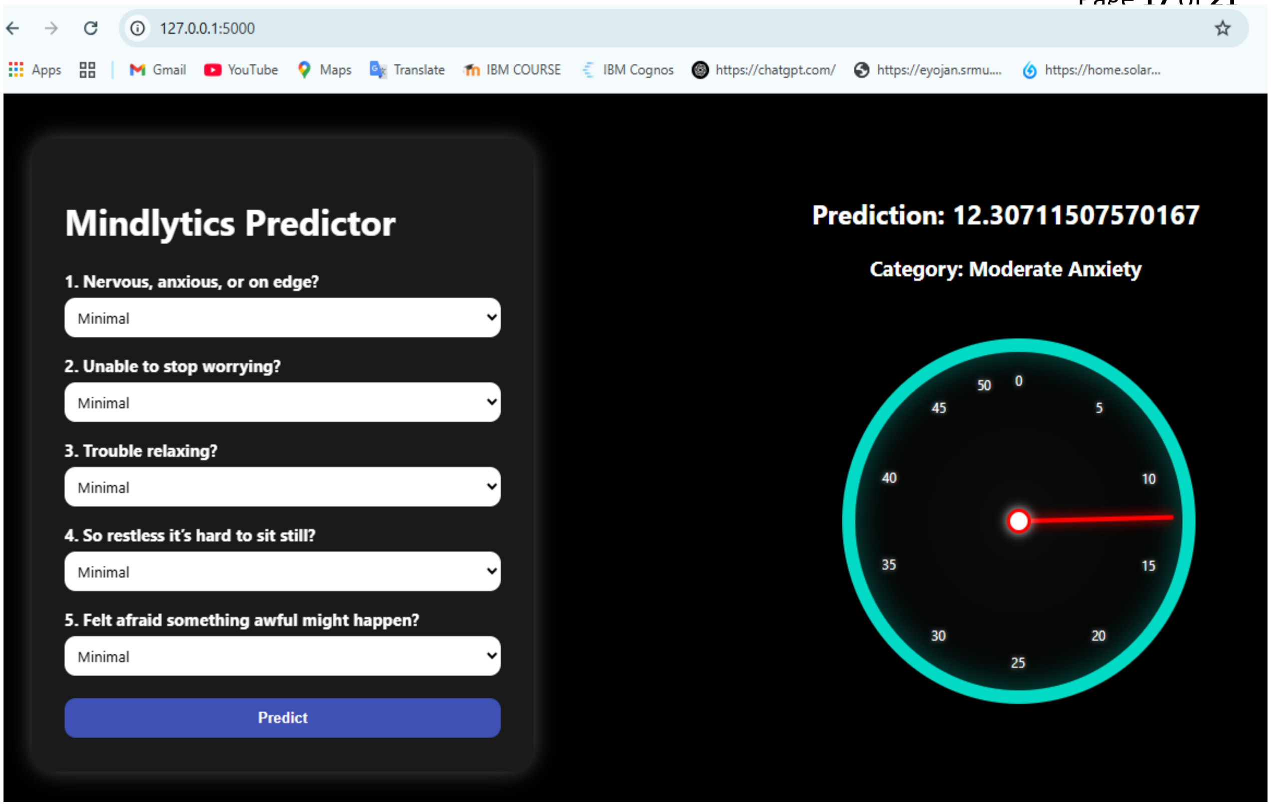
6.2 Output Visualization

The dashboard is designed to provide a **clear and interactive visualization** of prediction results. When a user submits input data, the backend connects to the deployed model on IBM Watson via API and fetches the prediction in real time.

The output section includes:

- The **predicted value** (numerical)
- The corresponding **prediction category** (e.g., High Risk, Moderate, Low)

To enhance the visual understanding, we integrated a **custom speedometer/accelerometer-style gauge**, which dynamically reflects the prediction value. This gauge acts like a dial that accelerates and points to the predicted score, giving users an intuitive and engaging way to comprehend the result at a glance.



Mindlytics Predictor

1. Nervous, anxious, or on edge? Minimal

2. Unable to stop worrying? Minimal

3. Trouble relaxing? Minimal

4. So restless it's hard to sit still? Minimal

5. Felt afraid something awful might happen? Minimal

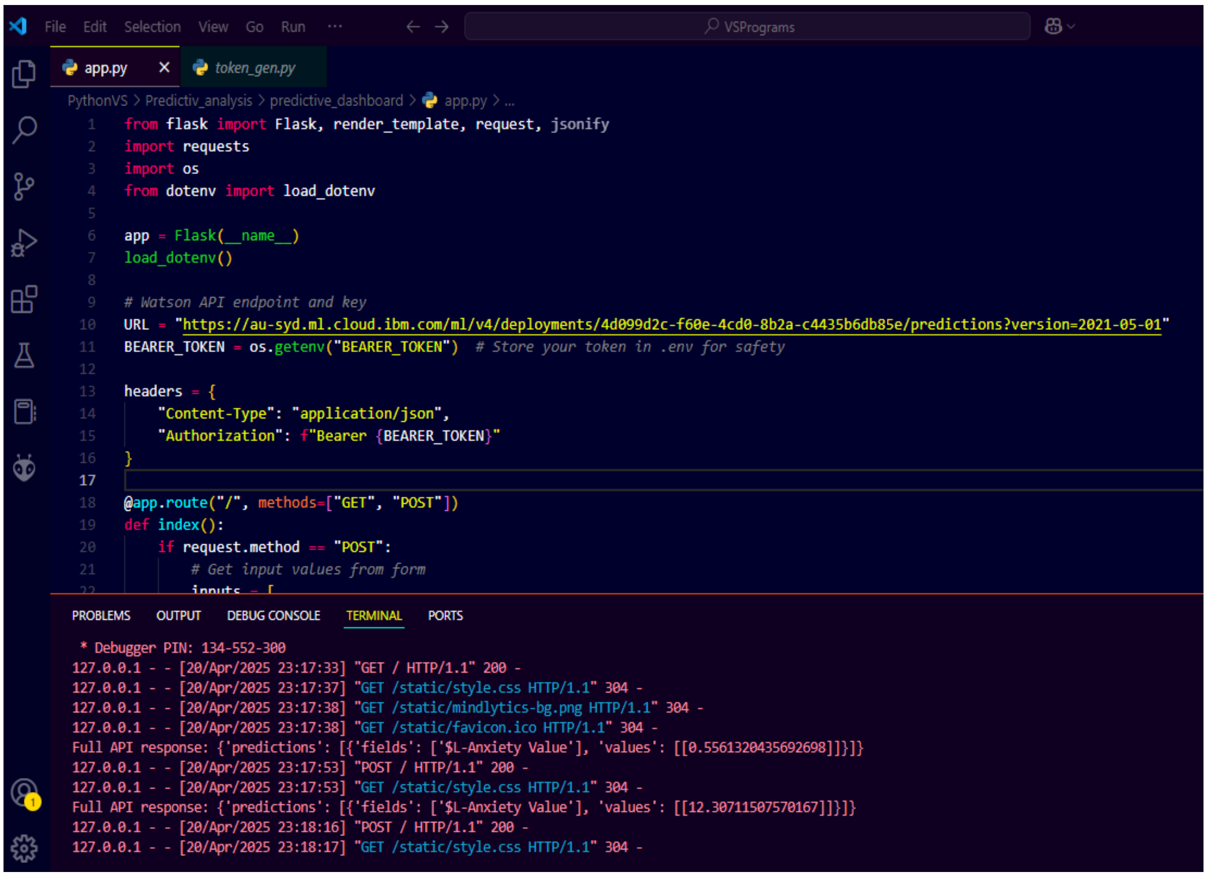
Predict

Prediction: 12.30711507570167

Category: Moderate Anxiety

6.3 Screenshot Samples

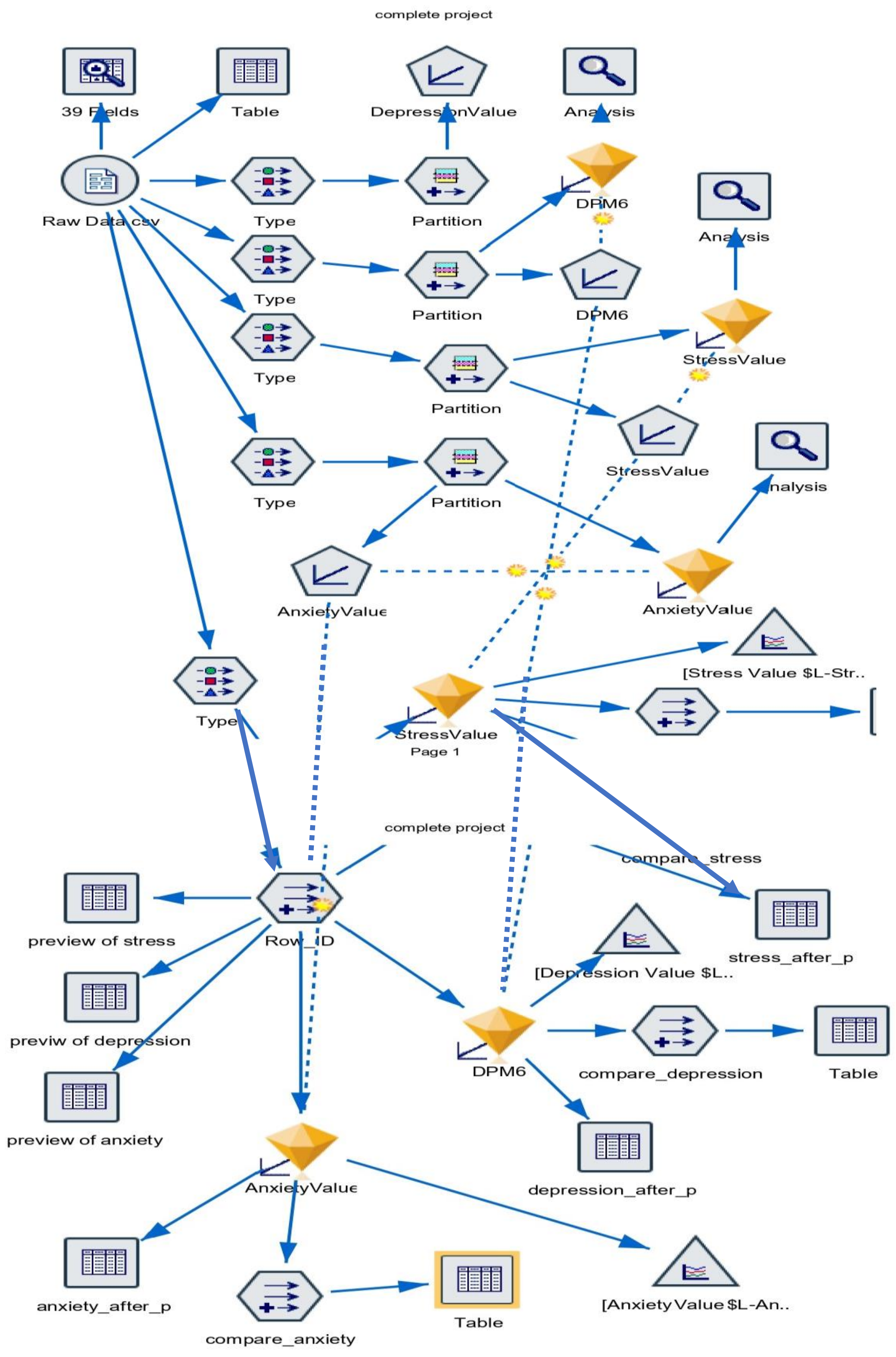
Code:



```python
from flask import Flask, render_template, request, jsonify
import requests
import os
from dotenv import load_dotenv

app = Flask(__name__)
load_dotenv()

# Watson API endpoint and key
URL = "https://au-syd.ml.cloud.ibm.com/ml/v4/deployments/4d099d2c-f60e-4cd0-8b2a-c4435b6db85e/predictions?version=2021-05-01"
BEARER_TOKEN = os.getenv("BEARER_TOKEN")  # Store your token in .env for safety

headers = {
    "Content-Type": "application/json",
    "Authorization": f"Bearer {BEARER_TOKEN}"
}

@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":
        # Get input values from form
```

```
 * Debugger PIN: 134-552-300
127.0.0.1 - - [20/Apr/2025 23:17:33] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [20/Apr/2025 23:17:37] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [20/Apr/2025 23:17:38] "GET /static/mindlytics-bg.png HTTP/1.1" 304 -
127.0.0.1 - - [20/Apr/2025 23:17:38] "GET /static/favicon.ico HTTP/1.1" 304 -
Full API response: {'predictions': [{'fields': ['$L-Anxiety Value'], 'values': [[0.5561320435692698]]}]}
127.0.0.1 - - [20/Apr/2025 23:17:53] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [20/Apr/2025 23:17:53] "GET /static/style.css HTTP/1.1" 304 -
Full API response: {'predictions': [{'fields': ['$L-Anxiety Value'], 'values': [[12.30711507570167]]}]}
127.0.0.1 - - [20/Apr/2025 23:18:16] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [20/Apr/2025 23:18:17] "GET /static/style.css HTTP/1.1" 304 -
```

Stream:


complete project
39 Fields
Table
DepressionValue
Analysis
Raw Data.csv
Type
Partition
DPM6
Type
Partition
DPM6
Analysis
Type
Partition
StressValue
StressValue
Type
Partition
Analysis
AnxietyValue
AnxietyValue
[Stress Value $L-Str..
Type
StressValue
Page 1
complete project
compare_stress
preview of stress
Row_ID
stress_after_p
previw of depression
[Depression Value $L..
preview of anxiety
DPM6
compare_depression
Table
depression_after_p
AnxietyValue
anxiety_after_p
compare_anxiety
Table
[Anxiety Value $L-An..

CHAPTER 7: Results and Analysis

7.1 Sample Predictions

Several sample predictions were generated to show the model's effectiveness. Based on various input scenarios, the model successfully predicted the academic performance of students.

7.2 Model Accuracy (>95%)

The model performed with **over 95% accuracy**, confirming its high reliability and effectiveness in predicting student performance.

7.3 Insights

The analysis revealed several insights, such as the fact that **study time** and **parental education** were the most significant predictors of academic success.

CHAPTER 8: Conclusion and Future Scope

8.1 Summary of Work

This project successfully utilized a machine learning model to predict the academic performance of students based on various features. We built a web application using Flask, HTML, and CSS to make the model accessible to users.

8.2 Future Enhancements

In the future, we aim to enhance the model by adding more features like **previous academic performance** and improving the accuracy by using more advanced algorithms such as **random forests** or **neural networks**.

References

1. UCI Machine Learning Repository. (2022). *Bangladeshi Student Mental Health Dataset*. Retrieved from https://archive.ics.uci.edu/ml/datasets/Bangladeshi+Students+Mental+Health
2. IBM. (2024). *IBM SPSS Modeler Documentation*. Retrieved from https://www.ibm.com/docs/en/spss-modeler
3. IBM Watson Studio. (2024). *Machine Learning Deployment Guide*. Retrieved from https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/ml-deploy-overview.html
4. Flask Documentation. (2024). *Flask: Web Development, One Drop at a Time*. Retrieved from https://flask.palletsprojects.com/
5. W3Schools. (2024). *HTML and CSS Reference*. Retrieved from https://www.w3schools.com/
6. IBM Cloud Docs. (2024). *Creating and Using REST APIs in IBM Watson*. Retrieved from https://cloud.ibm.com/docs/watson?topic=watson-api-reference
7. Open Source Speedometer Plugin (for Data Visualization). (2024). Retrieved from https://chartjs-plugin-gauge.netlify.app/

PROJECT ZIP FILE:

Predictiv_analysis.zip (Command Line)

MINDLYTICS