**Project 2ed semester**

process

**tep 1: Access and Permissions**

* **API Access**: Check if our API access includes capabilities for fine-tuning. This might require a specific tier of service or enterprise access, as not all API tiers include fine-tuning capabilities.
* **Documentation and Guidelines**: Review OpenAI's documentation thoroughly for the latest on how to use their fine-tuning service. This documentation will include technical details, limitations, and pricing.

**Step 2: Prepare Our Dataset**

* **Data Format**: Prepare our data according to the specifications required by OpenAI. This usually means formatting our data in JSONL format (JSON Lines format), where each line is a separate JSON object representing a prompt and its corresponding completion.
* **Quality and Quantity**: The quality and quantity of our data significantly impact the performance of our fine-tuned model. Ensure that the data is clean, well-labeled, and representative of the tasks we want the model to perform.

**Step 3: Fine-Tune the Model**

* **Configuration**: Configure our fine-tuning parameters, the number of training epochs, and other hyperparameters. OpenAI’s API documentation and user interface will guide these settings.
* **Start Fine-Tuning**: Initiate the fine-tuning process through the API. This process will train a new model based on our dataset and the base model we selected.

**Step 4: Test and Deploy**

* **Testing**: After the fine-tuning process is complete, we'll receive access to our fine-tuned model. Test the model thoroughly to evaluate its performance on tasks similar to those it was trained on.

**Step 5: Monitor and Iterate**

* **Performance Monitoring**: Continuously monitor the model’s performance and gather feedback. If the model drifts or its performance decreases, we might need to retrain or further fine-tune it.

**Iterative Improvement**: Based on feedback and performance, we might find that additional data or adjustments in the fine-tuning process are necessary.  
  
  
**Handling Responses**: Store the responses from the API. we might want to save them in a database or a file depending on our end use.

**Analysis**: Once we have the responses, we can analyze the results as per our requirements.

**then we will have the system diagnosis and the doctor diagnosis   
  
Classification Accuracy**: The percentage of predictions that are correct. It’s calculated as:

A black text on a white background

Description automatically generated

import pandas as pd

import openai

# Load your dataset

df = pd.read\_csv('db')

# Set your OpenAI API key

openai.api\_key = '-api-key'

def query\_chatgpt(text):

response = openai.Completion.create(

engine="text-davinci-003",

prompt=text,

max\_tokens=150

)

return response.choices[0].text.strip()

# Process each item in the dataset

results = []

for index, row in df.iterrows():

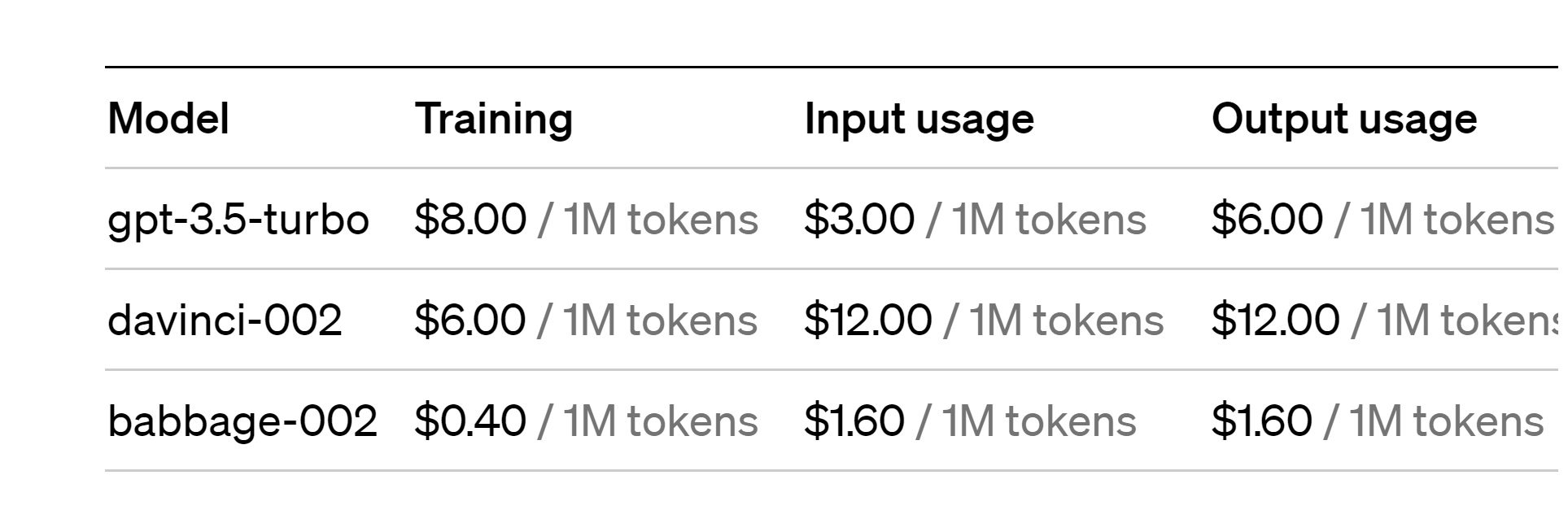
response\_text = query\_chatgpt(row['text'])

results.append(response\_text)

# Save results

df['response'] = results

df.to\_csv('processed\_dataset.csv', index=False)

- how much does it cost?  
  
NOW!!!!

- working on our data .  
- **Error Handling** We need manage cases where the API might return errors?

Second results  
1- I have llama2 throw youtube to fine tune it   
2- I have git hub to find some codes that can help me   
3- mybe I can find some video that fine tune chat gpt   
|4- my plane first is to fine tune llama2 on my data(write thw code that I will use it any where ) &&and use the chat gpt api on colab using the previous code  
  
………………………………………………………..  
gpt key is okey   
I m preparing the data >>>here q1.  
llama2