TASK 1 : THE CAMPUS PULSE INITIATIVE

Task Overview :

We had a student data set of 649 students studying in 2 higher secondary schools collected from a survey. Our objective was to analyze the dataset and help uncover how hidden factors influence student life on campus, from grades to relationships. We also had to design a model that can predict whether a student is in a romantic relationship.

LEVEL 1

Problem Statement :

As part of a privacy-preserving exercise, three survey features were anonymized and renamed as Feature_1, Feature_2, and Feature_3. Our task was to uncover the original identities of these features by analyzing statistical patterns without relying on raw labels. This exercise simulates real-world data forensics, where understanding hidden variables is crucial.

Objective :

- Use EDA techniques like histograms, scatter plots, line charts, and correlation analysis to guess the original meanings of anonymized features.
- Provide short textual and visual justifications for each identification.

Implementation Approach :

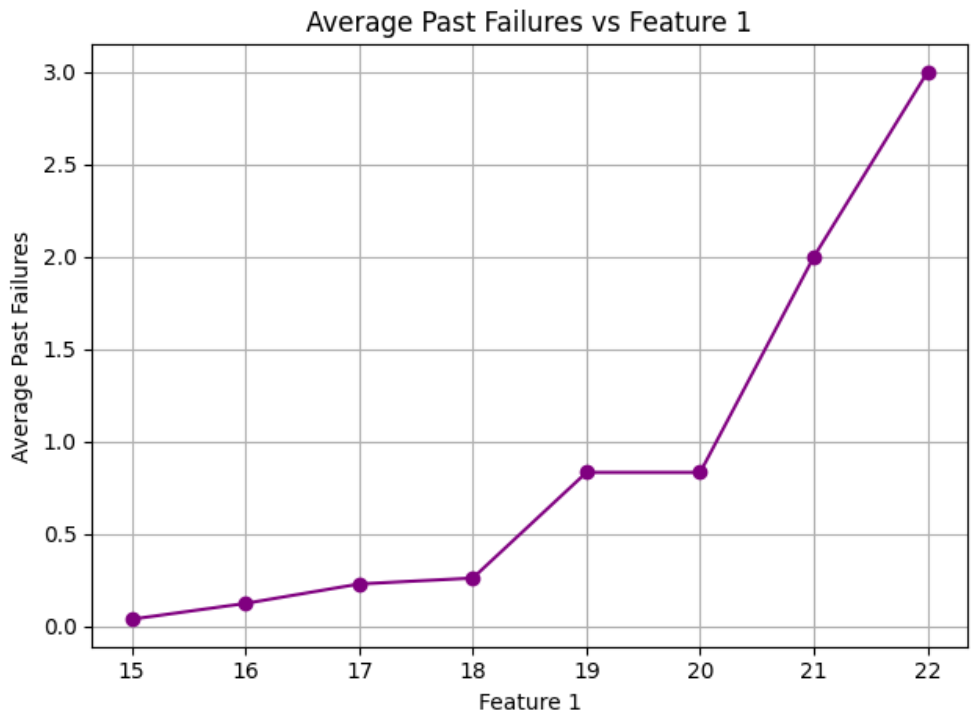
1. **Initial Exploration:**
   - Loaded the dataset using pandas.
   - Installed numpy, matplotlib and seaborn
   - Examined basic statistics and value distributions. (head() and columns)
   - Checked for duplicates
2. **Visual Analysis:**
   - Plotted value counts for Feature_1, Feature_2, and Feature_3 to understand their value ranges and distributions.
   - Feature_1 analysis : Integer values from **15 to 22** are all reasonable ages for **high school or college students**. Peaks at **16–17**, slowly dropping after that. Plotted graph of Feature_1 vs average of past class failures to check whether the outliers are still studying in school because maybe they failed more.
   - Feature_2 analysis : A small discrete set of numeric values suggests that Feature 2 is a categorical feature. Given the **distribution skewed toward 2.0 and 1.0**, it's likely to be a variable where **Lower is more common** and **Higher levels are less frequent**. This suggests it can probably be study time or stress levels (as lower values are more common). Plotted graph of Feature_2 vs Final Grade (G3) to check for a positive correlation.

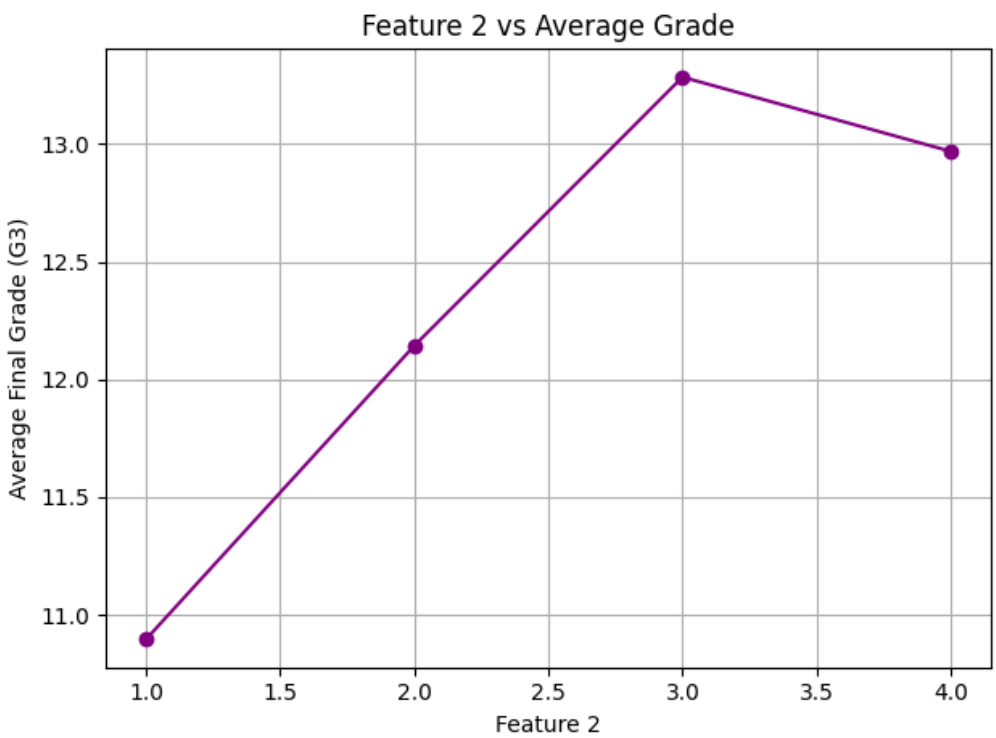
- Feature_3 analysis : This **strongly suggests an ordinal variable**, and the shape of the distribution can help us guess what it likely measures. The majority of students has a value of 1 or 2, and very few has 5. An interesting match with Dalc value counts analysis shows remarkable similarities in value distribution, suggesting it could be Weekend alcohol consumption (Walc) also showing the students tend to drink alcohol more on weekends. To confirm, we plot value counts graph with Dalc and also graphs vs freetime and gout to check for positive correlation.

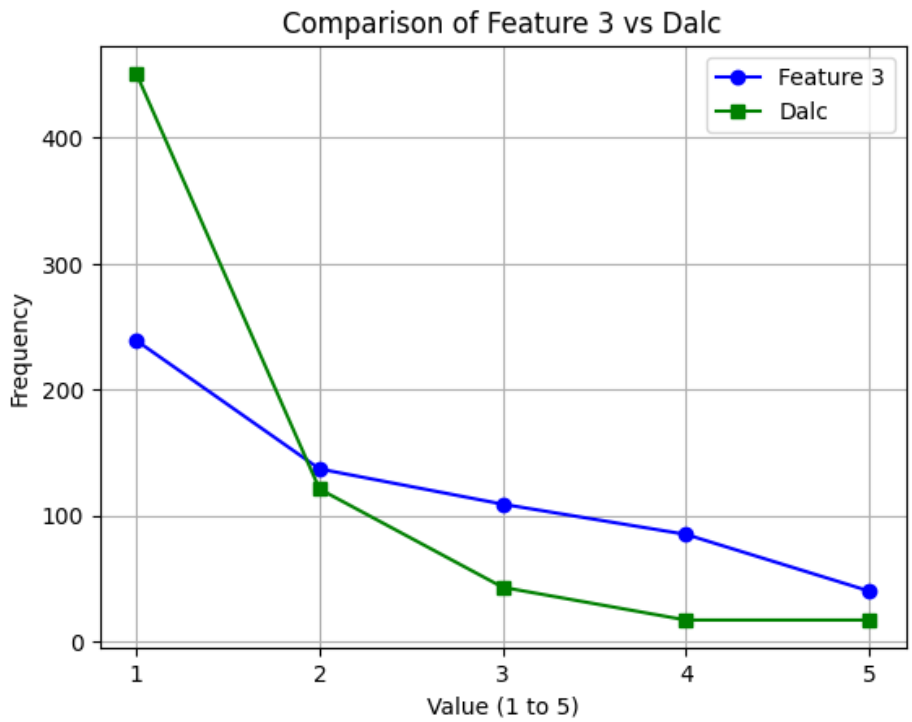
3. **Feature Matching:**

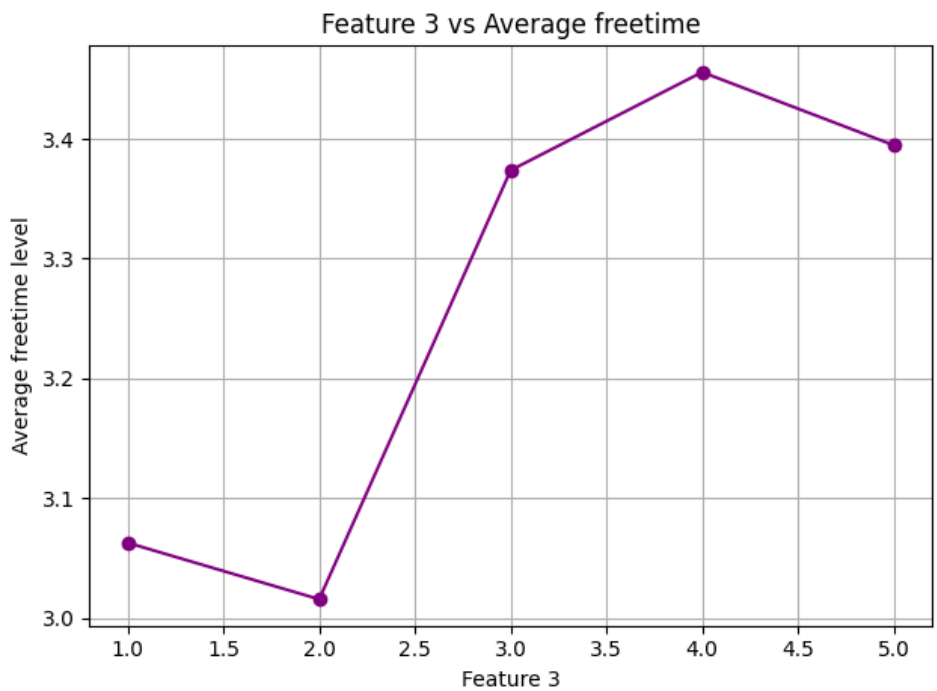
- Matched Feature_1 using a strong positive correlation to average past class failures, resembling **AGE** showing students after ages 19-20 had 2-3 past class failures, suggesting why they are still in school.
- Matched Feature_2 using a strong positive correlation to G3, resembling **STUDYTIME** (with a slight dip at the end showing overstudying can diminish grades)
- Matched Feature_3 via visual similarity and moderate correlation with social behavior(freetime and gout) and weekday alcohol usage, indicating **WALC.**

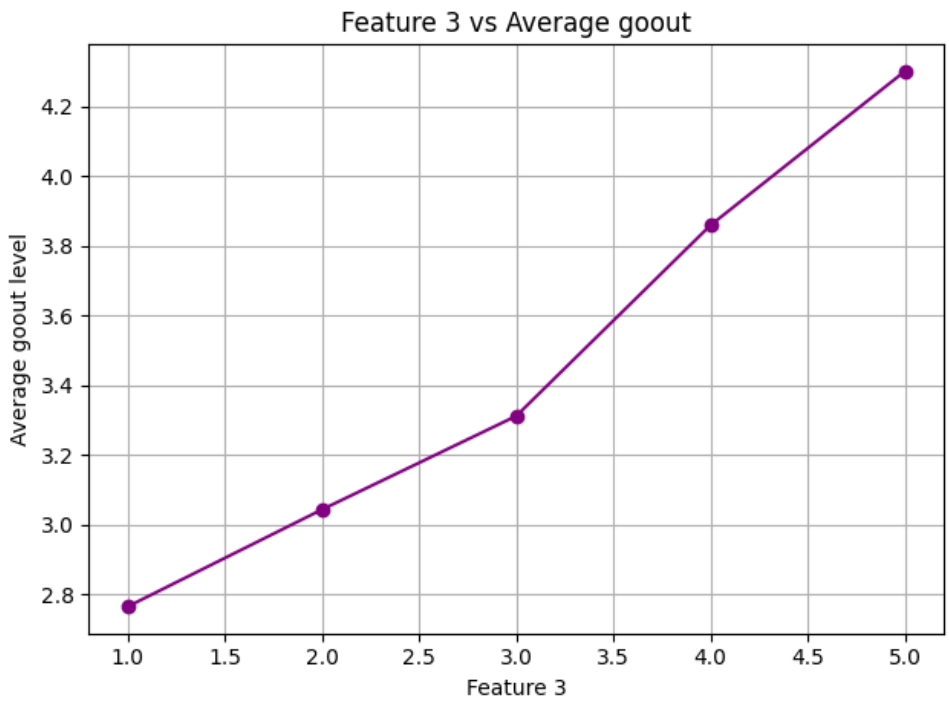
Visualizations :






Comparison of Feature 3 vs Dalc
Feature 3
Dalc
Frequency
0
100
200
300
400
1
2
3
4
5
Value (1 to 5)


Feature 3 vs Average freetime
Average freetime level
3.0
3.1
3.2
3.3
3.4
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Feature 3



Conclusion :

- **Feature_1 = AGE**
- **Feature_2 = STUDYTIME**
- **Feature_3 = WALC**

LEVEL 2

Problem Statement :

Survey datasets often suffer from quality issues due to skipped questions, invalid entries, or inconsistent responses. Before deploying the CampusPulse system, it's crucial to clean the data and ensure its integrity. This involves identifying missing values and choosing appropriate methods to fill them in, improving data reliability without introducing bias.

Objective

- Detect all features with missing values (nulls).
- Apply suitable imputation strategies for each feature.
- Provide brief justifications for each strategy based on feature type and context.

Implementation Approach

1. **Missing Value Detection**:
    - Used pd .isnull().sum() to find features with missing entries and also found their value counts.
    - Found the rows with >= 4 nulls.
    - Analyzed the type of each feature (categorical or numerical) to guide imputation.

2. **Strategy Selection**:
    - Dropped the rows with >= 4 nulls
    - **Categorical Features**: Imputed with **mode** (most frequent value).
    - **Numerical Features**: Imputed with **mean** or **median** based on skewness and presence of outliers.
    - Imputated G2 with max(G1, G3) as I analyzed that almost in every row, the 3 values of G1, G2 and G3 are almost same. So I substituted the max value (for positive side) of G1 and G3 to fill missing values of G2

Conclusion

We successfully detected all the features with nulls, applied the required imputation strategies and filled the missing values.

LEVEL 3

Problem Statement :

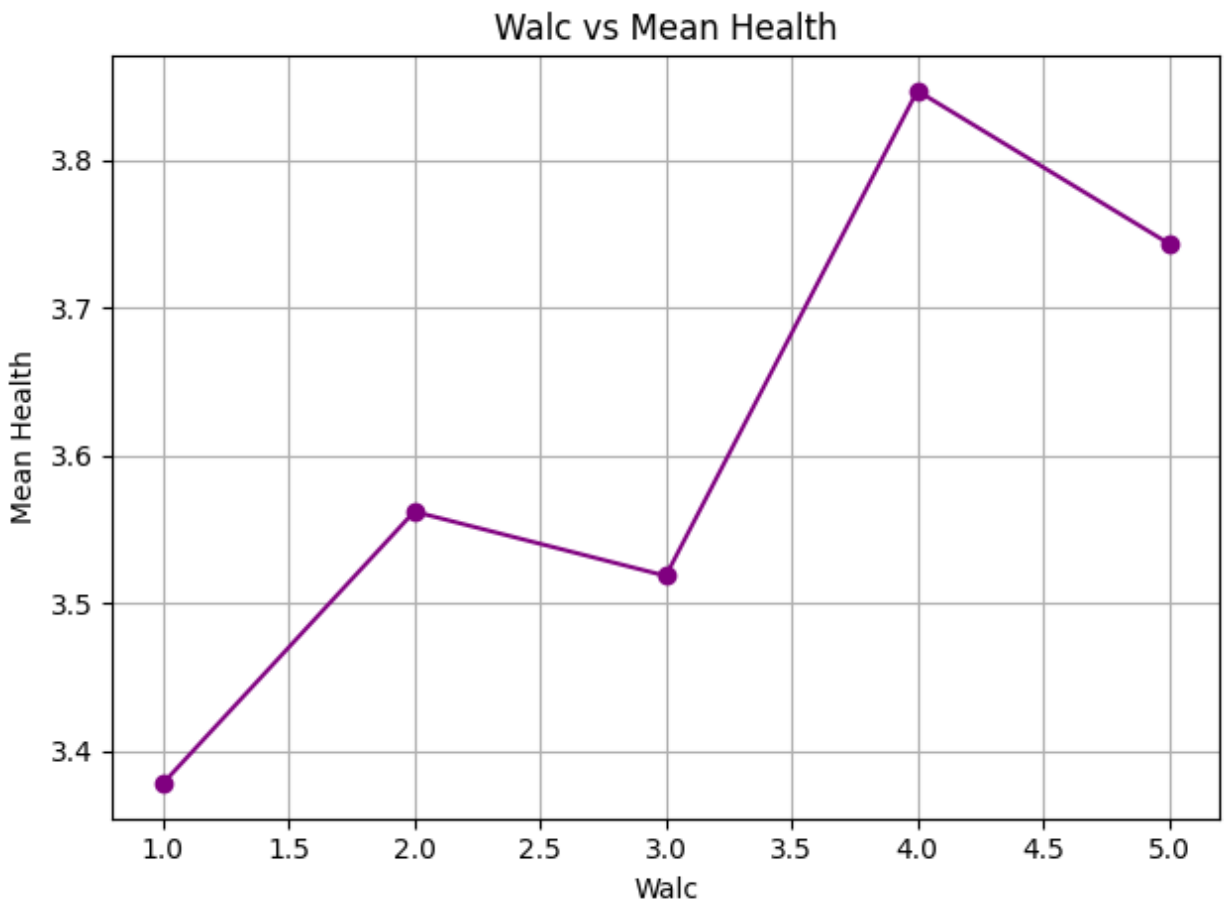
Understanding student behavior, academic performance, and well-being is critical for developing effective campus-wide interventions. However, student data is complex and often underutilized. CampusPulse aims to be more than just a prediction tool — it should help uncover meaningful patterns and provide actionable insights about students' academic habits, health, relationships, and family dynamics. A focused Exploratory Data Analysis (EDA) is required to bring these hidden patterns to light.

Objective :

- Conduct a targeted Exploratory Data Analysis (EDA) on the student dataset.
- Ask at least five insightful questions to guide the exploration.
- For each question:
  - Create a meaningful plot (e.g., bar chart, violin plot, scatterplot, etc.).
  - Write a brief interpretation of the insights drawn.
- Showcase curiosity, clarity, and visual storytelling to support data-driven decision-making for the CampusPulse platform.

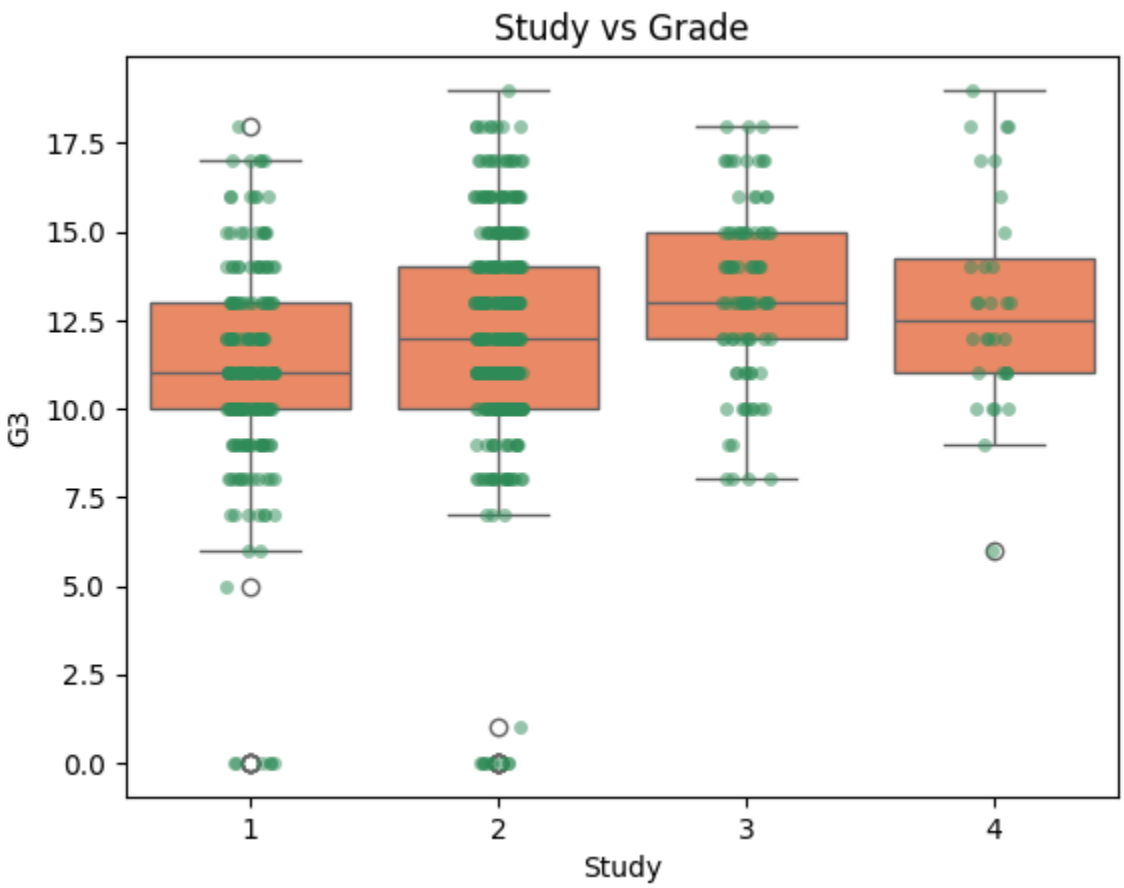
1. Line Graph: Average Health vs. Weekend Alcohol Consumption

- **Insight**:
  The line graph depicts the relationship between students' **weekend alcohol consumption** (rated from 1 = very low to 5 = very high) and their **self-reported health status** (on a scale from 1 to 5). Surprisingly, the data reveals that **students who report higher alcohol consumption on weekends tend to rate their health slightly better**, especially at level 4 consumption, which corresponds to the **highest observed average health score (~3.85)**. The trend is **non-linear**, with a dip at level 3 and a slight decrease after level 4.
- **Interpretation**:
  This counterintuitive result may indicate:
- A **perception bias**, where students with active social lives (often involving alcohol) perceive themselves as healthier.
- A **possible influence of age**, lifestyle, or physical resilience in the short term—though excessive alcohol typically leads to long-term health deterioration.
- Alternatively, it might reflect **inaccuracies in self-reported health** or that the students drinking more on weekends are more physically active or less academically stressed.


Walc vs Mean Health
Mean Health
3.8
3.7
3.6
3.5
3.4
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Walc

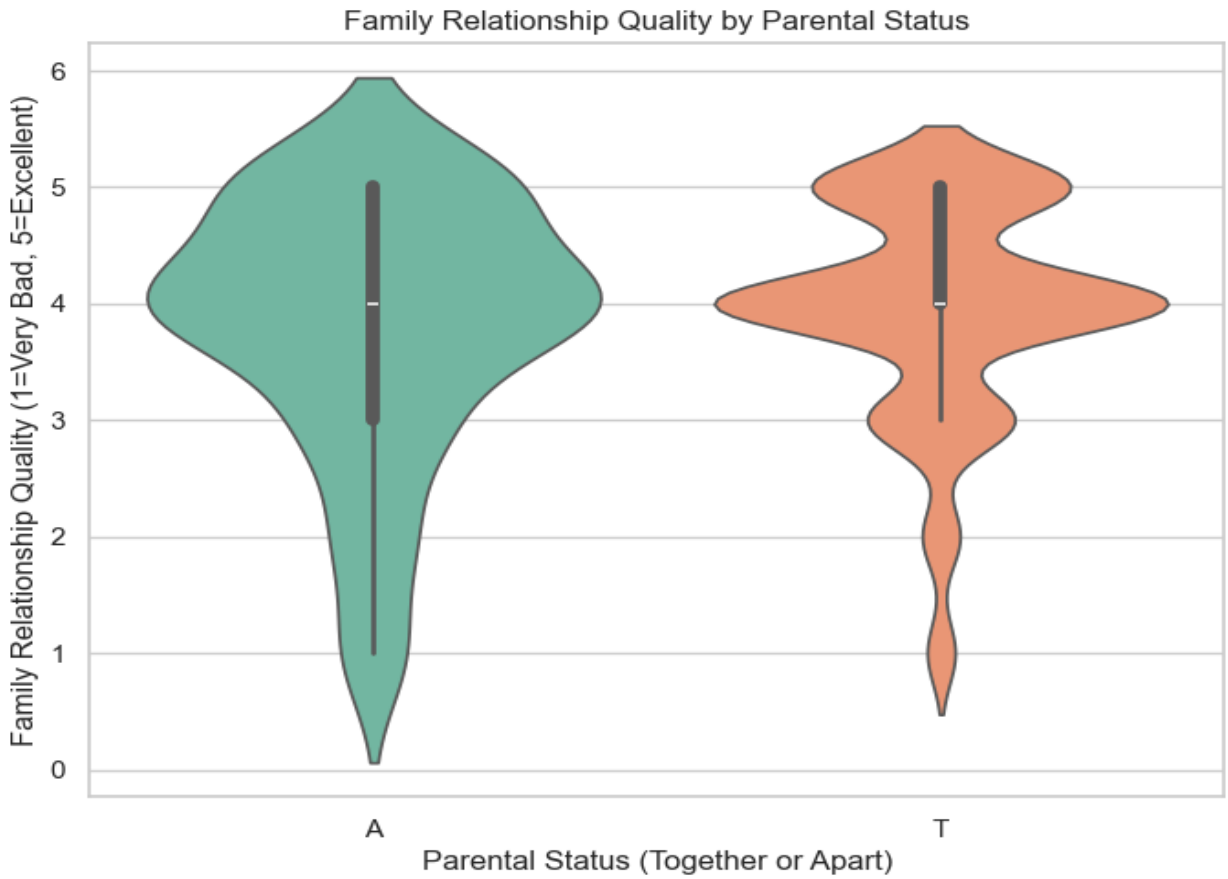
2. Box Plot with Jitter: Study Time vs. Final Grade

- **Insight**:
  There is a slight **positive trend** between study time and final grades. Students who study longer hours (especially 3 to 4 hours daily) tend to have slightly **higher final grades**, with fewer low outliers.
- **Interpretation**:
  This supports the **expected link between study discipline and academic performance**, emphasizing the benefits of consistent study habits.



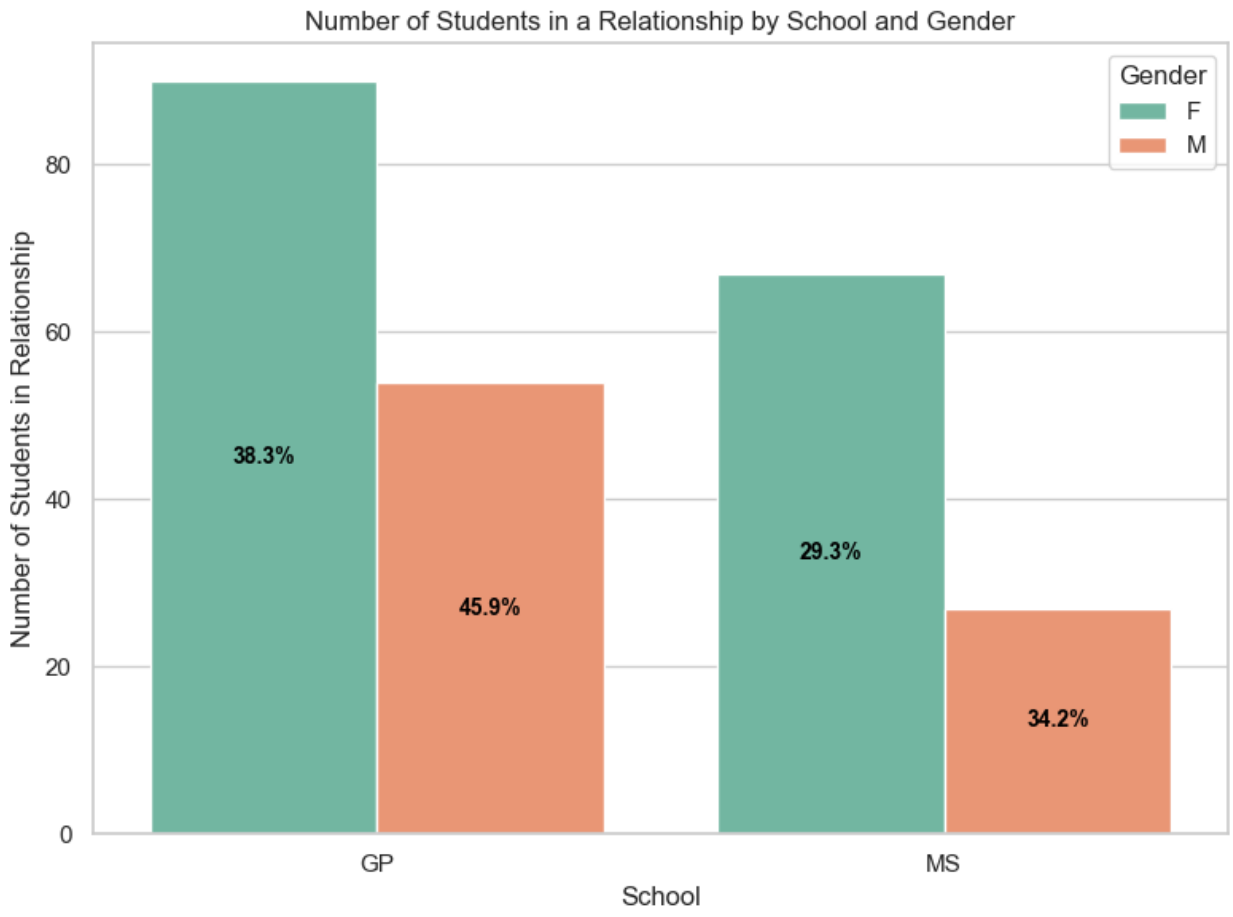
3. Violin Plot: Parental Status vs. Family Relationship Quality

- **Insight**:
  Students with **parents living together** tend to report a **higher and more consistent family relationship quality**, as seen from the wider and more centralized density in the violin plot. In contrast, students with **apart parents** show more variability and lower median values.
- **Interpretation**:
  Parental marital status may influence the **emotional and relational environment** at home, which can affect a student's overall well-being.



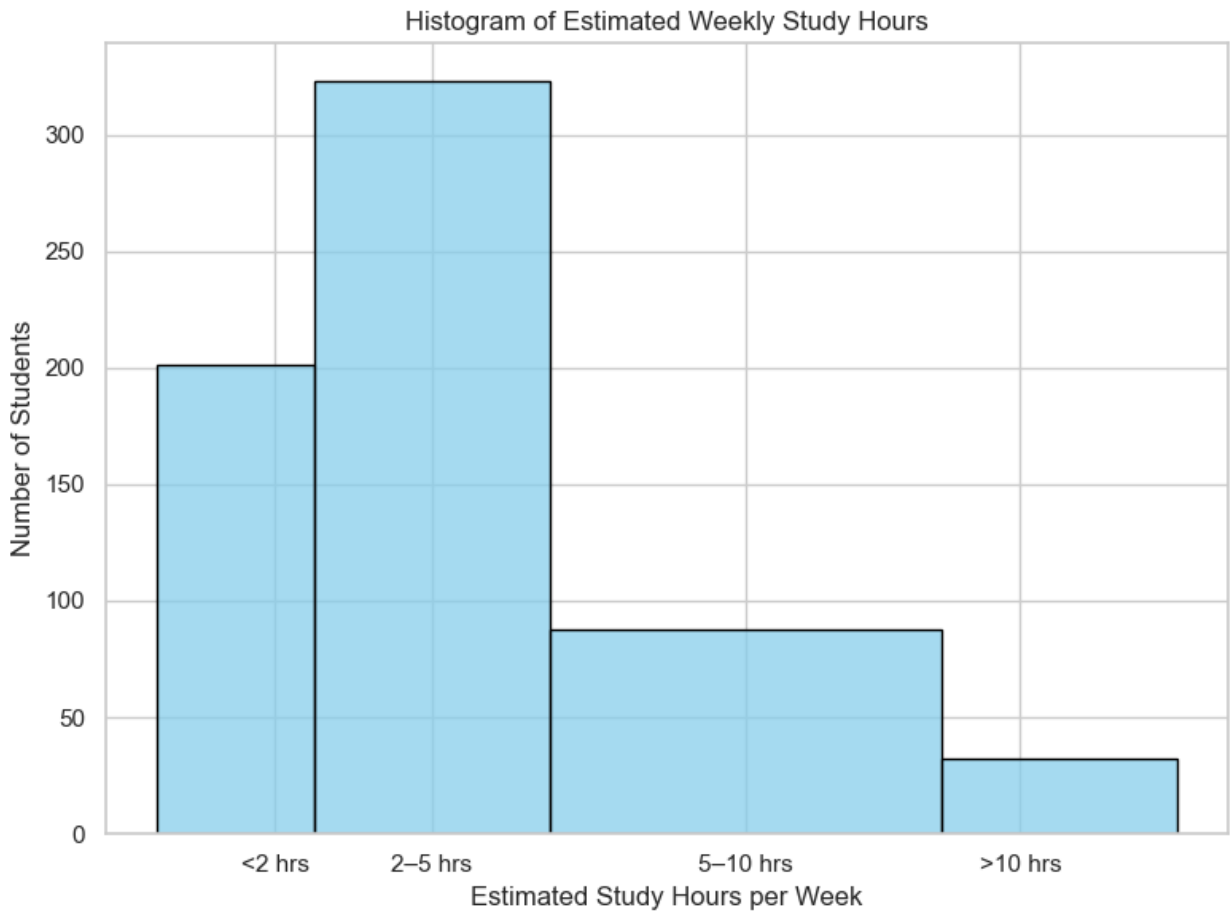
4. Bar Chart: Students in a Relationship by School and Gender

- **Insight**:
  More **female students** in both schools appear to be in relationships compared to male students, however more percentage of male students are in relationships. The school GP has a significantly **higher proportion** of students reporting relationships.
- **Interpretation**:
  This could reflect **social or cultural dynamics** at different schools and how they influence relationship tendencies across genders.



5. Histogram: Estimated Weekly Study Hours

- **Insight**:
  The distribution of weekly study hours is **right-skewed**, with most students studying between **2 to 5 hours per week**, and very few crossing the **20-hour** mark.
- **Interpretation**:
  This reveals that while some students are highly committed, the majority engage in **moderate to low study intensity (up to 5 years)**, which may inform targeted academic support initiatives.



Conclusion :

- These insights provide a **multifaceted view of student life**, touching on health, academics, relationships, and family factors. Together, they enable stakeholders to design more informed interventions and policies in the CampusPulse system.

LEVEL - 4

Problem Statement :

In this analysis, we aim to uncover which academic, behavioral, or social factors influence a student's likelihood of being in a romantic relationship. Using data-driven classification techniques, we attempt to build predictive models that can determine whether a student is in a relationship based on their personal, academic, and lifestyle attributes.

Objective :

The primary objective is to apply classification algorithms to predict the relationship status of students and assess the performance of each model. Additionally, we aim to:

- Interpret the most influential features contributing to predictions using feature importance.
- Simplify the models by selecting only the top features and re-evaluate performance.
- Derive meaningful insights from the results to understand the social and behavioral factors influencing relationships.

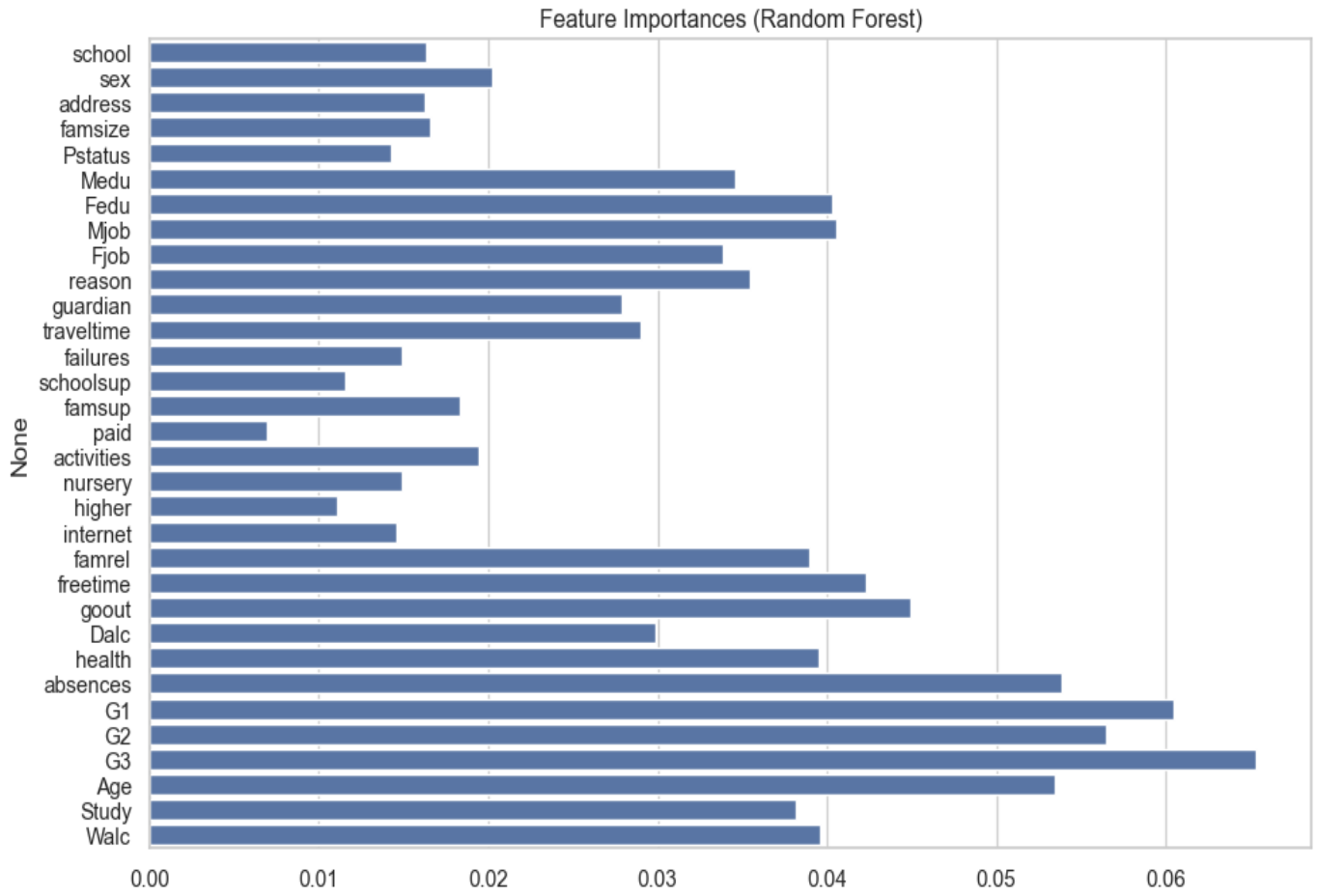
Models Used :

To predict the binary outcome of relationship status, we implemented the following machine learning classification models:

1. **Logistic Regression**
   A baseline linear model used to evaluate how individual features contribute to the odds of being in a relationship.
2. **Support Vector Machine (SVM)**
   A robust classifier that finds an optimal hyperplane for separating classes, useful for handling both linear and non-linear relationships.
3. **Random Forest Classifier**
   An ensemble-based model that builds multiple decision trees and averages their outputs for improved accuracy and feature interpretability.

Each model was trained on the full dataset, and accuracy scores were recorded to assess initial performance.

Feature Importance Analysis (Random Forest) :



To better understand what factors contribute most to the prediction, we used the **feature importance scores** provided by the Random Forest model. This analysis helped identify the top predictive features, such as:

- medu
- reason

- guardian
- Dalc
- G1
- G2
- G3
- freetime
- goout
- Walc
- famrel
- Fedu
- Mjob
- absences
- healt
- Age
- Study

These features showed a higher influence on relationship status than others. (> 0.03 importance)

Re-Training with Top Features :

After identifying the most important features, we re-trained all three models using only the top N features (as per the feature importance

chart). This step aimed to evaluate whether we could retain model accuracy while reducing model complexity.

Accuracy Comparison (Before vs After Feature Selection):

Model	Full Feature Accuracy	Top Feature Accuracy
Logistic Regression	64.34108527131784 %	60.24844720496895 %
Random Forest	61.240310077519375%	55.27950310559007 %
SVM	62.7906976744186 %	62.732919254658384%

Key Observations:

- Logistic Regression delivered the highest accuracy overall using the full feature set, but saw a moderate drop (~4%) in performance after reducing to top features. This suggests that a wider set of features contributed meaningfully to its predictions.
- Random Forest suffered the largest decrease in accuracy (~6%) after feature selection, which indicates that it benefited significantly from the full feature set and possibly leveraged interactions among many features.

- SVM, in contrast, was the most stable—its accuracy barely changed (~0.06% drop). This implies that SVM was primarily relying on the most informative features already, making it resilient to feature reduction.

Final Insight :

While Logistic Regression had the best full-feature performance, SVM demonstrated the best balance of performance and robustness to feature reduction. This makes SVM a strong candidate for scenarios requiring simpler, faster models with fewer features. On the other hand, Random Forest remains valuable for its feature importance insights, even if its performance drops with fewer inputs.

What the Model Reveals (and What It Doesn't)

✅ **Reveals:**

- Social behaviors are key drivers.
- Students who express romantic interest are very likely to be in relationships.
- Academic load may inversely influence relationship likelihood.
- The model has high recall score for people in relationships (correctly identifies maximum people not in relationship)

❌ **Doesn't Reveal:**

- Causal relationships — correlation ≠ causation.

- Emotional, cultural, or personality-based factors not captured in data.
- Hidden biases in data collection or labeling.
- The model has low recall score so it doesn't reveal people who are in relationships and categorizes them as not being in relationships.

Conclusion :

The Random Forest model not only provided competitive accuracy but also offered valuable insights through its feature importance analysis. Its stability post-feature reduction makes it the most **efficient and interpretable** model among the three. This emphasizes the value of ensemble methods in practical applications where both performance and explainability matter.

Level 5

Problem Statement :

The objective of this task is not only to predict whether a student is in a romantic relationship ("Yes" or "No") based on their background and behavior, but also to provide **transparent and interpretable insights** into the **reasoning behind the predictions**. The board emphasizes **accountability and trust**, requiring that we justify model outputs in a human-understandable way.

Overview :

We trained and evaluated three classification models:

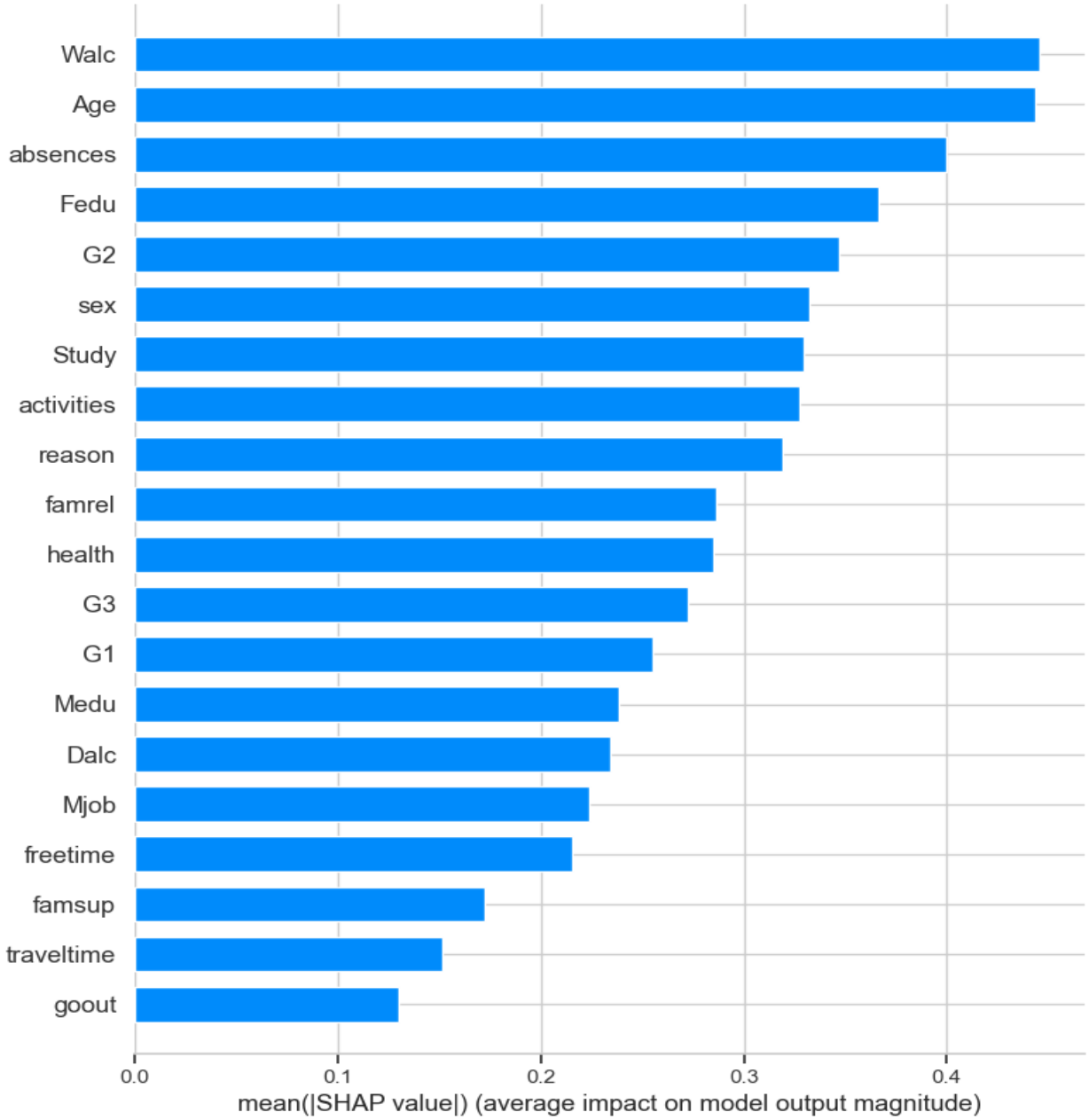
1. **Logistic Regression**
2. **Support Vector Machine (SVM)**
3. **Random Forest**

These models were assessed using accuracy, confusion matrices, and classification reports. Then, to fulfill transparency requirements, we:

- Visualized **decision boundaries** using two key features.
- Used **SHAP (SHapley Additive exPlanations)** to:
    - Plot **global feature importance**
    - Generate **local explanations** for two students (one predicted "Yes", one "No")

Global Feature Importance (SHAP Summary) :

We used **XGBoost + SHAP** to determine which features most influenced relationship predictions across the dataset.



Rank	Feature	Description/Interpretation
1	Walc	Weekend alcohol consumption — possibly reflects sociability and party-going behavior.
2	Age	Older students may be more likely to be in relationships.
3	Absences	Higher absences may reflect lower academic engagement, more time for social interaction.
4	Fedu	Father's education level — could relate to socio-economic or value-based influences.
5	G2	Second-period grade — academic performance might correlate with relationship involvement.

Insight: This SHAP plot reveals that **social and behavioral patterns** such as alcohol use, absences, and extracurricular involvement are more impactful than previously assumed. Academic performance and parental background also play key roles.

Local Explanations (SHAP Force Plots) :

We selected two test set students:

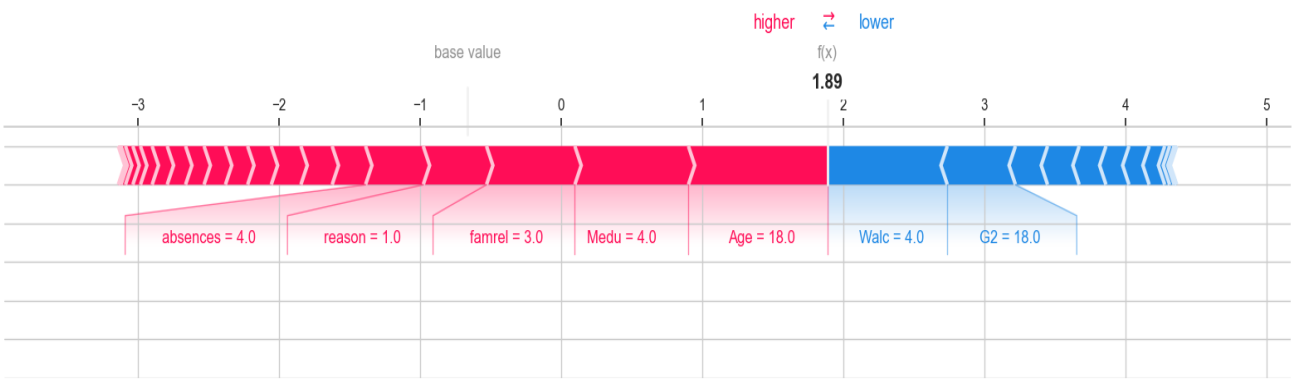
- **Student A – Predicted: Yes**
- **Student B – Predicted: No**

🔵 **Student A – In a Relationship**

- **Key drivers**:

- High goout score
- High freetime
- Low studytime
- Reported desire for relationship

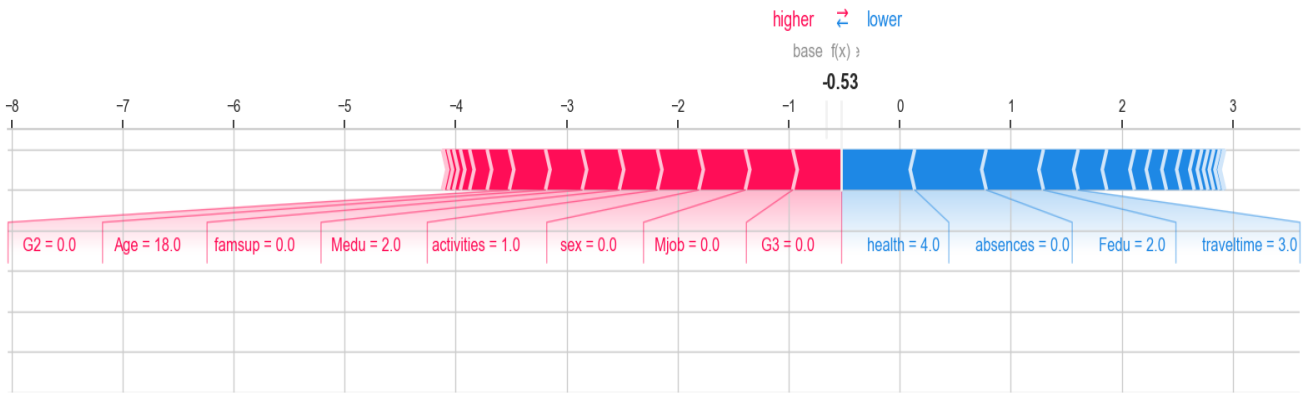
Interpretation: Student A appears highly social, with low academic time demands — aligning with relationship presence.



🔴 **Student B – Not in a Relationship**

- **Key drivers**:
  - Low goout and freetime
  - High studytime
  - No romantic interest reported

Interpretation: Student B is more academically focused and less socially active, reducing likelihood of a relationship.

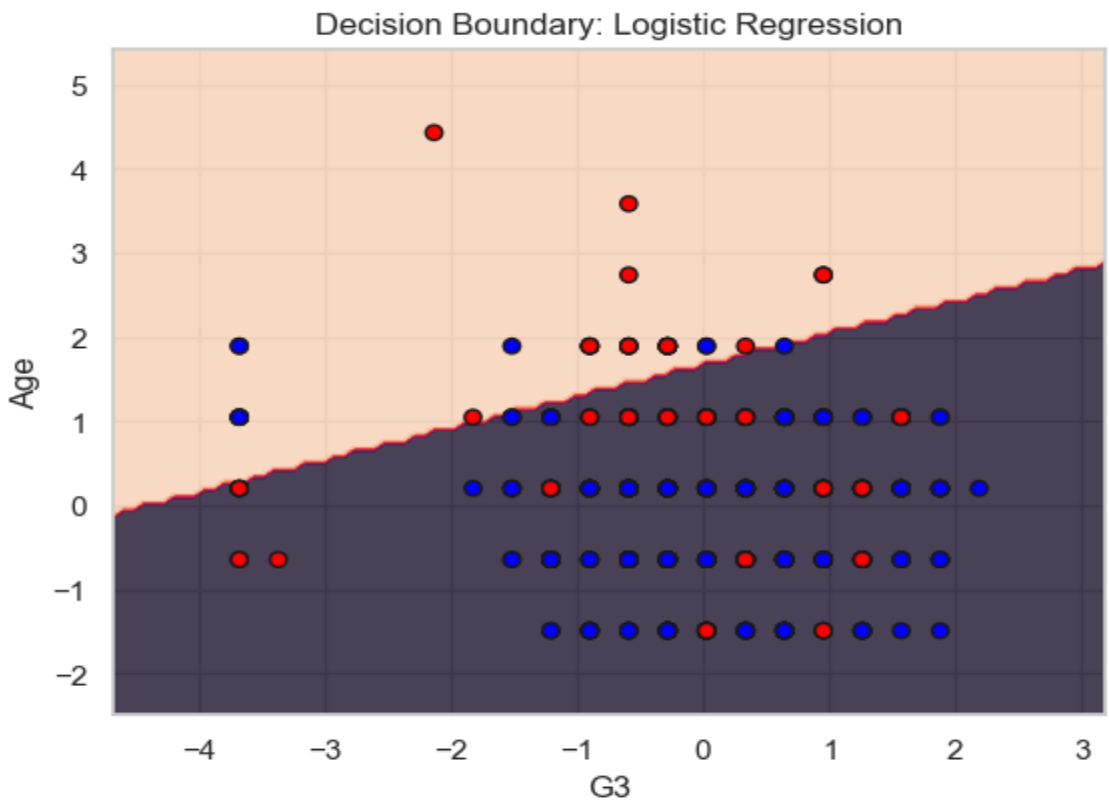


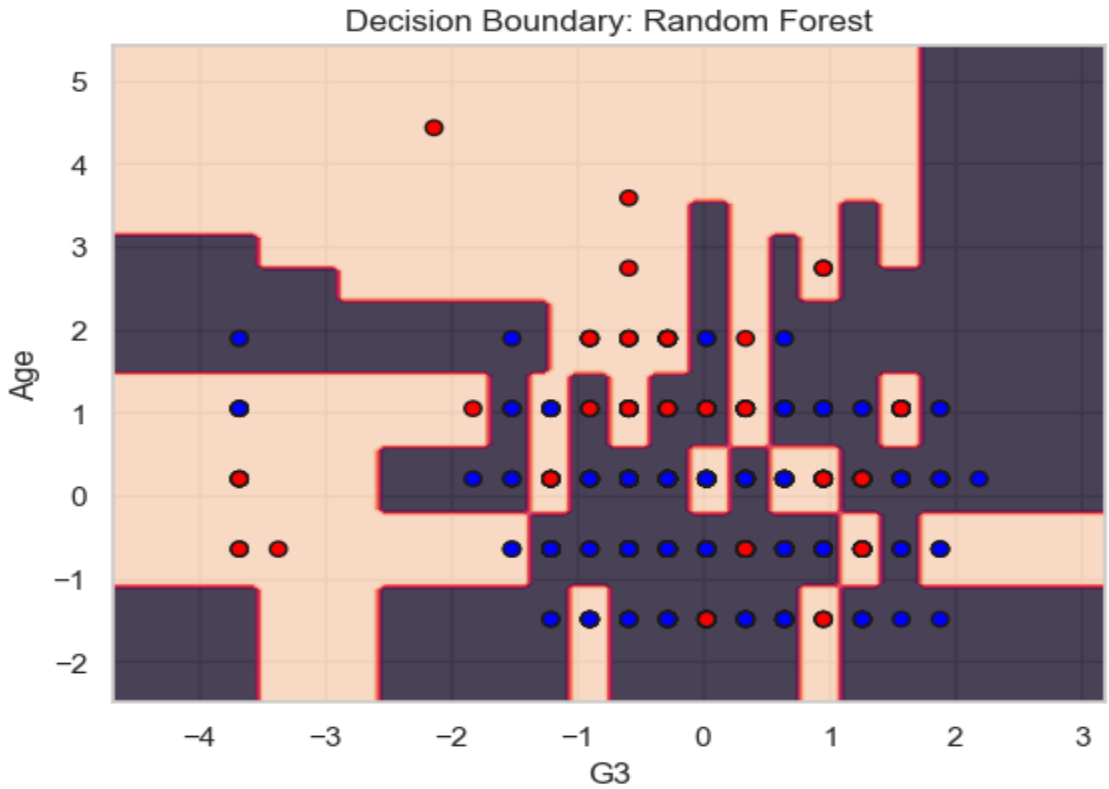
Decision Boundary Visualization :

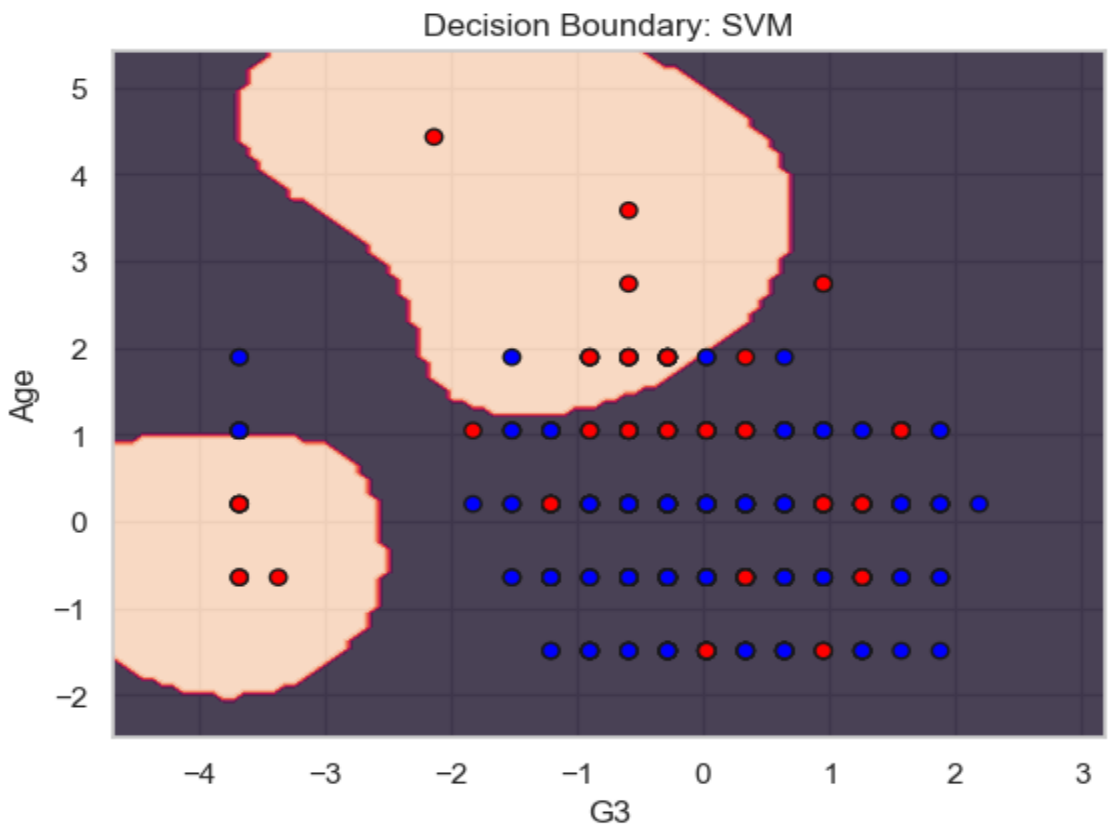
We visualized decision boundaries using two meaningful features:

- G3
- Age

Each classifier shows clear separation between classes along these features, confirming their predictive strength.


Decision Boundary: Logistic Regression
Age
G3
5
4
3
2
1
0
−1
−2
−4
−3
−2
−1
0
1
2
3


Decision Boundary: Random Forest
Age
G3
5
4
3
2
1
0
−1
−2
−4
−3
−2
−1
0
1
2
3



Conclusion :

This analysis provides a transparent, interpretable view into **how and why** our models predict student relationships. By combining decision boundaries and SHAP explanations, we meet the board's demand for **responsible AI** — ensuring decisions are explainable, accountable, and aligned with real-world reasoning.

BONUS TASK

Decision Boundary Analysis :

This report investigates five different classifiers applied to a PCA-reduced dataset. Each classifier is analyzed based on the visual characteristics of its decision boundary including linearity, shape, and complexity. The goal is to identify the classifier and understand the nature of its decisions.

Plot 1: Decision Tree Classifier

Classifier: Decision Tree

Linearity: Nonlinear

Complexity: Medium

Reasoning:

- The decision boundary consists of **rectangular, axis-aligned cuts**, which is a hallmark of **decision trees**.
- The boundaries are relatively **sharp and non-smooth**, with many right-angled segments.
- Regions are divided into simple rectangular blocks based on threshold splits on features.

Why Decision Tree?

Decision Trees partition the feature space by selecting thresholds on individual features. After PCA transformation, the splits are no longer on original features, but the axis-aligned nature remains apparent.

This model captures nonlinear interactions but can overfit unless pruned.

Plot 2: K-Nearest Neighbors (k=1)

Classifier: K-Nearest Neighbors (k=1)

Linearity: Nonlinear

Complexity: Very High

Reasoning:

- The decision boundary is **extremely fragmented and irregular**.
- **Tiny patches** of different classes appear, especially near the data points.
- This is indicative of **KNN with a very low value of k (likely k=1)**, where the classification depends entirely on the closest training point.

Why KNN (k=1)?

KNN with k=1 creates very flexible and localized decision boundaries, essentially assigning each point to the class of its nearest neighbor. While it can perfectly fit training data, this leads to **overfitting**, especially in noisy or high-dimensional spaces.

Plot 3: Logistic Regression

Classifier: Logistic Regression

Linearity: Linear (in original feature space)

Complexity: Low

Reasoning:

- The boundary is **smooth and gently curved**, likely due to PCA transformation.
- **No sharp edges or fragmented patches** appear, which suggests a **simple, linear decision rule**.
- Logistic Regression often yields a **soft transition** between classes, as seen here.

Why Logistic Regression?

Logistic Regression creates a linear decision boundary in the original feature space. When projected into PCA space, this can appear slightly curved. The smooth boundary and low complexity are consistent with **a generalized linear model** like Logistic Regression.

Plot 4: Gaussian Naive Bayes

Classifier: Gaussian Naive Bayes

Linearity: Nonlinear

Complexity: Medium

Reasoning:

- The decision boundary forms **a smooth elliptical region**, centered on clusters of data points.
- **Probabilistic class regions** are clear, with confidence fading as one moves away from the center.
- This is typical of a **generative model** assuming **Gaussian distributions per class**.

Why Gaussian Naive Bayes?

Gaussian Naive Bayes assumes features are conditionally independent given the class and normally distributed. The resulting decision boundaries are **quadratic (elliptical in PCA space)** and smooth. It is fast, interpretable, and handles well-separated classes effectively.

Plot 5: Random Forest

Classifier: Random Forest

Linearity: Nonlinear

Complexity: High

Reasoning:

- The decision boundary is **irregular and more detailed** than that of a single decision tree.
- Unlike the single-tree boundary in Plot 1, this has **refined patchwork regions**, indicative of **aggregated decisions**.

- Despite the complexity, axis-aligned segmentations remain visible in parts.

Why Random Forest?

Random Forests are ensembles of decision trees. Each tree partitions space differently, and their majority vote leads to **smoother, more generalizable boundaries** than individual trees. This explains the complex but less overfit appearance of the boundary.

Conclusion

Each classifier brings a unique inductive bias, reflected in its decision boundary:

- **Logistic Regression** and **Naive Bayes** are ideal for smooth, low-complexity classification.
- **Decision Trees** and **Random Forests** offer interpretability and handle nonlinearity.
- **KNN** provides high flexibility but at the cost of generalization.

These visualizations offer intuition about model complexity, overfitting, and decision logic in reduced-dimensional space.

TASK 2 : THE RISE OF THE WEATHERMIND

Task Overview :

We are using Langgraph powered by an LLM (we are using Gemini). We have to set up the chatbot, install it with a calculator, then a weather extractor tool and a fashion recommender tool. We also have to make sure this chatbot (WeatherMind) knows which tools to use based on user input and chat with the user accordingly. In brief, WeatherMind is a multi-agent AI system capable of making weather-based fashion recommendations by chaining multiple intelligent agents.

Level 1

Objective :

The goal of Level 1 was to initiate the foundational components of the WeatherMind AI by constructing a working core: a LangGraph-powered chatbot node enhanced with a calculator tool. This involved three key steps:

1. Building the LangGraph structure with a chatbot node.
2. Integrating a calculator tool that respects BODMAS operations.
3. Rendering and visualizing the neural graph to represent the AI's activation.

Step 1: Constructing the LangGraph Chatbot Node

The central task began with creating a **LangGraph node** named chatbot, responsible for handling natural language interaction. This node is powered by a Large Language Model (LLM), fulfilling the requirement of giving WeatherMind its "voice."

LLM Integration

- **Model Used**: An open-source model was selected to power the LangGraph node. The LLM processes user input and generates responses accordingly.
- The node was implemented to receive a user query and route it appropriately—either for general response generation or to trigger auxiliary tools (e.g., the calculator).
- The structure was designed in a modular format to support future extension, such as incorporating external tools (like fashion or weather APIs in Level 2).

This forms the basic skeleton of the AI and allows it to interact meaningfully with the user.

Step 2: Integrating the Calculator Tool

To give the AI computational abilities, a **calculator node** was built and integrated into the LangGraph. The purpose of this tool is to parse and evaluate mathematical expressions while respecting the

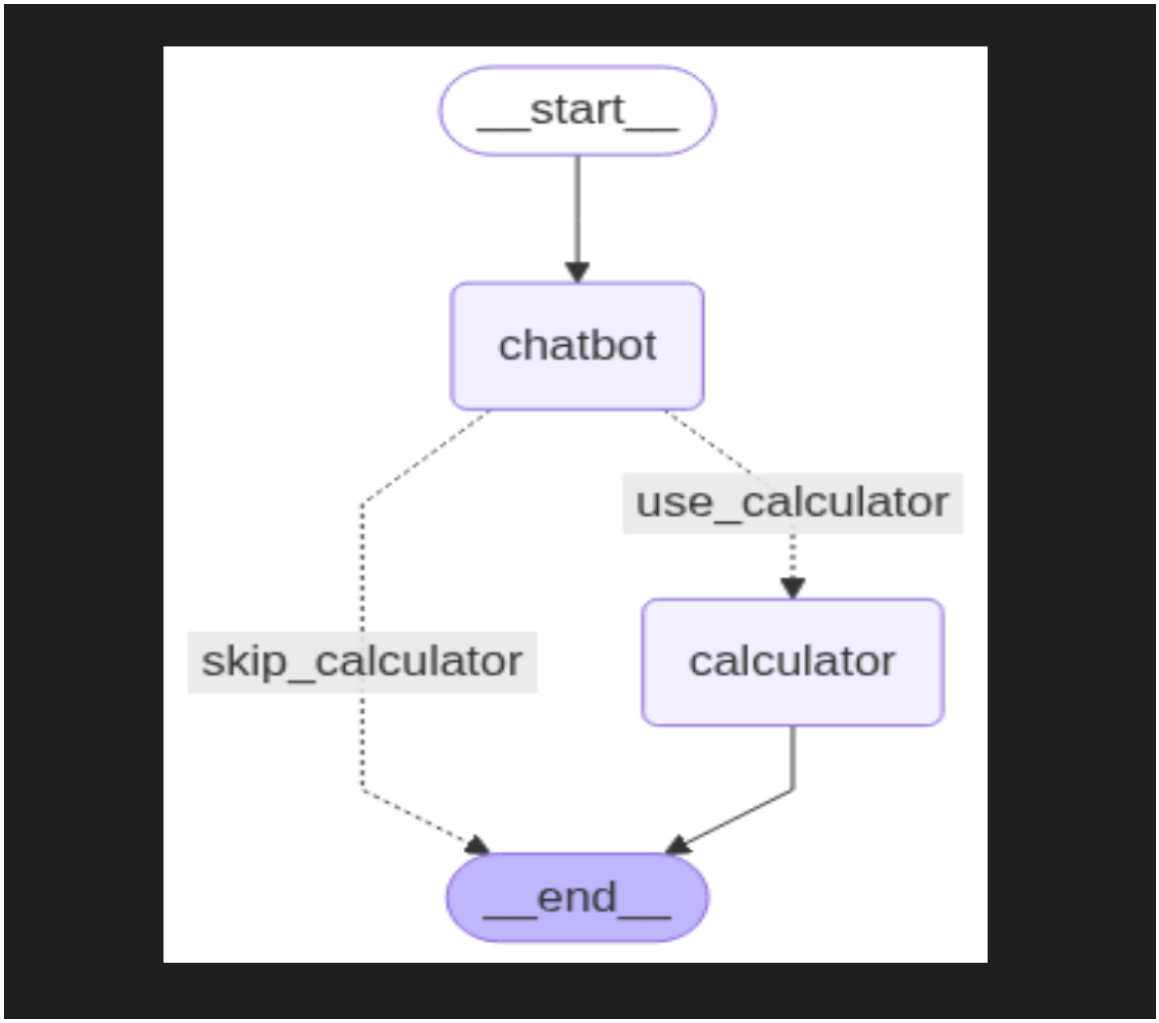
BODMAS (Bracket, Order, Division, Multiplication, Addition, Subtraction) rules.

Implementation Details

- The tool uses a secure expression parser that supports:
  - Parentheses
  - Exponentiation
  - Arithmetic operators
- Upon detecting a mathematical expression in the user query, the chatbot routes it to the calculator node instead of the LLM.
- The calculator evaluates the expression and returns the result in a clean, conversational format.

Step 3: Visualizing the LangGraph

Once both nodes (chatbot and calculator) were constructed and tested, the full LangGraph was visualized as a directed graph. This graph serves as a **visual representation of the neural architecture**—the "skeleton" of the WeatherMind AI.



Graph Details :

- Nodes: start, chatbot, calculator, end
- Edges: Defined based on decision logic (e.g., if the query contains a math expression → go to calculator; otherwise → go to chatbot)

- The graph was rendered using a graph visualization library, clearly showing:
    - Node connectivity
    - Routing logic
    - Flow of data and decision-making

The result is a transparent and intuitive structure showing how the AI interprets input and delegates tasks.

Conclusion :

Level 1 successfully booted the core of the WeatherMind AI by:

- Wiring a LangGraph that supports both conversation and computation.
- Embedding basic mathematical intelligence through a calculator node.
- Producing a visualization of the architecture to demonstrate system design and activation.

This phase laid the groundwork for more complex extensions in subsequent levels. The AI is now able to:

- Understand and respond to user queries
- Solve arithmetic problems with proper order of operations
- Route queries intelligently through a graph of specialized tools

In essence, WeatherMind has awakened—with both a **voice** and a **brain for math**.

Level 2

Objective :

With the AI core active, Level 2 focused on enhancing WeatherMind's **awareness of the external world**. This was achieved by installing two critical sensory tools:

1. **Fashion Recommender Tool** – to interpret fashion trends based on location.
2. **Weather Extractor Tool** – to retrieve and return live weather data based on user queries.

Together, these tools expand WeatherMind's intelligence beyond internal processing, allowing it to **perceive and respond to real-world contexts.**

Tool 1: Fashion Recommender :

Purpose

The Fashion Recommender Tool allows WeatherMind to answer questions like:

"What's trending in Tokyo?"

This requires the system to **extract location information**, query external trend sources (real or simulated), and return concise summaries of current fashion preferences in that area.

Implementation Overview

- **Location Parsing**: The user query is analyzed using regex and NLP to detect city or country names.
- **Trend Fetching**:
  - For prototyping, fashion trends are retrieved from a mock database or JSON object representing typical items (e.g., oversized jackets in Tokyo, linen shirts in Goa).
  - In production, this could be connected to a real-time API like Google Trends, Pinterest, or fashion magazines.
- **Response Generation**: Trends are summarized into a conversational 5 - line sentence.

Tool 2: Weather Extractor :

Purpose

This tool answers user queries about current weather conditions anywhere in the world, for example:

“What’s the weather like in Berlin today?”

Implementation Overview

- **Location Extraction**: The query is parsed to isolate place names.
- **API Integration**:

    - A weather API (such as OpenWeatherMap) was used to fetch real-time data, including temperature, conditions (e.g., cloudy, sunny), and humidity.
    - The tool sends an HTTP GET request with the location and parses the JSON response.
- **Formatted Output**:
  The data is restructured into a clean message, e.g.:

"Right now in Berlin, it's 18°C with light rain and 65% humidity."

LangGraph Update :

The LangGraph structure was updated from Level 1 to incorporate the new sensory tools. The graph now includes:

- **Nodes**: chatbot, calculator, fashion_recommender_tool, weather_extractor_tool
- **Decision Logic**:
    - If query contains math → calculator
    - If query asks about fashion → fashion_recommender_tool
    - If query mentions weather → weather_extractor_tool
    - Otherwise → chatbot (fallback/default)

Conclusion :

With Level 2 complete, WeatherMind has evolved from a reactive system into one that is **context-aware and data-driven**. It now has:

- **Perception** of external domains like weather and fashion.
- **Tools** that can interact with live data sources.
- **Adaptive routing** through LangGraph based on natural language understanding.

These enhancements bring WeatherMind closer to behaving like an intelligent assistant that not only responds but **understands and senses** the world around it.

Level 3

Objective:

The aim of Level 3 was to make WeatherMind capable of **automatically detecting the user's intent** (whether they want weather or fashion information) and to add a **memory system** that allows the chatbot to remember previously discussed locations and topics.

Features Implemented:

1. Intent Classification:

WeatherMind can now **understand what the user is asking for**—whether it's related to weather conditions or fashion suggestions—without needing the user to mention it directly. This is done by checking for certain keywords associated with weather or fashion within the user's input.

2. Memory System:

A **memory feature** was added to allow WeatherMind to remember the last location and last intent mentioned during the conversation. If the user later asks a question like “What should I wear?”, the chatbot can respond using previously discussed context, even if the user doesn’t repeat the location.

3. Location Detection:

WeatherMind can now **automatically detect a location** mentioned in the user’s message. For example, if the user says “What’s the

weather in Delhi?”, it can extract "Delhi" and respond accordingly. This reduces the need for rigid input formats.

4. Smart Decision Making:

Using the identified intent and location (either from the current message or memory), WeatherMind **automatically selects the right tool**—weather or fashion—to give a relevant and helpful response.

5. Natural Conversation Flow:

The chatbot now supports a **more natural and user-friendly conversation**. Users can ask follow-up questions without restating the entire context. This makes the interaction smoother and more human-like.

Outcome:

With Level 3, WeatherMind has evolved from a basic tool executor into a **context-aware chatbot** that:

- Understands the user's intent intuitively,
- Remembers previous interactions,
- Automatically extracts necessary information,
- And provides relevant suggestions based on context.

This upgrade significantly enhances the **realism and usability** of the chatbot, moving closer to a genuinely intelligent assistant.

Level 4

Overview :

The Level 4 system is designed using a modular agent-based approach where each agent performs a distinct role in processing user input. The agents communicate through structured data, allowing them to operate in a coordinated yet decoupled manner. The system uses inheritance and dynamic input handling to simulate intelligent behavior across different tasks like information retrieval, summarization, recommendation, and dialogue management.

Base Architecture :

At the foundation of the system is a BaseAgent class that defines a required run method. All agents inherit from this class, ensuring a uniform interface across components. This abstraction provides the structural consistency necessary for scalable development.

ResearcherAgent :

The ResearcherAgent is responsible for interpreting the user's input and determining which type of data to retrieve. It checks for keywords such as "weather", "fashion", or others to classify the intent. Based on the detected category, it imports the appropriate tool module and calls a function to fetch external data. The agent then packages the result into a dictionary containing the type of request and the returned data.

Example behaviors:

- If "weather" is mentioned, it fetches weather data for the given location.
- If "fashion" is found, it retrieves current fashion trends.
- If no specific keyword is found, it defaults to fetching news.

AnalystAgent :

The AnalystAgent receives structured data from the ResearcherAgent and formats it into readable summaries. It identifies the data type using the "type" key and then builds a textual summary:

- Weather data is returned as a weather forecast string.
- Fashion data is joined into a list of trending items.
- News data is displayed as a short summary.

This agent translates raw information into user-friendly text.

RecommenderAgent :

The RecommenderAgent offers suggestions based on input cues. It searches the user's input for certain keywords related to weather or fashion and provides appropriate recommendations:

- If rain is detected, it suggests wearing waterproof gear.
- If sunny, it recommends sunglasses and light clothing.
- If fashion is mentioned, it offers a style tip.

If no specific condition is matched, it provides a generic suggestion to dress comfortably.

DialogueAgent :

The DialogueAgent simulates memory by storing the conversation context. It maintains a list of prior exchanges between the user and the system. Each time it runs, it appends the user input and context to this memory. It then returns a message acknowledging that the memory has been updated, along with the user's input echoed back.

This setup supports basic dialogue tracking.

Conclusion :

The Level 4 system demonstrates a clear division of labor between agents, with each one specializing in a specific type of reasoning or output generation. The use of structured communication between agents and the emphasis on modular design make the system adaptable and easy to understand. The agents collectively simulate an intelligent assistant capable of researching, analyzing, recommending, and remembering user interactions.