



Project Title:

AI-Powered Personalized Study Planner

Group Members:

- Rehan Asif - 70147344
- Khudija Kashif - 70147016

Submitted to: DR. HAMEDOON

Introduction:

The **AI-Powered Personalized Study Planner** is a Jupyter Notebook application designed to help students organize and optimize their study schedules using data-driven predictions and automated planning. The system trains a **Linear Regression model** on historical or synthetic data—including subject difficulty, target marks, and hours studied—to estimate how much time is required for each subject.

Users interact with the planner through an interface built using **ipywidgets**, where they can add subjects, set goal marks, select difficulty levels, and define their available study hours for each day of the week. Once the inputs are provided, the application breaks the predicted study time into **one-hour blocks** and distributes them across a **14-day schedule** based on the user's availability.

To support real-world usage, the planner includes **Google Calendar integration**, allowing users to export the generated study schedule directly into their calendar as individual events.

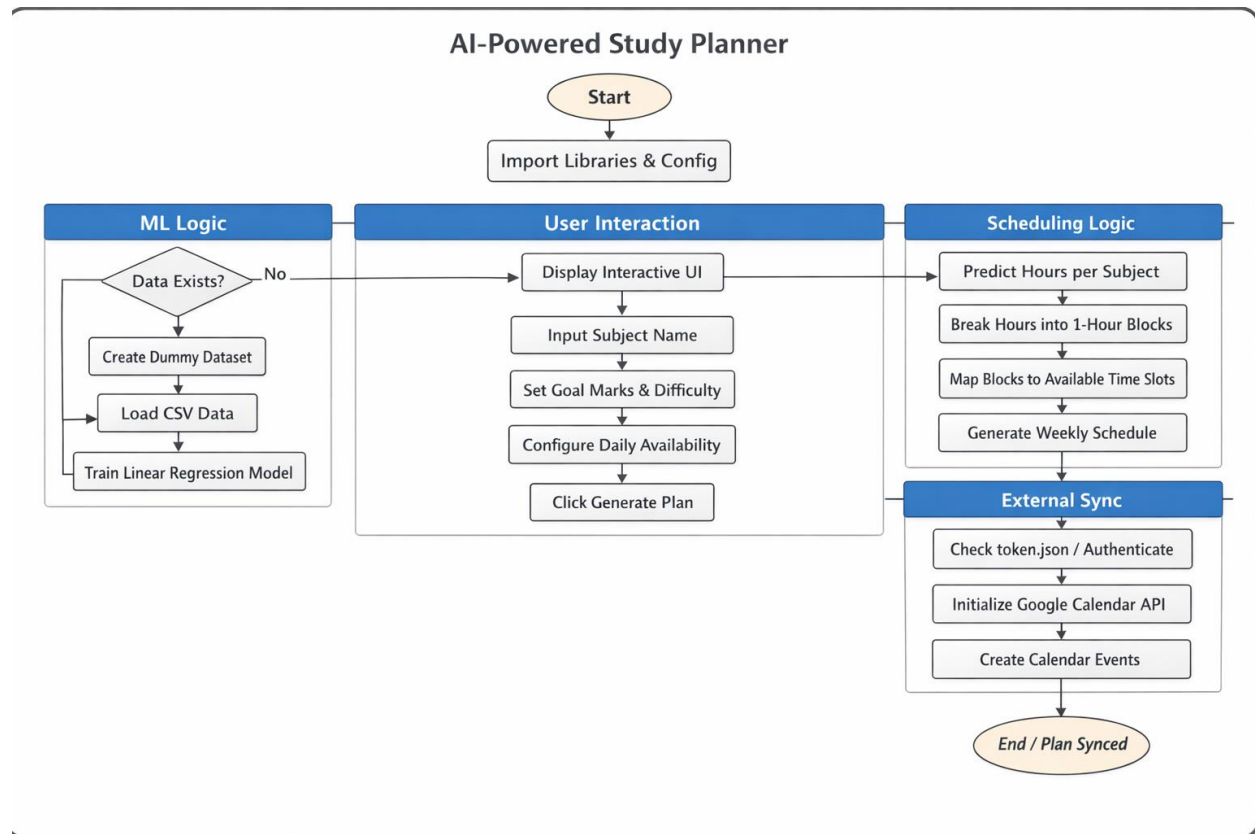
Technologies Used

The project is built using a Python-based data science stack. **Scikit-learn** is used to implement the Linear Regression model, while **Pandas** and **NumPy** handle data processing and synthetic dataset generation. The interactive interface is created using **ipywidgets**, enabling sliders, buttons, and input fields within the notebook.

For calendar synchronization, the project uses the **Google APIs Client Library** along with **Google Auth** to manage OAuth2 authentication and event creation. Scheduling logic is handled using Python's **datetime** module, and **Matplotlib** and **Seaborn** are included for optional data visualization.

- **AI-Based Study Time Prediction**
Estimates required study hours based on subject difficulty and performance goals.
- **Personalized Weekly Scheduling**
Allows users to define daily study start times and available hours.
- **Interactive Notebook Dashboard**
Enables subject management and schedule generation without manual coding.
- **Google Calendar Synchronization**
Automatically exports study sessions as calendar events.
- **Automatic Data Generation**
Creates dummy training data when no prior dataset is available.

Flow Diagram:



Project Code:

```
# 1. Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import logging as logging
from IPython.display import display, clear_output
import datetime

# Create Logger Objects
from google.auth.transport.requests import Request
from google.auth.credentials import Credentials
from google.auth.oauth2 import Flow, InstalledAppFlow
from googleapiclient.discovery import build

# Configuration
DATA_PATH = "datastudy_data.csv"
SCOPES = ["https://www.googleapis.com/auth/calendar"]

print("Libraries Imported Successfully!")
```

✓ 112s Libraries Imported Successfully!

2. Train the Model

```
def train_model():
    try:
        if not os.path.exists(DATA_PATH):
            # Create dummy data if missing
            print("Data file not found. Creating dummy data...")
            os.makedirs('data', exist_ok=True)
            df_dummy = pd.DataFrame({
                'Attendance_Level': np.random.randint(1, 5, 50),
                'Hours_Required': np.random.randint(10, 50),
                'Hours_Studied': np.random.randint(1, 10, 50)
            })
            df_dummy.to_csv(DATA_PATH, index=False)

        df = pd.read_csv(DATA_PATH)
        x = df[['Attendance_Level', 'Hours_Required']]
        y = df['Hours_Studied']

        model = LinearRegression()
        model.fit(x, y)

        print("Model Trained Successfully!")
        return model
    except Exception as e:
        print(f"Error Training model: {e}")
        return None

model = train_model()
```

3. Google Calendar Authentication Logic

```
def authenticate_google_calendar():
    creds = None
    # The file token.json stores the user's access and refresh tokens.
    if os.path.exists('token.json'):
        creds = Credentials.from_authorized_user_file('token.json', SCOPES)
    except ValueError:
        print('token.json' found. Will re-authenticate.')
        try:
            os.remove('token.json')
        except OSError:
            pass
        creds = None

    # If there are no (valid) credentials available, let the user log in.
    if not creds or not creds.valid:
        if creds and creds.expired and creds.refresh_token:
            try:
                creds.refresh(Request())
            except Exception:
                print('token expired and refresh failed. Please re-authenticate.')
                creds = None
        else:
            if not os.path.exists('credentials.json'):
                print('token: ' 'credentials.json' not found. Please place it in the project root.')
                return None
            try:
                flow = InstalledAppFlow.from_client_secrets_file(
                    'credentials.json', SCOPES)
                # Run local server for auth
                creds = flow.run_local_server(port=80)
            except Exception as e:
                print('Authentication error: (e)')
                return None

    # Save the credentials for the next run
    with open('token.json', 'w') as token:
        token.write(creds.to_json())

    return build('calendar', 'v3', credentials=creds)

def create_calendar_event(service, summary, start_time, end_time):
    event = {
        'summary': summary,
        'start': {
            'dateTime': start_time.isoformat(),
            'timeZone': 'UTC',
        },
        'end': {
            'dateTime': end_time.isoformat(),
            'timeZone': 'UTC',
        },
    }
    event = service.events().insert(calendarId='primary', body=event).execute()
    print('Event created: (event.get("htmlLink"))')
```

10

Python

4. Interactive Planner Application

```
# Main
subjects_list = []
generated_schedule = []

# --- WIDGETS ---

# 1. Subject Description
in_subject = widgets.Text(description='Subject', placeholder='e.g. AI')
title_marks = widgets.Text(description='Title', placeholder='e.g. Mark 1')
toggle_difficulty = widgets.ToggleButtons(
    options=['easy', 'h'], ('easy', 1), ('hard', 2)),
description_difficulty =
)

# 2. Weekly Schedule Settings (per day)
days_of_week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
day_widgets = []

# Create widgets for each day
schedule_row = []
for day in days_of_week:
    # Start time (in hours)
    start_drop = widgets.Dropdown(
        options=[(f'{h}:00', h) for h in range(1, 24)] + [(f'{h}:00', h) for h in range(4, 7)],
        value='',
        description=f'{day} Start',
        style={'description_width': 'initial'}
    )
    # Duration slider
    duration_slider = widgets.Slider(
        value=0, min=0, max=1,
        description='Hours',
        style={'description_width': 'initial'})
    day_widgets[day] = [start_drop, duration_slider]

schedule_row.append(widgets.Button('start_drop', duration_slider))
accordion_schedule = widgetsAccordion(children=[widgets.Widget(schedule_row)])
accordion_schedule.set_title(0, 'Configure Weekly Availability (Click to Expand)')

# 3. Actions
btn_add = widgets.Button(description='Add Subject', button_style='info')
btn_remove = widgets.Button(description='Remove Item', button_style='danger')
btn_save = widgets.Button(description='Save to Calendar', button_style='success')
btn_clear = widgets.Button(description='Clear Subject', button_style='warning')
output_area = widgets.Output()

# --- UI ---

def refresh_ui(message):
    with output_area:
        clear_output()
        if message:
            print(f'{message}')
            print("-" * 30)

    # Show subjects
    print(f'Current Subjects: ({len(subjects_list)})')
    for i, s in enumerate(subjects_list):
        print(f'{i+1}. {s["name"]} | Day: {s["days"]} | Diff: {s["difficulty"]}')

    # Show schedule of events
    if generated_schedule:
```

Output panel:

AI Study Planner

Subject:

Add Subject

Clear Subjects

Goal Marks:

Difficulty: Easy Medium Hard

Available Study Hours

Configure Weekly Availability (Click to Expand)

Monday Start:

Hours:

Tuesday Start:

Hours:

Wednesday Start:

Hours:

Thursday Start:

Hours:

Friday Start:

Hours:

Saturday Start:

Hours:

Sunday Start:

Hours:

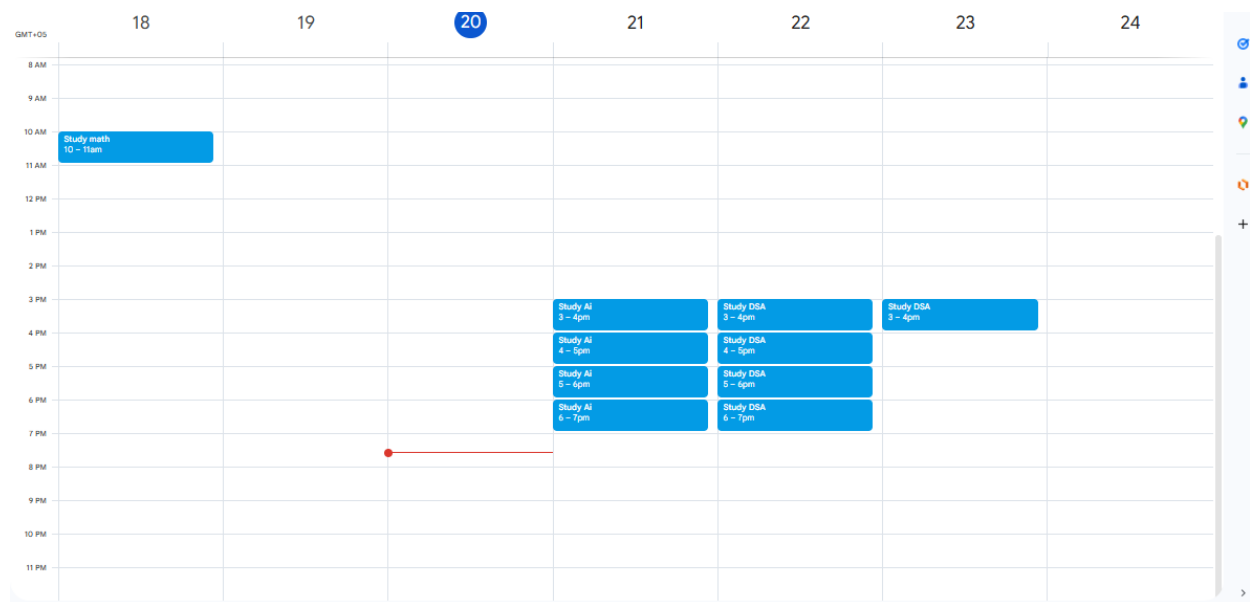
Generate Plan

Sync to Calendar

[Synced 9 events]

Current Subjects (2):
1. AI | Goal: 80 | Diff: 1
2. DSA | Goal: 80 | Diff: 3

--- Generated Weekly Plan ---
[] Wednesday 10:00 : Study AI
[] Wednesday 11:00 : Study AI
[] Wednesday 12:00 : Study AI
[] Wednesday 13:00 : Study AI
[] Thursday 10:00 : Study DSA
[] Thursday 11:00 : Study DSA
[] Thursday 12:00 : Study DSA
[] Thursday 13:00 : Study DSA
[] Friday 10:00 : Study DSA



Scope of Work

The scope of this project focuses on designing and implementing an AI-assisted study planning system within a Jupyter Notebook environment. The system aims to convert academic goals into a structured and realistic study schedule using machine learning and automation.

The project includes the following components:

- Development of a **Linear Regression-based machine learning model** to predict required study hours based on subject difficulty, target marks, and past study data.

- Automatic **generation of synthetic training data** when historical data is unavailable to ensure the system remains functional.
- Creation of an **interactive user interface using ipywidgets** for subject input, goal setting, and weekly availability configuration.
- Implementation of **study schedule generation logic** that breaks predicted study hours into one-hour blocks and allocates them across a 14-day period based on user availability.
- Integration with **Google Calendar** using OAuth2 authentication to export generated study sessions as calendar events.
- Handling of **capacity limitations**, including alerts when the total study load exceeds the user's available time.

GitHub repository Link:

<https://github.com/RehanAsif02/AI-Powered-Personalized-Study-Planner.git>