### ****Task 1: Machine Learning Model – KNN for Wavelength Classification****

#### **Approach:**

For this task, I developed a machine learning model using the K-Nearest Neighbors (KNN) algorithm to classify wavelengths based on their features into two categories: **Hadron** and **Gamma**. The dataset includes multiple features related to the properties of each wavelength.

I performed the following steps:

1. **Data Preprocessing:** Loaded and cleaned the dataset using **NumPy** and **Pandas**.
2. **Visualization:** Explored the data distribution and relationships using **Matplotlib**.
3. **Model Building:** Implemented the KNN classifier from **scikit-learn**.
4. **Training & Testing:** Split the data, trained the model, and evaluated its accuracy.

#### **Tools & Libraries:**

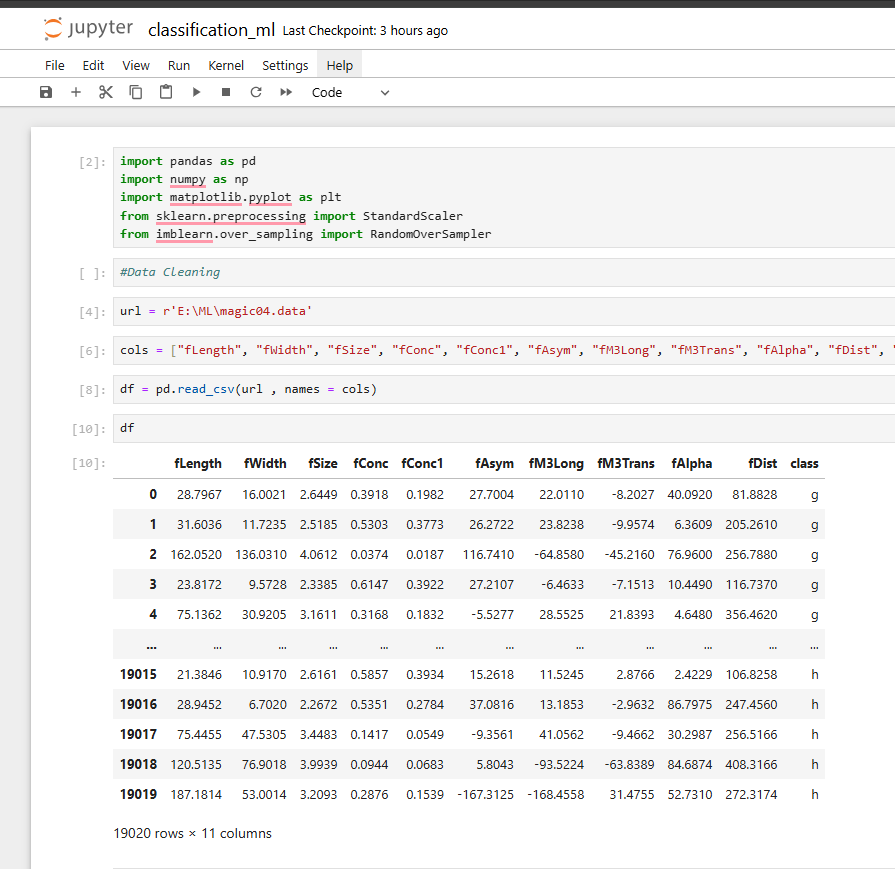
* **Jupyter Notebook** – for interactive development and visualization.
* **Libraries:**
  + numpy – for numerical operations
  + pandas – for data handling
  + matplotlib.pyplot – for visualization
  + sklearn – for building and evaluating the KNN model

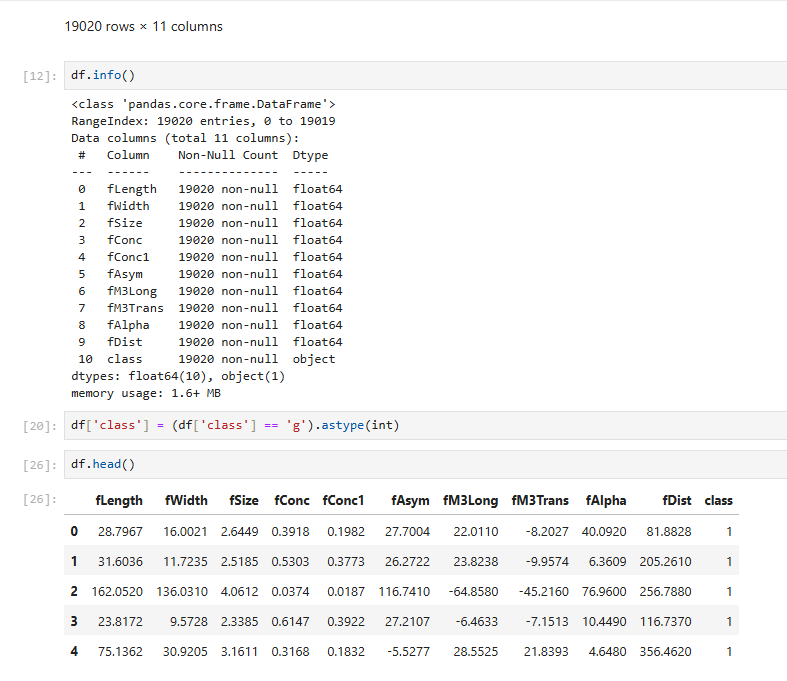
#### **Functionality Demonstration:**

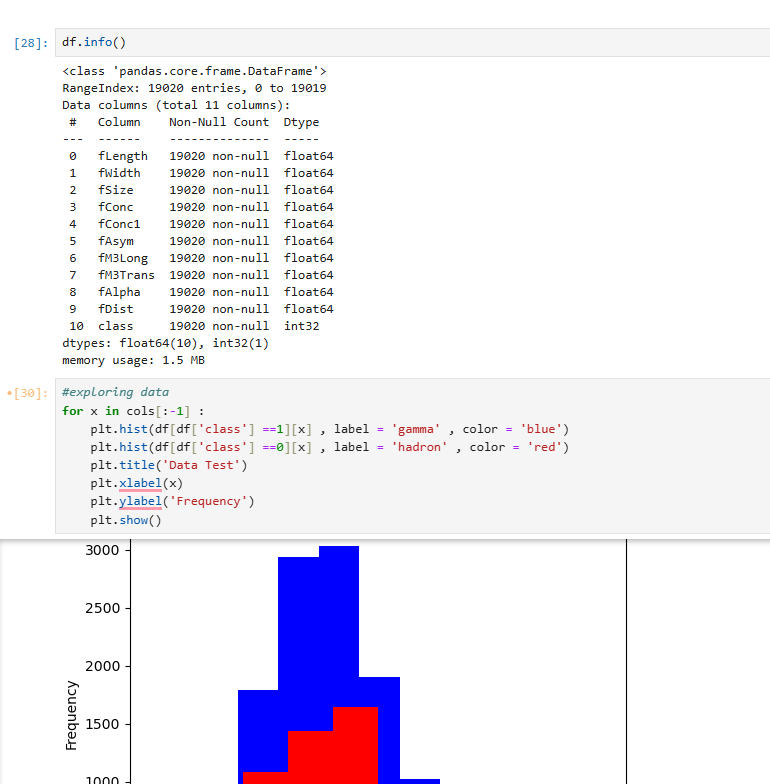
Include a screenshot here showing your code, dataset preview, and model accuracy/output.

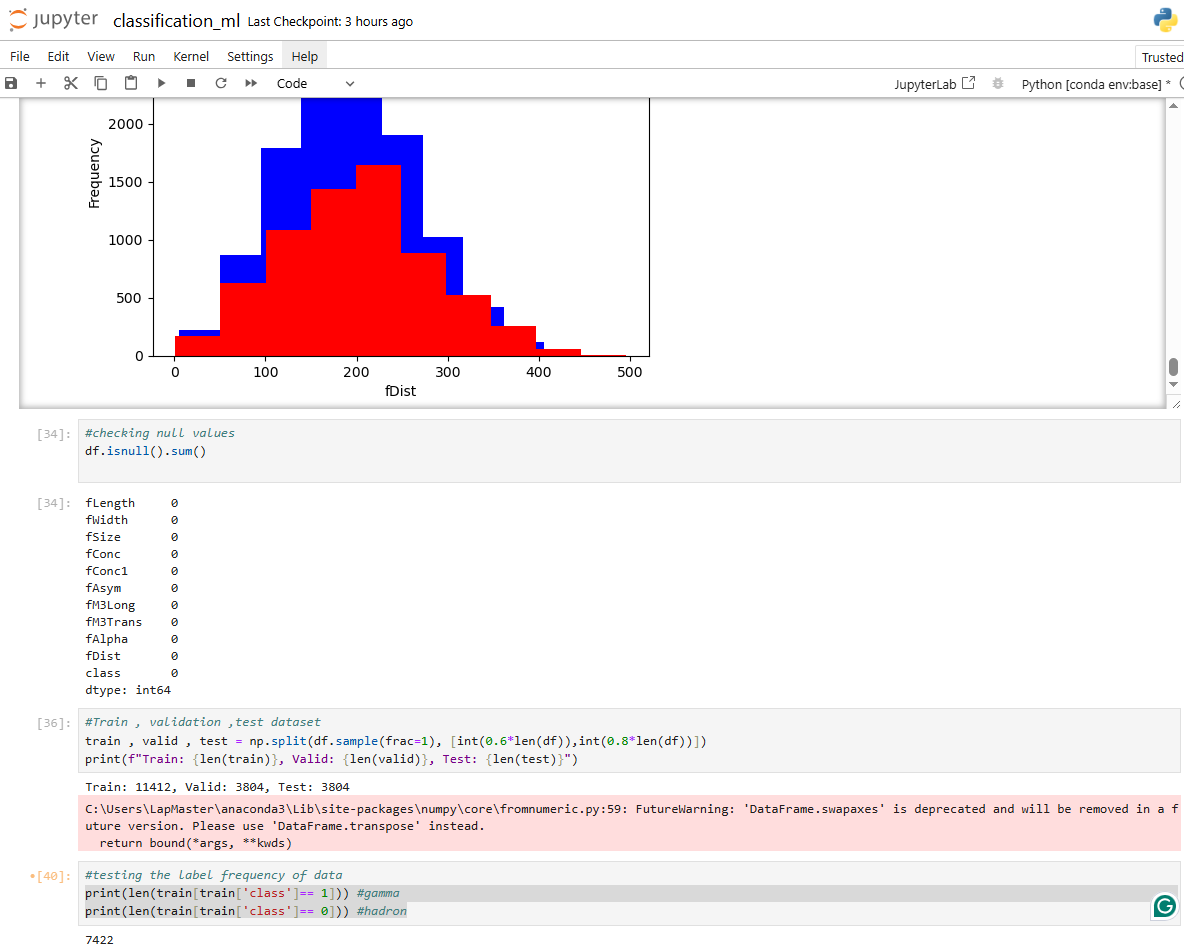
#### **Challenges & Solutions:**

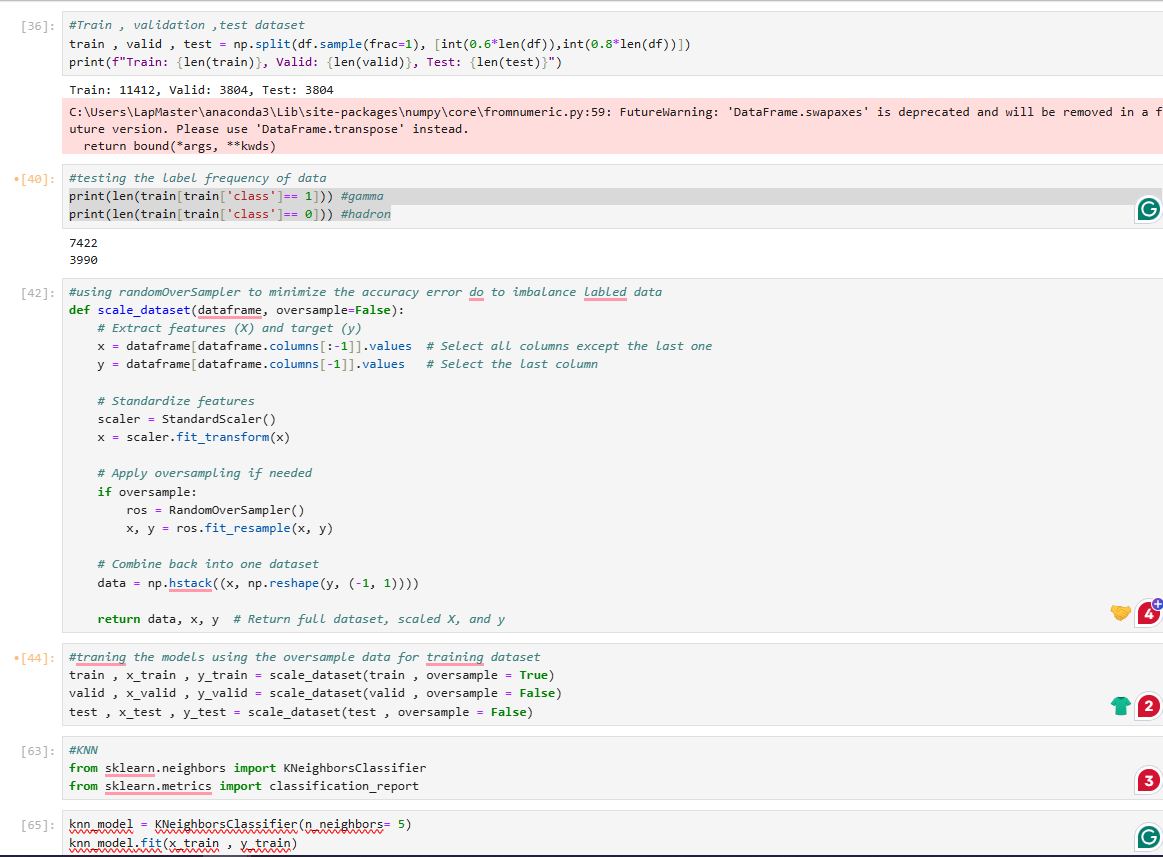
* **Data Imbalance:** Initially faced minor imbalance in class distribution. I handled it by evaluating model performance using metrics beyond accuracy, like the confusion matrix.
* **Choosing Optimal K:** Had to experiment with different values of K for best results. Used cross-validation to identify the most suitable value.





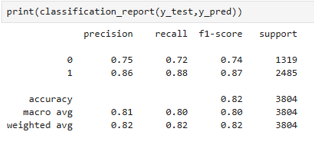




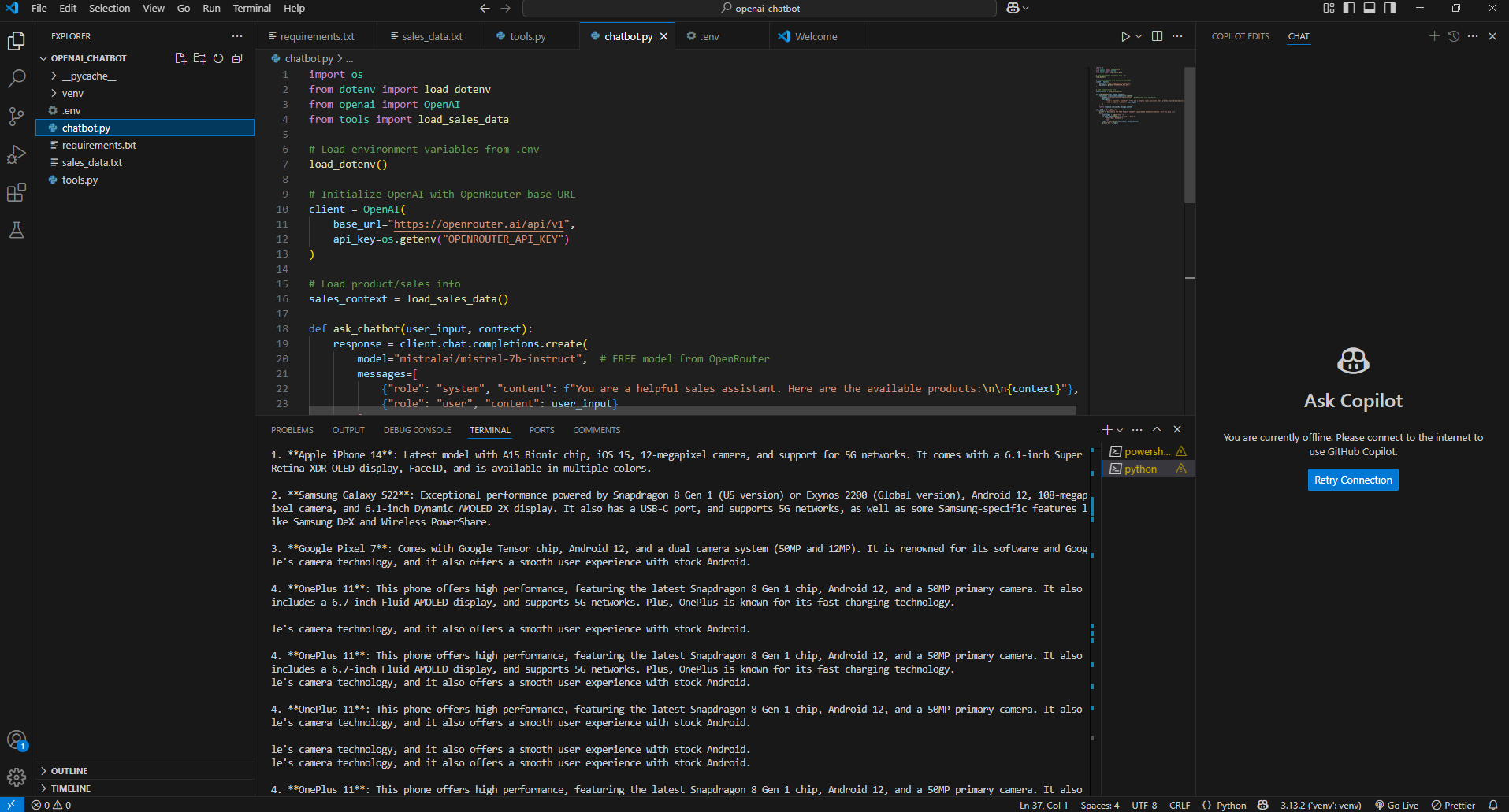




Classification Report



**Task 2: Building a Product Chatbot**



**Approach**

1. **Environment Setup**:
   * Created a Python virtual environment and installed dependencies using requirements.txt.
   * Used the python-dotenv package to securely store the OpenRouter API key in a .env file.
2. **OpenRouter Integration**:
   * Switched from OpenAI's default API (which requires a paid plan) to OpenRouter, which provides access to free models like mistralai/mistral-7b-instruct.
   * Configured the OpenAI client with OpenRouter's base\_url.
3. **Sales Context Loading**:
   * Created a sales\_info.txt file with sample product data.
   * Loaded this data into the chatbot using a helper module (tools.py) for context injection.
4. **Chatbot Logic**:
   * Developed the main interaction in chatbot.py using OpenAI’s modern SDK structure.
   * Used chat completions with system/user roles to instruct the chatbot how to respond.
5. **Terminal Chat Interface**:
   * Implemented a simple terminal loop that accepts user input and returns product-related responses from the chatbot.

**Challenges & Solutions**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Challenge** | **Solution** | | OpenAI API returned a RateLimitError due to zero quota. | Replaced it with [OpenRouter](https://openrouter.ai/) which supports free-tier models. | | openai.ChatCompletion.create caused a APIRemovedInV1 error. | Migrated to the new OpenAI() client approach (openai>=1.0.0) as per the updated SDK documentation. | | Needed secure handling of API keys. | Used .env file and python-dotenv to load API key safely. | | Model not recognizing product data initially. | Passed product list as part of the system prompt (sales\_context). | |

### ****4 Task: Simple Webpage with Navbar for an E-Commerce Platform****

#### **Approach:**

For the final task, I created a **simple yet modern webpage** with a navigation bar, designed as part of an **e-commerce platform**. I reused the same frontend design from my **2nd-year university final project**, which featured both frontend and backend components.

The goal for this task was to demonstrate the frontend portion only.

#### **Technologies Used:**

* **HTML** – for page structure.
* **CSS (via Bootstrap)** – for responsive styling.
* **JavaScript** – for interactivity and basic frontend logic.
* **Bootstrap Framework** – to ensure mobile-first design and professional UI layout.

#### **Functionality Demonstration:**

* A responsive **navbar** with links like **Home**, **Products**, **Cart**, and **Contact**.
* A hero section or banner showcasing promotional items.
* Sample product grid cards styled with Bootstrap.
* The design is optimized for desktop and mobile views.

Include a screenshot here showing the full webpage with the navbar and sample product section.

#### **Challenges & Solutions:**

* **Bootstrap Integration:** Initially faced conflicts between custom CSS and Bootstrap classes. Solved this by carefully overriding Bootstrap styles without breaking layout responsiveness.
* **Code Reusability:** Adapting a larger project to fit this simpler task required restructuring some parts. I modularized code sections for clarity and easier reuse.

