PROJECT REPORT CODING WEEK

📄 CampusPulse: Task 1 Project Report

1. Introduction

The objective of Task 1 in the CampusPulse project is to perform student data analysis and build predictive models based on survey results. This includes exploratory data analysis, preprocessing, model training, and evaluation.

This report documents the step-by-step approach I took to analyze the data, perform preprocessing, experiment with different machine learning models, and evaluate the final results.

2. Step-by-step Approach

2.1 Initial Inspection

The dataset was loaded and the first few rows were examined using .describe() and .info() methods. Key aspects such as the number of rows and columns, data types, and presence of null values were reviewed.

Then I started with the level 1 by drawing the histogram and heatmap for the dataset trying to find the anonymous features.

Based on the observations from the plots I guessed the intermediate values of the features (cant find any better till the end 😟).

The features were identified to be

- Feature 1 – Age
- Feature 2 – Daily hours of study
- Feature 3 – Partying frequency

Also explored about the possibility of them being anything else like screentime, travel time to school etc. best they weren't.

2.2 Data Preprocessing

Columns with missing values were identified.

Imputation strategies varied by feature type:

- Numerical features were imputed with mean or median.
- Categorical features were imputed with mode.
- Ratings were imputed with median.

2.3 Exploratory Data Analysis

As per the objective asked 5 meaningful questions that reveal more interesting facts about the data (summarised in notebook).

Also plotted a variety of graphs for the same.

Plotted all possible graphs for the things initially but then chose the final version, sometimes because they gave more information but mostly because they looked better (Visual Impression matters afterall 😊)

Also some popular stereotypes were debunked looking at the data.

2.4 Model Building

So started with feature scaling (because logistic regression wont work otherwise, tried to do everything else). Used the standard scalar to scale the features.

A variety of models were evaluated then:

- **Logistic Regression**: Started with this but gave very bad accuracy.
- **Random Forest**: Initially did without hyperparameter tuning, later on added gridsearchCV also used all the 5 hyperparameter tuning methods as mentioned in the resources, all gave bad accuracy.
- **Naïve Bayes**: Gave a slightly better accuract and F1 score than the others, my final pick.
- **KNN**: Added later on after level 5 was done, still not better.

So the best accuracy achieved was 72%. Also made the confusion matrix and classification report for each.

Yeah also worked with the parameters of gridsearch and randomsearch like max depth etc. whatever was there don’t know about all of them.

2.5 Model Analysis

Made the SHAP plots for the best model and wrote down the inferences.

Also drew decision boundaries for each model.

3. Bonus Task

Based on the decision boundaries from the above I initially classified the plots but later on realising it was wrong changed those to the correct ones.

The Rise of the WeatherMind: Task 2 Project Report

1. Introduction

This project showcases the construction of an advanced LangChain-based intelligent agent using **LangGraph**. The goal is to create a robust, modular, and scalable agent capable of reasoning, information retrieval, and dynamic tool invocation.

2. Step-by-step Approach

2.1 Level 1:

Constructed a langraph node called chatbot using Gemini API and added a calculator tool to it, also visualised its graph.

Here initially I did something wrong following the wrong section of documentation making it difficult to proceed further however corrected that on time and made it the same way as it is documentation.

Also with static prompts I faced the problem where the calculator tool would return no value, however changed the prompt and did that.

2.2 Level 2:

Added Tavily Search and Open Weather to the chatbot using APIs, ah the video of weather API in the resource was very insufficient, so had to refer to documentations for that too.

Did everything as was in documentations, used references on the internet for syntax issues (links in the notebook).

2.3 Level 3:

Added checkpointer to it so that it would remember previous interactions.

2.4 Level 4:

Implemented multi-agent support using supervisor (one of the two ways mentioned in the documentation), also added some dummy agents like hotel booking and flight booking and also separated the previously made tools into new different agents ensuring all can collaborate smoothly by implementing routing logic between agents through supervisor.

Experimented with different prompts to the tools ensuring this is the best version.