Project 1: Ai Lecture note generator

Transform lecture audio into structured, comprehensive notes using AI.

Features:
- 🎙️ Transcribes lecture audio using OpenAI's Whisper Large V3 model
- 📝 Generates well-structured markdown notes using Meta-Llama-3.1-8B
- 🧠 Identifies key concepts, points, and takeaways automatically
- 🚀 Simple web interface built with Gradio
- ⚡ Optimized with 4-bit quantization for efficient inference

How It Works

1. Audio Transcription: Converts lecture audio to text using Whisper Large V3
2. Note Generation: Processes transcription through Llama 3.1 to create structured notes
3. Format Cleaning: Ensures consistent markdown formatting with proper headers

Link to project: Abdulllah-Rizwan/AI-Lecture-s-Note-Generator: An AI agent that takes lecture audio as input and generates detailed, structured notes using Whisper for transcription and LLaMA 3.1 for language modeling.

Project 2: RAG-based Personal Knowledge Assistant

Overview

This project creates a personalized AI assistant that can understand and respond to questions about your personal information, projects, goals, and any other information you've stored in markdown files. It uses a combination of document retrieval techniques and Large Language Models to provide accurate, contextual responses.

Features

- **Personalized Knowledge Base**: Uses your own markdown files as a knowledge base
- **Smart Retrieval**: Finds relevant information from your knowledge files based on semantic similarity
- **Conversational Interface**: Chat with your knowledge through an intuitive Gradio web interface
- **Source Attribution**: Responses include references to which knowledge documents were used
- **Memory**: Remembers conversation context for more natural interaction

How It Works

1. **Document Processing**: The system reads all markdown files from your knowledge base
2. **Chunking**: Long documents are split into smaller, manageable chunks
3. **Embedding**: Each chunk is converted into a vector representation using OpenAI's embedding model
4. **Storage**: Embeddings are stored in a FAISS vector database for efficient retrieval
5. **Retrieval**: When you ask a question, the system finds the most relevant chunks
6. **Generation**: The chunks and your question are sent to the LLM, which generates a response
7. **Citation**: The sources of information are added to the response

Example questions:
- "What are my main goals and dreams?"
- "Tell me about my academic background"
- "What projects am I currently working on?"
- "What is my daily routine?"

Acknowledgments
This project uses the following libraries:
- LangChain for the RAG pipeline
- FAISS for vector storage
- Gradio for the user interface
- OpenAI for embeddings and text generation

Link to github: Abdulllah-Rizwan/Personal-Knowledge-Worker-Agent

Project 3: Code Language convertor

A versatile AI-powered tool for converting code between different programming languages while preserving functionality and idiomatic practices. The tool can also automatically generate comments, docstrings, and unit tests for your code.

Features

- **Multi-language Support**: Convert between Python, JavaScript, PHP, C++, and Golang
- **AI-Powered Conversion**: Utilizes both frontier and open-source language models
- **Docstring Generation**: Automatically add well-formatted comments and documentation
- **Write Comments**: Generate clear comments for the given code
- **Unit Test Creation**: Generate comprehensive unit tests for your given code
- **Multiple Model Options**: Choose between:
  - OpenAI GPT-4o Mini
  - Anthropic Claude 3.7 Sonnet
  - Grok 2.1
  - Code Llama 13B

How It Works

The application uses a Gradio interface to provide a user-friendly experience:

1. Select your preferred AI model
2. Choose source and target programming languages if you want to convert your code
3. Toggle options for writing docstrings/comments and unit tests
4. Paste your source code
5. Click "Convert Code" and watch results stream in real-time

Link to github: Abdulllah-Rizwan/Code-Language-Convertor

Project 4: Multimodal Ai career assistant

About the project
This project is a comprehensive AI-based career assistant designed to help users explore potential career options based on their interests, goals, and aspirations. The assistant provides career advice in a friendly and concise manner and leverages external tools to enhance the interaction. For example, after suggesting suitable career paths, it can retrieve the average cost of studying at a specified university and even generate a visual representation of a student at that institution using DALL-E 3.

Key features include:

- **Natural Language Processing:** Uses OpenAI's GPT models to analyze user input and generate context-aware career suggestions.
- **Tool Integration:** Supports function calls (tool calls) to fetch additional information such as the average tuition cost of universities.
- **Image Generation:** Uses DALL-E 3 to create vibrant pop-art style images that represent students in their respective universities.
- **Text-to-Speech:** Converts the assistant's text responses into audio using OpenAI's TTS model, providing an engaging multimodal interaction.
- **Real-Time Streaming:** Implements a streaming chat function to display responses as they are generated, ensuring a smooth user experience.
- **Web Interface:** Built with Gradio, the project features an interactive web UI where users can converse with the assistant, view generated images, and listen to the audio output.

This project demonstrates how to integrate multiple modalities (text, images, audio) with function calling capabilities to build a more interactive and informative AI assistant. It is ideal for users interested in exploring career options and associated educational costs through a rich, multimodal interface.

Features
- **Career Guidance:** Analyze user goals to suggest career paths.
- **University Cost Lookup:** Fetch average tuition costs for various universities using integrated tool calls.
- **Image Generation:** Create custom images that visually represent university experiences.
- **Audio Feedback:** Convert text responses to speech for a hands-free experience.
- **Interactive UI:** A clean and responsive interface built with Gradio.
- **Streaming Responses:** Real-time incremental display of AI responses.

Code Structure
- **app.ipynb:** Main application file containing the logic for setting up the chat interface, processing tool calls, streaming responses, and integrating image/audio functionalities.

- **Functions:**
  - `get_avg_university_cost`: Retrieves average tuition fees for a given university.
  - `handle_tool_calls`: Processes tool call data received from OpenAI and returns relevant information.
  - `artist`: Generates an image based on the provided university name using DALL-E 3.
  - `talker`: Converts text responses to speech using OpenAI's TTS model.
  - `chat`: Manages the conversation flow, handling streaming responses, tool calls, and updating the conversation history.
- **UI:** Constructed with Gradio, allowing users to interact with the assistant via a web interface.

Link to github: Abdulllah-Rizwan/Multimodal-Ai-Career-Assistant: A multimodal AI assistant that analyzes users' interests and goals to suggest suitable career paths. It integrates tool calls to provide additional data—such as the average cost of studying at a given university—and uses image generation and text-to-speech capabilities to deliver a rich, interactive user experience.

Project 5: Sentiment Analysis using simple RNN

This project on sentiment analysis was developed using a simple Recurrent Neural Network (RNN). The core of the project involved training the RNN model with TensorFlow on the IMDB dataset, focusing on accurately classifying sentiments in textual data. For the frontend, I utilized Streamlit, a powerful and user-friendly framework, to create an interactive web application. This streamlined approach allowed for real-time sentiment analysis, making the tool accessible and easy to use for end-users. The combination of TensorFlow's robust deep learning capabilities and Streamlit's seamless frontend integration resulted in a highly efficient sentiment analysis tool.

Link to github: Abdulllah-Rizwan/Sentiment-Analysis-Using-Simple-RNN

Project 6: **Customer Churn Prediction Using Artificial Neural Network**

This project focuses on predicting customer churn by analyzing various factors such as age, gender, and estimated salary. Utilizing an Artificial Neural Network (ANN) trained with TensorFlow, one of the most widely used libraries, the model achieved an accuracy exceeding 80% on test data. This high level of accuracy underscores the model's effectiveness in identifying potential customer churn, providing valuable insights for strategic decision-making.

Link to github: Abdulllah-Rizwan/ANN-Customer-Churn-Classification-Deep-Learning-Project

Project 7: End to end Student performance ML

In this end-to-end machine learning project, I explored the factors impacting student performance, specifically analyzing how variables like Gender, Ethnicity, Parental Education Level, Lunch Type, and Test Preparation Course influence test scores across Math, Reading, and Writing. The goal was to identify correlations between these variables and academic performance, using insights from data analysis and predictive modeling.

During the analysis phase, it was observed that students with a standard lunch achieved higher scores, suggesting a link between nutrition and academic performance. Students whose parents had higher education levels also performed better, indicating greater educational support at home. Group differences showed Group C outperforming others, Group E having notable high achievers, and Group A facing academic challenges. Females scored higher overall, while males excelled slightly in Math. Multiple regression models were trained to predict student performance, with linear regression being the most accurate. Modular pipelines were implemented for data ingestion, transformation, model training, and predictions, ensuring scalability and reproducibility. The analysis provided insights into key influences on student performance, with implications for improving educational support and resource allocation.
Link to github: Abdulllah-Rizwan/End-To-End-Student-Performance-ML-Project

Project 8: Holiday package prediction ML

This project aims to classify holiday package preferences based on extensive data analysis and machine learning techniques. After thorough data cleaning and feature engineering, various classification algorithms were trained and evaluated. The Random Forest model emerged as the most accurate predictor for holiday package preferences. To maximize its predictive power, the model's hyperparameters were meticulously fine-tuned. The project includes a comprehensive tutorial notebook, which guides readers through the entire process with detailed comments. The notebook is available on GitHub, providing valuable insights into the methodology and results.

Link to github: Abdulllah-Rizwan/Holiday-Package-Predication-Classification-Problem

Project 9: Google Advanced data analytics capstone project

# Employee Retention Prediction using Random Forest and Decision Trees
A predictive analysis project on identifying factors influencing employee turnover using machine learning models.

# Project Overview
This project seeks to predict employee retention at Salifort Motors using various machine learning techniques. Leveraging employee-related data, we employed decision trees and random forest algorithms to predict which employees are most likely to leave the company. The goal was to identify the most influential factors behind employee turnover, which can guide HR in taking actionable steps to improve retention. The random forest model outperformed the decision tree model in predictive accuracy, offering robust insights.

# Business Understanding
Salifort Motors has faced challenges with employee retention and wanted to address the root causes behind their high turnover rates. The business stakeholders needed to understand which factors most significantly contribute to employees leaving the company and how they could reduce attrition rates. By identifying key factors such as workload, evaluation frequency, tenure, and overwork, this project provides a foundation for HR to implement better policies to improve job satisfaction and reduce turnover.

# Data Understanding
The data used in this project consisted of employee records including metrics such as 'last_evaluation', 'number_project', 'overworked', 'tenure', and others that

could contribute to employee attrition. The dataset was a historical record spanning over several years. Some limitations included the exclusion of employee sentiment data and external economic factors which may also influence turnover. Exploratory analysis revealed the most critical features, such as the number of projects, last evaluation scores, and work conditions.

# Modeling and Evaluation
We utilized both decision trees and random forest classifiers to build predictive models. The random forest model performed slightly better than the decision tree model due to its ability to handle more complex feature interactions. Feature importance metrics revealed that 'last_evaluation', 'number_project', and 'tenure' were the top three factors driving employee turnover. The evaluation metrics, including accuracy, precision, and recall, showed that the random forest model was more reliable for predicting employee exit.

# Conclusion
Based on our analysis, we recommend that Salifort Motors caps the number of projects assigned to employees and ensures a better work-life balance to prevent burnout. Additionally, recognizing employees who consistently meet expectations and further investigating why long-tenured employees are leaving will help improve retention. Future steps involve integrating employee satisfaction surveys and exploring deeper connections between specific departmental challenges and turnover rates.

Link to github:
Abdulllah-Rizwan/Google-Advanced-Data-Analytics-Capstone-Project

Project 10: Ecommerce Apis

The project evolved into a multifaceted system with segments dedicated to users, categories, carts, orders, and products. Users could be standard users or admins, with admins handling administrative tasks. The API supported user authentication, CRUD operations, and ensured a robust user experience. Product images were organized in a unique directory structure under relevant categories. Carts and orders were managed to ensure a smooth shopping experience. Alembic handled database migrations seamlessly. Security was ensured with JWT tokens for authentication, and Pydantic validated requests and responses for data integrity. The API was crafted following best coding practices, clear folder structures, and versioning for future scalability. Ultimately, this project aimed to create a seamless and delightful shopping experience through a well-developed eCommerce API.

Link to github: https://github.com/Abdulllah-Rizwan/Ecommerce-Apis

Project 11: Social Media Apis

This project focuses on developing social media-like APIs using Python and the FastAPI framework, emphasizing efficient API creation. It includes user registration and login, enabling functionalities like creating, reading, updating, and deleting posts. A robust authentication and authorization system using JSON Web Tokens (JWT) ensures secure access. The project uses PostgreSQL for data management and the Pydantic library for data validation, ensuring integrity and consistency. It incorporates database joins for efficient data retrieval and Alembic for smooth database migrations. Comprehensive unit testing with Pytest ensures code reliability. This project demonstrates a thorough approach to building secure, robust, and scalable APIs

Link to github: Abdulllah-Rizwan/Social-Media-Apis