GitHub Projects Portfolio

Profile: https://github.com/Muhammad-Muzammil-Shah

Generated: February 19, 2026

1. AI-Chat-Assistant

■ https://github.com/Muhammad-Muzammil-Shah/AI-Chat-Assistant

README

AI Chat Assistant

An AI-powered chat assistant that supports both text and voice inputs. The assistant generates responses using a conversational AI model and provides voice feedback to the user. This project uses **Streamlit** for the UI, **Speech Recognition** for voice input, and **Google Text-to-Speech (gTTS)** for voice output.

Features

- **Text Input**: Users can interact with the assistant by typing text.

- **Voice Input**: Users can speak to the assistant and get a text response.

- **Voice Output**: The assistant will speak its responses using Text-to-Speech.

Table of Contents

1. [Installation Instructions](#installation-instructions)

2. [Running the Project](#running-the-project)

3. [Project Structure](#project-structure)

Activate the virtual environment:

- **API Errors**:

AI Chat Assistant

An AI-powered chat assistant supporting both text and voice inputs. The assistant generates responses using Groq's conversational AI model and provides voice feedback. Built with **Streamlit** for UI, **SpeechRecognition** for voice input, and **gTTS** for voice output.

■ Features

- **Text Input**: Type messages to interact with the assistant.

- **Voice Input**: Speak to the assistant and get text/voice responses.

- **Voice Output**: Assistant speaks responses using Text-to-Speech.

■■ Installation

1. Clone the repository:

```bash

git clone https://github.com/your-username/ai-chat-assistant.git

cd ai-chat-assistant
```

```

2. Create a virtual environment:

```bash

python -m venv venv

.\venv\Scripts\activate # Windows

# Or

source venv/bin/activate # macOS/Linux

```

3. Install dependencies:

```bash

pip install -r requirements.txt

```

■ Usage

1. Set your Groq API key in `chatbot.py`:

```python

GROQ_API_KEY = "your_groq_api_key_here"

```

2. Run the Streamlit app:

```bash

streamlit run chatbot.py

```

3. Interact via text or voice in your browser at `http://localhost:8501`.

■ Project

... (truncated)

2. AI-IGNITE-WEEK-Technical-Track

■ https://github.com/Muhammad-Muzammil-Shah/AI-IGNITE-WEEK-Technical-Track

README

■ AI Ignite Week - Technical Track

■ Code Explanation Summary

This repository contains 7 AI/ML projects, each implemented in Python using libraries like scikit-learn, NLTK, TextBlob, and Flask. Here’s what each task’s code does:

- **Task 1: Smart To-Do Priority Predictor**

- Uses machine learning (scikit-learn) to predict and prioritize tasks based on features. Compares different ML algorithms for best accuracy.

- **Task 2: Sentiment Analyzer**
```

- Analyzes customer reviews using TextBlob for sentiment (positive, negative, neutral). Visualizes results with charts and word clouds.

- **Task 3: Meeting Time Tracker**

- Parses meeting transcripts, calculates speaking time for each participant, and visualizes balance and distribution using matplotlib.

- **Task 4: Email Classification System**

- Classifies emails into categories using Naive Bayes and TF-IDF vectorization. Includes EDA, word cloud, and confusion matrix.

- **Task 5: Study Notes Summarizer**

- Summarizes lecture notes using NLTK and keyword frequency. Generates short and detailed summaries for quick revision.

- **Task 6: Music Mood Classifier**

- Predicts song mood from audio features using decision trees. Handles large datasets and provides performance metrics.

- **Task 7: AI Study Buddy (Web App)**

- Uses Groq API and Flask to create a web app that generates quiz questions from study notes. Interactive and API-powered.

Each task folder contains Jupyter notebooks or Python scripts with step-by-step code, comments, and visualizations to help you understand the implementation.

Complete implementation of all 7 AI/ML tasks with modern solutions using Python, Flask, and various AI APIs.

■ Project Overview

This repository contains **7 complete AI/ML projects** covering different aspects of artificial intelligence and machine learning.

■ Completed Tasks

| Task | Project Name | Technology Stack | Status | Key Features |

|:----:|:-------------|:-----------------|:------:|:----------

... (truncated)

3. AI_Voice_Assistant_Bots

■ https://github.com/Muhammad-Muzammil-Shah/AI_Voice_Assistant_Bots

README

AI Voice Assistant Bots

A real-time voice chatbot web application that combines browser speech recognition, streaming AI responses via Groq LLM, and text-to-speech synthesis. The entire application is contained in a single Python file for simplicity.

Features

■ **Real-time Voice Recognition** - Uses browser's built-in speech recognition

■ **AI-Powered Responses** - Powered by Groq LLM with streaming responses

■ **Text-to-Speech** - Speaks responses back with natural voice synthesis

■ **Live Chat Interface** - Real-time conversation display with visual feedback

■ **Responsive Design** - Works on desktop and mobile browsers

■■ **Graceful Fallback** - Works offline without AI when API unavailable

Architecture

- **Backend**: Flask server with embedded HTML/CSS/JS template

- **AI Integration**: Groq API for chat completions with graceful fallback mode

- **Frontend**: Single-page application with real-time voice interface

- **Streaming**: Server-Sent Events (SSE) for token-by-token response delivery

Requirements

- Python 3.8+

- Chrome/Edge browser (recommended for speech recognition)

- Microphone access

AI Voice Assistant Bots

A real-time voice chatbot web application that combines browser speech recognition, streaming AI responses via Groq LLM, and text-to-speech synthesis. The entire application is contained in a single Python file for simplicity.

■ Features

- Real-time voice recognition (browser-based)

- AI-powered responses via Groq LLM (streaming)

- Text-to-speech for spoken responses

- Live chat interface with visual feedback

- Responsive design for desktop and mobile

- Graceful fallback (works offline if API unavailable)

■■ Installation

1. Clone the repository:

```bash
git clone https://github.com/Muhammad-Muzammil-Shah/AI_Voice_Assistant_Bots.git
cd AI_Voice_Assistant_Bots
```

2. Create a virtual environment:

```bash
python -m venv .venv
.venv\Scripts\activate # Windows
# Or
source .ve
```

... (truncated)

4. Assignment01

■ https://github.com/Muhammad-Muzammil-Shah/Assignment01

README

Assignment01

5. Assignment02

■ https://github.com/Muhammad-Muzammil-Shah/Assignment02

README

README not found.

6. Automation-fb-whatsapp

■ https://github.com/Muhammad-Muzammil-Shah/Automation-fb-whatsapp

README

Facebook & WhatsApp Automation Scripts

Automate Facebook and WhatsApp tasks using Python scripts powered by pywhatkit. Includes:

- WhatsApp message sending (`massege.py`)
- WhatsApp image sharing (`Picture_Sending.py`)
- WhatsApp group messaging (`Msg in Group.py`)
- Facebook posting via Jupyter notebook

Quick Start

1. Install dependencies:

```bash
pip install pywhatkit
```

2. Update phone numbers, group IDs, and image paths in scripts as needed.
3. Run scripts:

```bash
python Whatsapp Automation/massege.py
python Whatsapp Automation/Picture_Sending.py
python Whatsapp Automation/Msg in Group.py
```

Example Usage

```python
import pywhatkit
pywhatkit.sendwhatmsg("+923302358711", "Hello, This is a test message", 5, 5, 15, True, 2)
pywhatkit.sendwhats_image("+923302358711", "img.png", "Image sent from Python", 15, True, 3)
pywhatkit.sendwhatmsg_to_group_instantly("E9HUUeote6J2SDkqVCxRYA", "Hey All!")
```

```

## License

Open source. Feel free to use, modify, and distribute.

## 7. Azure-AI-Studio---Multi-Modal-Generator

■ https://github.com/Muhammad-Muzammil-Shah/Azure-AI-Studio---Multi-Modal-Generator

***README***

# ■ Azure AI Studio - Multi-Modal Generator

A powerful desktop application for AI-powered content generation using Azure AI services. Generate high-quality **Text-to-Speech audio** and **AI videos** with an intuitive graphical interface.

![Python](https://img.shields.io/badge/Python-3.10+-blue.svg)

![Platform](https://img.shields.io/badge/Platform-Windows-lightgrey.svg)

![Azure](https://img.shields.io/badge/Azure-AI%20Services-0078D4.svg)

![License](https://img.shields.io/badge/License-MIT-green.svg)

---

## ■ Features

### ■ Text-to-Speech (TTS)

- **23+ AI Voices** - alloy, echo, fable, onyx, nova, shimmer, coral, verse, ballad, ash, sage, marin, cedar, and more

- **Word Document Support** - Upload `.docx` files directly

- **Audio Playback Controls** - Play, Pause, Resume functionality

- **Download Audio** - Save generated audio as MP3 files

### ■ AI Video Generation (Sora)

- **Text-to-Video** - Generate videos from text prompts

- **Multiple Resolutions** - Support for 480p to 1080p (1920x1080)

- **Customizable Duration** - Set video length in seconds

- **Auto Status Checking** - Automatic polling for video generation status

- **Download Videos** - Save generated videos to your computer

### ■ Configuration

- **Separate API Settings** - Independent configuration for TTS and Video services

- **Persistent Settings** - Configuration saved automatically

- **Easy Setup** - Simple GUI for API key management

---

## ■ Project Structure

```

multi-model/

■■■ azure_ai_studio_app.py # Main application (GUI)

■■■ app.py # Simple video generation script

■■■ azure_ai_config.json # API configuration file

■■■ requirements.txt # Python dependencies

■■■ build_desktop_app.py # PyInstaller build script

■■■ build_desktop.bat # Windows build batch file

■■■ build_config.py # Build configuration settings

■■■ move_app.bat # App distribution helper

■■■ AzureAIStudio.spec # PyInstaller spec file

■■■ DESKTOP_A

... (truncated)

8. AzureOpenAI

■ https://github.com/Muhammad-Muzammil-Shah/AzureOpenAI

README

AzureOpenAI

Here’s a step-by-step README file for your project, including instructions on cloning, installing dependencies, and running the script:

Azure OpenAI DALL-E 3 Image Generator

This project uses Azure OpenAI's DALL-E 3 model to generate images based on a given prompt. The generated images are saved locally and displayed using Python.

Prerequisites

- Python 3.12 or later installed on your system.

- Azure OpenAI API Key and Endpoint.

Getting Started

1. Clone the Repository

Clone this repository to your local machine:

```bash

git clone https://github.com/Muhammad-Muzammil-Shah/AzureOpenAI.git

```

2. Set Up a Virtual Environment (Optional but Recommended)

Create and activate a virtual environment:

```bash

python -m venv venv

```

- On Windows:

```bash

.\venv\Scripts\activate
```

```

### 3. Install Dependencies

Install the required Python packages:

```bash
pip install -r requirements.txt
```

Configuration

4. Add Azure OpenAI Credentials

Update the script with your Azure OpenAI API Key and Endpoint:

- Replace `<API Key>` with your Azure OpenAI API key.
- Replace `<API Endpoint>` with your Azure OpenAI endpoint URL.

Running the Script

5. Execute the Script

Run the Python script to generate an image:

```bash
python DALL-E-3.py
```

The script will:

1. Generate an image based on the provided prompt.
2. Save the image in the `images` directory (created automatically if it doesn't exist).
3. Open the image in your default image viewer.

Notes

- Ensure your Azure subscription allows access to DALL-E 3.
- If you encounter errors, check for missing dependencies or verify the API credentials.

Troubleshooting

Common Errors

1. **ModuleNotFoundError**: Ensure all dependencies are installed using `pip install -r requirements.txt`.
2. **API Access Issues**: Verify your API Key and Endpoint.

Feel free to modify the code and experiment with different prompts. Happy coding!

9. Blood-Report-Analyzer-Using-Llm

■ https://github.com/Muhammad-Muzammil-Shah/Blood-Report-Analyzer-Using-Llm

README

Blood Report Analyzer Using LLM
```

This project is a Streamlit web app that analyzes blood test results and provides medical advice in Urdu using Groq LLM. It supports text-to-speech for Urdu responses and covers a wide range of medical test categories.

■ Features

- Enter blood test results and get AI-powered advice in Urdu
- Supports multiple test categories (CBC, Chem 7, Heart Disease, Thyroid, etc.)
- Custom test entry supported
- Uses Groq LLM for medical analysis
- Urdu text-to-speech via gTTS
- Streamlit UI for easy interaction

■■ Installation

1. Clone the repository:

```bash
git clone https://github.com/Muhammad-Muzammil-Shah/Blood-Report-Analyzer-Using-Llm.git
cd Blood-Report-Analyzer-Using-Llm
```

2. (Optional) Create a virtual environment:

```bash
python -m venv venv
venv\Scripts\activate # Windows
# Or
source venv/bin/activate # macOS/Linux
```

3. Install dependencies:

```bash
pip install -r Requirements.txt
```

■ Usage

1. Add your Groq API key to the script (replace `GROQ_API_KEY` value).
2. Run the Streamlit app:

```bash
streamlit run blood_report.py
```

3. Enter patient info and test results in the web UI.
4. Click "Analyze" to get Urdu medical advice and listen to the audio response.

■ Technologies

- Python

- Streamlit

- Groq LLM API

- gTTS (Google Text-to-Speech)

- requests

■ Notes

- Supports custom test entry for flexibility

- All advice is generated by AI and should be reviewed by a medical professional

---

■ License

Open source. Feel free to use, modify, and distribute.

10. Buildables-Artificial-Intelligence-Fellowship

■ https://github.com/Muhammad-Muzammil-Shah/Buildables-Artificial-Intelligence-Fellowship

README

■# Buildables-Artificial-Intelligence-Fellowship

■ Overview

This repository is designed for beginners and students to learn about Artificial Intelligence (AI), Large Language Models (LLMs), APIs, and Python programming. It includes hands-on Jupyter notebooks that explain key concepts and provide practical coding exercises.

■ Notebook Summaries

- **Week 1:** Introduction to AI, LLMs, APIs, and Python for API interaction. Learn the basics of how computers mimic human thinking, how LLMs process language, and how APIs enable automation. Includes practical examples using Python's `requests` library and JSON data.

- **Week 2:** Conversational AI concepts, model pipelines, and hands-on chatbot development. Covers LLMs, Speech-to-Text (STT), Text-to-Speech (TTS), and using APIs (Groq) for building chatbots with memory and preprocessing.

- **Week 3:** Design and build a complete chatbot project using Groq LLMs. Learn conversation flow, prompt engineering, and justification for educational Q&A bots (e.g., for mathematics).

- **Week 5:** Retrieval Augmented Generation (RAG) with LangChain, Pinecone, and Groq. Learn how to build knowledge bases, use vector databases, and connect external data to LLMs for up-to-date answers.

- **Week 6:** Deep dive into API development with Flask and FastAPI. Covers REST principles, CRUD operations, authentication, and building scalable web APIs with Python.

■ What You'll Learn

- Basics of AI and how computers can mimic human thinking

- Introduction to LLMs (like ChatGPT) and how they process language

- Understanding APIs and their role in automation
- Using Python (`requests` library) to interact with APIs and handle JSON data
- Setting up coding environments in Google Colab or Jupyter Notebook
- How to share your work on GitHub
- Building RESTful APIs with Flask and FastAPI

■ Example Activities

- Make API calls using Python and display results
- Work with JSON data from APIs
- Sign up for

... (truncated)

11. CallBotX-AI-Powered-Voice-Agents-for-E-Commerce-Support

■ https://github.com/Muhammad-Muzammil-Shah/CallBotX-AI-Powered-Voice-Agents-for-E-Commerce-Support

README

CallBotX: AI-Powered Voice Agents for E-Commerce Support

■ Code Explanation

This project uses Jupyter notebooks and Python scripts to build a Retrieval-Augmented Generation (RAG) chatbot for e-commerce support. The main notebook (`RAG_Chatbot.ipynb`) demonstrates:

- Loading product data (PDFs, docs)
- Splitting and embedding documents for semantic search
- Using Langchain and Groq Llama3 for question answering
- Building a retriever and prompt template for context-aware responses
- Streaming answers and converting them to voice (gTTS)

Example Workflow

1. Load product data (e.g., iPhone specs PDF)
2. Split text into chunks and embed for retrieval
3. Use Langchain retriever to find relevant context
4. Pass context and user question to Llama3 model
5. Return answer and optionally convert to speech

Usage

Run `RAG_Chatbot.ipynb` in Jupyter. Follow the code cells to:

- Set up API keys and environment
- Load and process product documents
- Ask questions and get AI-powered answers
- Convert answers to voice using gTTS

This repo is ideal for building custom voice/text AI agents for product support.

CallBotX is a platform that leverages Retrieval-Augmented Generation (RAG) techniques and natural voice interaction to answer user queries about specific products (initially focusing on iPhones 8–16). It enables companies to upload their product information and, within minutes, deploy a custom AI assistant capable of handling real customer conversations in both text and voice formats.

Features

- **AI Voice & Text Agents:** Instantly create AI agents that can interact with customers using natural language via voice and chat.

- **Retrieval-Augmented Generation (RAG):** Uses advanced retrieval techniques to provide accurate, up-to-date answers based on product documentation and FAQs.

- **Rapid Onboarding:** Companies can upload their product information and quickly deploy an AI-powered assistant.

- **Multimodal Support:** Handles both voice and text interactions

... (truncated)

12. cde-b1-assignments

■ https://github.com/Muhammad-Muzammil-Shah/cde-b1-assignments

README

$ cd C:\Users\Anwaar-ul-Karim Shah\Desktop\Assignment

bash: cd: too many arguments

Anwaar-ul-Karim Shah@MAAAKS MINGW64 ~

$ cd Desktop\Assignment

bash: cd: DesktopAssignment: No such file or directory

Anwaar-ul-Karim Shah@MAAAKS MINGW64 ~

$ cd Desktop/Assignment

Anwaar-ul-Karim Shah@MAAAKS MINGW64 ~/Desktop/Assignment

$ git clone https://github.com/Muhammad-Muzammil-Shah/cde-b1-assignments.git

Cloning into 'cde-b1-assignments'...

remote: Enumerating objects: 4, done.

remote: Counting objects: 100% (4/4), done.

remote: Compressing objects: 100% (3/3), done.

remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0 (from 0)

Receiving objects: 100% (4/4), done.

Anwaar-ul-Karim Shah@MAAAKS MINGW64 ~/Desktop/Assignment

$ cd cde-b1-assignments/

Anwaar-ul-Karim Shah@MAAAKS MINGW64 ~/Desktop/Assignment/cde-b1-assignments (main)

$ ls

M_muzammil-340493/ assignment/

Anwaar-ul-Karim Shah@MAAAKS MINGW64 ~/Desktop/Assignment/cde-b1-assignments (main)

$ git status

On branch main

Your branch is up to date with 'origin/main'.

Untracked files:

(use "git add <file>..." to include in what will be committed)

M_muzammil-340493/

nothing added to commit but untracked files present (use "git add" to track)

Anwaar-ul-Karim Shah@MAAAKS MINGW64 ~/Desktop/Assignment/cde-b1-assignments (main)

$ git add .

warning: in the working copy of 'M_muzammil-340493/Assignment 01 (Python Basics).ipynb', LF will be replaced by CRLF the next time Git touches it

Anwaar-ul-Karim Shah@MAAAKS MINGW64 ~/Desktop/Assignment/cde-b1-assignments (main)

$ git commit -m "saylani 1st assinment"

[main a8f0fbd] saylani 1st assinment

1 file changed, 951 insertions(+)

create mode 100644 M_muzammil-340493/Assignment 01 (Python Basics).ipynb

Anwaar-ul-Karim Shah@MAAAKS MINGW64 ~/Desktop/Assignment/cde-b1-assignments (main)

$ git push -u origin

Enumerating objects: 5, done.

Counting objects: 100% (5/5), done.

Delta compression using up to 8 threads

Compressing objects: 100% (4/4), done.

Writing objects:

... (truncated)

13. chatboot_using_llama3.1

■ https://github.com/Muhammad-Muzammil-Shah/chatboot_using_llama3.1

README

Chatbot Using Llama2 (Langchain + Streamlit)

Overview

This project is a simple chatbot web app built with Python, Streamlit, and Langchain. It uses the Llama2 language model (via Ollama) to answer user questions interactively. The app demonstrates how to integrate LLMs with a prompt template and a user-friendly interface.

Chatbot Using Llama2 (Langchain + Streamlit)

Overview

This project is a simple chatbot web app built with Python, Streamlit, and Langchain. It uses the Llama2 language model (via Ollama) to answer user questions interactively. The app demonstrates how to integrate LLMs with a prompt template and a user-friendly interface.

■ How It Works

- Uses Langchain's prompt template to structure user queries
- Integrates Llama2 model via Ollama for generating responses
- Streamlit provides a web interface for user input and displaying answers
- Environment variables are loaded using `python-dotenv` for API keys and configuration

---

■ Main Files

- `app.py`: Streamlit app for chatbot interface and Llama2 integration
- `main.py`: Additional logic or entry point (if present)
- `requirements.txt`: List of required Python packages

■■ Usage

1. Install dependencies:

```bash
pip install -r requirements.txt
```

2. Run the Streamlit app:

```bash
streamlit run app.py
```

3. Enter your question in the input box and get answers from Llama2

■ Example Code

```python
import streamlit as st
from langchain_community.llms import Ollama
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.output_parsers import StrOutputParser
st.title('Langchain Demo With LLAMA2 API')
input_text = st.text_input("Search the topic you want")
llm = Ollama(model="llama2")
prompt = ChatPromptTemplate.from_messages([
("system", "You are a helpful assistant. Please respond to the user queries."),
("user", "Question: {question}")
])
output_parser = StrOutputParser()
chain = prompt | llm | output_parser
if input_
... (truncated)
```

14. chatbot

■ https://github.com/Muhammad-Muzammil-Shah/chatbot

README

README not found.

15. Chatbot_llama3

■ https://github.com/Muhammad-Muzammil-Shah/Chatbot_llama3

README

Groq Chat Completion Example

This repository demonstrates how to use the **Groq API** to create a simple chat application with Python. The script takes user input, sends it to the Groq API, and prints the model's response.

Getting Started

Follow these steps to set up and run the script:

Step 1: Create an API Key

1. Visit the [Groq API Key Page](#) to generate your API key.

Step 2: Set Up Your Environment

It’s recommended to set your API key as an environment variable to enhance security.

In your terminal:

```bash
import os
os.environ["GROQ_API_KEY"] = ""
```

Step 3: Install Dependencies

Install the required Groq Python library:

```bash
pip install groq
```

Step 4: Run the Script

1. Clone this repository:

```bash
git clone https://github.com/your-username/groq-chat-example.git
cd groq-chat-example
```

2. Run the Python script:
```bash
python app.py
```

Code Overview
Here’s what the script does:
Python Code:
```python
import os
os.environ["GROQ_API_KEY"] = "GROQ_API_KEY"
from groq import Groq
# Initialize the Groq client with the API key
client = Groq(
api_key=os.environ.get("GROQ_API_KEY"),
)
# Prompt user input and send a message to the Groq API
chat_completion = client.chat.completions.create(
messages=[
{
"role": "user",
"content": input("Chat with me Dear: "),
}
],
model="llama3-8b-8192", # Specify the model
)
# Print the response from the model
print(chat_completion.choices[0].message.content)
```
How It Works:
1. **API Key**: The script fetches the API key from the environment variable `GROQ_API_KEY`.
2. **User Input**: Prompts the user with `Chat with me Dear: ` and captures the input.
3. **Groq API Call**: Sends the input to the `llama3-8b-8192` model.
4. **Prints Response**: Displays the model’s response in the terminal.

Next Steps
- Experiment with oth
... (truncated)

16. Cloud_Data_Engineer

■ https://github.com/Muhammad-Muzammil-Shah/Cloud_Data_Engineer

README

Cloud Data Engineer: Gapminder Data Visualization

This project demonstrates data visualization techniques using the Gapminder dataset in Python. The included Jupyter notebook walks through loading, analyzing, and visualizing global development data with pandas, numpy, and matplotlib.

■ Features

- Load and explore the Gapminder dataset
- Visualize GDP per capita vs. life expectancy
- Use bubble charts to show population size
- Color-code countries by region
- Add custom labels and grid for clarity
- Interpret trends and regional disparities

■■ Installation

1. Clone the repository:

```bash
git clone https://github.com/Muhammad-Muzammil-Shah/Cloud_Data_Engineer.git
cd Cloud_Data_Engineer
```

2. (Optional) Create a virtual environment:

```bash
python -m venv venv
venv\Scripts\activate # Windows
# Or
source venv/bin/activate # macOS/Linux
```

3. Install dependencies:

```bash
pip install numpy pandas matplotlib
```

■ Usage

1. Open `CaseStudy Data Visualization.ipynb` in Jupyter or Colab.
2. Run the cells step-by-step to:
- Load and inspect the dataset
- Create scatter and bubble plots

- Analyze trends in GDP, life expectancy, and population

3. Review the interpretation and key observations at the end of the notebook.

■ Technologies

- Python

- Jupyter Notebook

- pandas, numpy, matplotlib

■ License

Open source. Feel free to use, modify, and distribute.

17. Computer_vision

■ https://github.com/Muhammad-Muzammil-Shah/Computer_vision

README

Computer Vision: Image Filtering in Python

This project demonstrates basic image filtering and manipulation techniques using Python, Pillow, and NumPy. The included Jupyter notebook walks through pixel-level operations, color channel adjustments, grayscale conversion, color space transformations, and more.

■ Features

- Load and display images

- Manipulate pixel values (e.g., set red channel to zero)

- Save and view modified images

- Convert images to grayscale

- Shift color channels and clamp pixel values

- Adjust saturation in HSV color space

- Print image details and pixel values

■■ Installation

1. Clone the repository:

```bash

git clone https://github.com/Muhammad-Muzammil-Shah/Computer_vision.git

cd Computer_vision

```

2. (Optional) Create a virtual environment:

```bash

python -m venv venv

venv\Scripts\activate # Windows
```

Or

source venv/bin/activate # macOS/Linux

```

3. Install dependencies:

```bash

pip install pillow numpy

```

■ Usage

1. Open `Image Filter.ipynb` in Jupyter or Colab.

2. Run the cells step-by-step to:

- Load and inspect images

- Apply pixel and color transformations

- Save and view results

3. Experiment with your own images by replacing `dog.jpg`.

■ Technologies

- Python

- Jupyter Notebook

- Pillow (PIL)

- NumPy

■ License

Open source. Feel free to use, modify, and distribute.

18. Crypto-Codex

■ https://github.com/Muhammad-Muzammil-Shah/Crypto-Codex

README

■# Crypto-Codex

Crypto Codex is an educational web tool for basic encoding and decoding methods. It is now a fully static website (HTML, CSS, JavaScript) and works perfectly on GitHub Pages or any static web host.

Features

- Caesar Cipher (encode/decode with custom shift)

- ASCII <-> Hex conversion

- ASCII <-> Decimal conversion

- Modern, responsive UI (light color theme)

How to Use

1. Clone or download this repository:
```

```

git clone https://github.com/Muhammad-Muzammil-Shah/Crypto-Codex.git

```

2. Open `index.html` in your browser.

- Or deploy to GitHub Pages for free static hosting.

GitHub Pages Deployment

1. Push your code to a GitHub repository.

2. Go to your repository settings > Pages.

3. Set the source branch to `main` and the folder to `/root` (or `/docs` if you move files).

4. Visit your published site at `https://<your-username>.github.io/<repo-name>/`.

Usage

- Select a tool tab (Caesar Cipher, ASCII <-> Hex, ASCII <-> Dec)

- Enter your message and choose the desired operation

- View the result in the output area

License

This project is licensed under the MIT License.

19. Data-Analytics-with-R

■ https://github.com/Muhammad-Muzammil-Shah/Data-Analytics-with-R

README

Data Analytics with R

This project demonstrates data analytics and visualization techniques using R. The repository includes scripts and notebooks for exploratory data analysis, multiple plots, aggregate visualizations, and practical R exercises.

■ Features

- Exploratory data analysis with R

- Multiple types of plots and visualizations

- Aggregate data visualization

- Practice scripts for learning R basics

- Example assignments and documentation

■■ Installation

1. Clone the repository:

```bash

git clone https://github.com/Muhammad-Muzammil-Shah/Data-Analytics-with-R.git

cd Data-Analytics-with-R

```

2. Open RStudio or your preferred R environment.

■ Usage

1. Explore the R scripts (`R_practice_01.R`, `R_practice_02.R`, `R_Script_02.R`) for hands-on analytics and visualization.

2. Review the folders for assignments and example plots.

3. Run scripts in RStudio or directly in the R console.

■ Technologies

- R

- RStudio

- Data visualization libraries (base R, ggplot2, etc.)

■ License

Open source. Feel free to use, modify, and distribute.

20. Data_Mining

■ https://github.com/Muhammad-Muzammil-Shah/Data_Mining

README

Data Mining: Real Estate Price Analysis

This project demonstrates data mining and statistical analysis techniques using Python and pandas. The included Jupyter notebook walks through cleaning, transforming, and visualizing real estate price data from Karachi.

■ Features

- Load and clean real estate data (CSV)

- Map categorical variables to numeric codes

- Analyze price distributions with descriptive statistics

- Visualize data with histograms, boxplots, and log-log plots

- Remove outliers using IQR filtering

- Fit and visualize power-law distributions

■■ Installation

1. Clone the repository:

```bash

git clone https://github.com/Muhammad-Muzammil-Shah/Data_Mining.git

cd Data_Mining

```

2. (Optional) Create a virtual environment:

```bash
python -m venv venv
venv\Scripts\activate # Windows
# Or
source venv/bin/activate # macOS/Linux
```

3. Install dependencies:

```bash
pip install pandas numpy matplotlib seaborn scipy
```

■ Usage

1. Open `Lab_03.ipynb` in Jupyter or Colab.
2. Run the cells step-by-step to:
- Load and clean the dataset
- Map categorical variables
- Visualize price distributions and remove outliers
- Fit and plot power-law distributions
3. Review the interpretation and key observations at the end of the notebook.

■ Technologies

- Python
- Jupyter Notebook
- pandas, numpy, matplotlib, seaborn, scipy

■ License

Open source. Feel free to use, modify, and distribute.

21. Data_Mining_labs

■ https://github.com/Muhammad-Muzammil-Shah/Data_Mining_labs

README

Data Mining Labs: Normalization and MapReduce

This project contains Jupyter notebooks for data mining lab exercises, focusing on data normalization and MapReduce concepts using real estate data from Karachi.

■ Features

- Load and clean real estate data (CSV)
- Map categorical variables to numeric codes

- Analyze price distributions with descriptive statistics
- Visualize data with histograms, boxplots, and log-log plots
- Remove outliers using IQR filtering
- Fit and visualize power-law distributions
- MapReduce lab for scalable data processing

■■ Installation

1. Clone the repository:

```bash
git clone https://github.com/Muhammad-Muzammil-Shah/Data_Mining_labs.git
cd Data_Mining_labs
```

2. (Optional) Create a virtual environment:

```bash
python -m venv venv
venv\Scripts\activate # Windows
# Or
source venv/bin/activate # macOS/Linux
```

3. Install dependencies:

```bash
pip install pandas numpy matplotlib seaborn scipy
```

■ Usage

1. Open `Normalization_data_Lab03.ipynb` and `MapRedduce_lab04.ipynb` in Jupyter or Colab.
2. Run the cells step-by-step to:
- Load and clean the dataset
- Map categorical variables
- Visualize price distributions and remove outliers
- Fit and plot power-law distributions
- Explore MapReduce concepts
3. Review the interpretation and key observations at the end of the notebooks.

■ Technologies

- Python
- Jupyter Notebook
- pandas, numpy, matplotlib, seaborn, scipy

■ License

Open source. Feel free to use, modify, and distribute.

22. demogit

■ https://github.com/Muhammad-Muzammil-Shah/demogit

README

README not found.

23. EDA_OF_PAKISTAN_PROPERTIES

■ https://github.com/Muhammad-Muzammil-Shah/EDA_OF_PAKISTAN_PROPERTIES

README

EDA of Pakistan Properties

This project provides exploratory data analysis (EDA) on real estate data scraped from Zameen.com, Pakistan's leading property platform. The analysis is performed in a Jupyter notebook (`projectf.ipynb`) using Python data science libraries.

■ Dataset Overview

- **Source:** Zameen.com (scraped data)

- **Geography:** Pakistan

- **Time Period:** 02-04-2019 – 07-18-2019

- **Unit of Analysis:** Real estate listings

- **Variables:** property_id, location_id, page_url, property_type, price, location, city, province_name, latitude, longitude, baths, area, purpose, bedrooms, date_added, agency, agent, Area Type, Area Size, Area Category

---

■ Features & Analysis Steps

- Data loading and inspection

- Summary statistics and missing value analysis

- Data cleaning and reduction (removing unnecessary columns)

- Feature engineering and selection

- Univariate and multivariate EDA

- Visualization: histograms, boxplots, bar plots, heatmaps

- Correlation analysis of numerical features

■■ Installation

1. Clone the repository:

```bash

git clone https://github.com/Muhammad-Muzammil-Shah/EDA_OF_PAKISTAN_PROPERTIES.git
```

cd EDA_OF_PAKISTAN_PROPERTIES
```

2. (Optional) Create a virtual environment:

```bash
python -m venv venv
venv\Scripts\activate # Windows
# Or
source venv/bin/activate # macOS/Linux
```

3. Install dependencies:

```bash
pip install pandas numpy matplotlib seaborn
```

■ Usage

1. Open `projectf.ipynb` in Jupyter or Colab.
2. Run the notebook step-by-step to:
- Load and inspect the dataset
- Analyze missing values and summary statistics
- Clean and reduce data
- Engineer features for analysis
- Perform EDA with visualizations
- Explore correlations and insights
3. Review the interpretation and key findings at the end of the notebook.

■ Technologies

- Python
- Jupyter Notebook
- pandas, numpy, matplotlib, seaborn

■ License

Open source. Feel fr

... (truncated)

24. Educational-Q-A-Bot-for-Mathematics

■ https://github.com/Muhammad-Muzammil-Shah/Educational-Q-A-Bot-for-Mathematics

README

Educational Q&A Bot for Mathematics
```

A web-based chatbot that answers math questions step-by-step using the Groq API and Llama-3.1 model. Built with Flask, it provides clear, structured solutions and mathematical explanations for students and learners.

■ Features

- Ask math questions and get step-by-step solutions
- Clear formatting for mathematical expressions
- Web interface with Flask
- Uses Groq API (Llama-3.1-8b-instant)
- Supports LaTeX-style math rendering

■■ Installation

1. Clone the repository:

```bash
git clone https://github.com/Muhammad-Muzammil-Shah/Educational-Q-A-Bot-for-Mathematics.git
cd Educational-Q-A-Bot-for-Mathematics
```

2. Create a virtual environment (optional but recommended):

```bash
python -m venv venv
venv\Scripts\activate # Windows
# Or
source venv/bin/activate # macOS/Linux
```

3. Install dependencies:

```bash
pip install -r requirements.txt
```

4. Add your Groq API key to a `.env` file:

```env
GROQ_API_KEY=your_api_key_here
```

■ Usage

1. Start the Flask app:

```bash
python app.py
```

2. Open your browser and go to `http://localhost:5000`

3. Enter a math question and get a step-by-step solution.

■ Technologies

- Python
- Flask
- Groq API
- Llama-3.1-8b-instant
- HTML/CSS/JS (templates/static)

■ License

Open source. Feel free to use, modify, and distribute.

25. Emotion-Detection

■ https://github.com/Muhammad-Muzammil-Shah/Emotion-Detection

README

Emotion Detection

This project demonstrates emotion detection from text using NLP and machine learning techniques. It uses a labeled dataset (`Emotion_Dataset1.csv`) and a Jupyter notebook (`Emotion_dedection.ipynb`) for data preprocessing, feature engineering, and classification.

■ Features

- Tokenization, POS tagging, stemming, and lemmatization
- Frequency analysis of tokens, bigrams, trigrams, and n-grams
- Data cleaning and preprocessing with spaCy
- Feature extraction using TF-IDF
- Handling class imbalance with SMOTE
- Emotion classification using Random Forest
- Evaluation with classification report and confusion matrix heatmap

---

■■ Installation

1. Clone the repository:

```bash
git clone https://github.com/Muhammad-Muzammil-Shah/Emotion-Detection.git
cd Emotion-Detection
```

2. (Optional) Create a virtual environment:

```bash
python -m venv venv
```

venv\Scripts\activate # Windows

Or

source venv/bin/activate # macOS/Linux

```

3. Install dependencies:

```bash

pip install pandas numpy scikit-learn nltk spacy seaborn imbalanced-learn matplotlib

python -m spacy download en_core_web_sm

```

■ Usage

1. Open `Emotion_dedection.ipynb` in Jupyter or Colab.

2. Run the notebook step-by-step to:

- Preprocess and tokenize text data

- Engineer features and extract n-grams

- Train and evaluate emotion classifier

- Visualize results with confusion matrix

3. Review the interpretation and key findings at the end of the notebook.

■ Technologies

- Python

- Jupyter Notebook

- pandas, numpy, scikit-learn, nltk, spaCy, seaborn, imbalanced-learn, matplotlib

---

■ License

Open source. Feel free to use, modify, and distribute.

26. Etl_With-Python

■ https://github.com/Muhammad-Muzammil-Shah/Etl_With-Python

README

ETL With Python

This project demonstrates a basic Extract, Transform, Load (ETL) pipeline for financial data using Python. It scrapes bank data from Wikipedia, transforms market capitalization values to multiple currencies, and loads the results into CSV and SQLite database formats. Logging is included for process tracking.

■ Features

- Web scraping with BeautifulSoup

- Data extraction and transformation with pandas
```

- Currency conversion using exchange rates
- Logging of ETL steps
- Data loading to CSV and SQLite database
- SQL queries for analysis
- Jupyter notebook walkthrough

■■ Installation

1. Clone the repository:

```bash
git clone https://github.com/Muhammad-Muzammil-Shah/Etl_With-Python.git
cd Etl_With-Python
```

2. (Optional) Create a virtual environment:

```bash
python -m venv venv
venv\Scripts\activate # Windows
# Or
source venv/bin/activate # macOS/Linux
```

3. Install dependencies:

```bash
pip install pandas requests beautifulsoup4
```

■ Usage

1. Open `banks_project.ipynb` in Jupyter or Colab.
2. Run the notebook step-by-step to:
- Scrape and extract bank data
- Transform market cap values to GBP, EUR, INR
- Save results to CSV and SQLite database
- Run SQL queries for analysis
- View logs for ETL process
3. Review the interpretation and key findings at the end of the notebook.

■ Technologies

- Python
- Jupyter Notebook
- pandas, requests, beautifulsoup4, sqlite3

■ License

Open source. Feel free to use, modify, and distribute.

```bash

## 27. Excel-to-Word-Result-Generate-with-Python

■ https://github.com/Muhammad-Muzammil-Shah/Excel-to-Word-Result-Generate-with-Python

***README***

# Excel to Word Result Generate with Python

This project automates the generation of student mark sheets in Word format using data from an Excel/CSV file and a Word template. The Jupyter notebook (`Excel-to-Word-Result-Generate-with-Python.ipynb`) demonstrates how to read student data, fill a Word template for each student, and save individual or combined mark sheets.

---

## ■ Features

- Read student results from CSV (Excel-compatible)
- Fill Word template (`Resultcard.docx`) with student data
- Generate individual mark sheets for each student
- Optionally combine all mark sheets into a single Word file
- Easy-to-follow Jupyter notebook

---

## ■■ Installation

1. Clone the repository:

```bash
git clone https://github.com/Muhammad-Muzammil-Shah/Excel-to-Word-Result-Generate-with-Python.git
cd Excel-to-Word-Result-Generate-with-Python
```

2. (Optional) Create a virtual environment:

```bash
python -m venv venv
venv\Scripts\activate # Windows
# Or
source venv/bin/activate # macOS/Linux
```

3. Install dependencies:

```bash
pip install pandas docxtpl python-docx
```

■ Usage
```

1. Place your student data in `R.csv` and your Word template as `Resultcard.docx`.

2. Open `Excel-to-Word-Result-Generate-with-Python.ipynb` in Jupyter or Colab.

3. Run the notebook step-by-step to:

- Read student data

- Fill and save mark sheets for each student

- Optionally combine all mark sheets into one file

4. Review the generated Word files in the project folder.

■ Technologies

- Python

- Jupyter Notebook

- pandas, docxtpl, python-docx

■ License

Open source. Feel free to use, modify, and distribute.

28. Face-Detection-with-Python-using-OpenCV

■ https://github.com/Muhammad-Muzammil-Shah/Face-Detection-with-Python-using-OpenCV

README

Face Detection with Python using OpenCV

This project demonstrates face detection in images and videos using Python and OpenCV. The Jupyter notebook (`Face Detection.ipynb`) walks through using Haar Cascade classifiers for face detection, including real-time webcam and video file processing.

■ Features

- Face detection in images using Haar Cascade

- Face detection in video files and webcam streams

- Bounding box visualization for detected faces

- Step-by-step Jupyter notebook tutorial

- Example input image and video included

■■ Installation

1. Clone the repository:

```bash

git clone https://github.com/Muhammad-Muzammil-Shah/Face-Detection-with-Python-using-OpenCV.git

cd Face-Detection-with-Python-using-OpenCV

```

2. (Optional) Create a virtual environment:

```bash
python -m venv venv
venv\Scripts\activate # Windows
# Or
source venv/bin/activate # macOS/Linux
```

3. Install dependencies:

```bash
pip install opencv-python matplotlib
```

■ Usage

1. Place your input image as `input_image.jpg` and video as `videoplayback.mp4` in the project folder.
2. Open `Face Detection.ipynb` in Jupyter or Colab.
3. Run the notebook step-by-step to:
- Detect faces in images and videos
- Visualize results with bounding boxes
- Try real-time webcam face detection
4. Review the interpretation and key findings at the end of the notebook.

■ Technologies

- Python
- Jupyter Notebook
- OpenCV, matplotlib

■ License

Open source. Feel free to use, modify, and distribute.

Thankfully, OpenCV includes **pre-trained models** for face detection, eliminating the need for users to train models from scratch. OpenCV employs a machine learning approach called **Haar cascade classifiers** to identify faces and other objects in visual data efficiently.

29. generative-ai-for-beginners

■ https://github.com/Muhammad-Muzammil-Shah/generative-ai-for-beginners

README

(./images/repo-thumbnailv4-fixed.png?WT.mc_id=academic-105485-koreyst)

21 Lessons teaching everything you need to know to start building Generative AI applications

[

](https://github.com/microsoft/Generative-AI-For-Beginners/blob/master/LICENSE?WT.mc_id=academic-105485-k

oreyst)

[![GitHub contributors](https://img.shields.io/github/contributors/microsoft/Generative-AI-For-Beginners.svg)](https://GitHub.com/microsoft/Generative-AI-For-Beginners/graphs/contributors/?WT.mc_id=academic-105485-koreyst)

[![GitHub issues](https://img.shields.io/github/issues/microsoft/Generative-AI-For-Beginners.svg)](https://GitHub.com/microsoft/Generative-AI-For-Beginners/issues/?WT.mc_id=academic-105485-koreyst)

[![GitHub pull-requests](https://img.shields.io/github/issues-pr/microsoft/Generative-AI-For-Beginners.svg)](https://GitHub.com/microsoft/Generative-AI-For-Beginners/pulls/?WT.mc_id=academic-105485-koreyst)

[![PRs Welcome](https://img.shields.io/badge/PRs-welcome-brightgreen.svg?style=flat-square)](http://makeapullrequest.com?WT.mc_id=academic-105485-koreyst)

[![GitHub watchers](https://img.shields.io/github/watchers/microsoft/Generative-AI-For-Beginners.svg?style=social&label=Watch)](https://GitHub.com/microsoft/Generative-AI-For-Beginners/watchers/?WT.mc_id=academic-105485-koreyst)

[![GitHub forks](https://img.shields.io/github/forks/microsoft/Generative-AI-For-Beginners.svg?style=social&label=Fork)](https://GitHub.com/microsoft/Generative-AI-For-Beginners/network/?WT.mc_id=academic-105485-koreyst)

[![GitHub stars](https://img.shields.io/github/stars/microsoft/Generative-AI-For-Beginners.svg?style=social&label=Star)](https://GitHub.com/microsoft/Generative-AI-For-Beginners/stargazers/?WT.mc_id=academic-105485-koreyst)

[](https://aka.ms/genai-discord?WT.mc_id=academic-105485-koreyst)

■ Mul

... (truncated)

30. github-profile-readme-generator

■ https://github.com/Muhammad-Muzammil-Shah/github-profile-readme-generator

README

GitHub Profile README Generator

This project provides an easy way to create a GitHub profile README with the latest add-ons such as visitors count, GitHub stats, streak stats, and more. It features a minimal UI for entering profile details and generates a markdown README you can preview and use.

■ Features

- Uniform Dev and Social Icons

- Visitors Counter Badge

- GitHub Profile Stats Card

- GitHub Top Skills

- GitHub Streak Stats

- Dynamic Dev.to, Medium, and RSS Blogs (GitHub Actions)

- Wakatime Stats

- Buy Me A Coffee button

■■ Installation

1. Clone the repository:

```bash
git clone https://github.com/rahuldkjain/github-profile-readme-generator.git
cd github-profile-readme-generator
```

2. Install dependencies:

```bash
npm install
```

3. Run the app:

```bash
npm start
```

■ Usage

1. Open [GitHub Profile README Generator Demo](https://rahuldkjain.github.io/gh-profile-readme-generator)

2. Fill in your profile details and click "Generate README"

3. Preview and copy your markdown README

■ Technologies

- Gatsby
- Tailwind CSS
- GSAP

■ License

Open source. Feel free to use, modify, and distribute.

31. Humanize-Editorial-Web-App

■ https://github.com/Muhammad-Muzammil-Shah/Humanize-Editorial-Web-App

README

Humanize-Editorial-Web-App

A comprehensive Python-based system for converting AI-generated text into human-like writing that can bypass AI detection systems.

Features

- Multiple Humanization Techniques

- Advanced AI Models
- Production Ready API
- High Performance
- Batch Processing
- Customizable

Quick Start

See the full documentation in the main project README for setup and usage instructions.

32. Interview_system

■ https://github.com/Muhammad-Muzammil-Shah/Interview_system

README

Interview System

This project is a Streamlit app for recording interview videos, extracting audio, and generating transcripts using Whisper AI. It provides a simple interface for candidates to record, save, and transcribe their interview sessions.

■ Features

- Record interview videos in-browser
- Save video and extract audio automatically
- Generate transcripts using Whisper AI
- Display video, audio, and transcript in the app

■■ Installation

1. Clone the repository:

```bash
git clone https://github.com/Muhammad-Muzammil-Shah/Interview_system.git
cd Interview_system
```

2. (Optional) Create a virtual environment:

```bash
python -m venv venv
venv\Scripts\activate # Windows
# Or
source venv/bin/activate # macOS/Linux
```

3. Install dependencies:

```bash
pip install -r requirements.txt
```

```

---

## ■ Usage

1. Run the app:

```bash
streamlit run app1.py
```

2. Record your interview, click 'End and Process Video', and view the transcript.

■ Technologies

- Python
- Streamlit, streamlit-webrtc
- moviepy, whisper, pydub

■ License

Open source. Feel free to use, modify, and distribute.

33. Job-Email-Sender-AI

■ https://github.com/Muhammad-Muzammil-Shah/Job-Email-Sender-AI

README

AI Job Application Assistant

This is an end-to-end AI-powered application that generates and sends professional job application emails. It uses Streamlit for the frontend, OpenAI for email generation, and the Gmail API for sending emails.

Prerequisites

- Python 3.8+
- A Groq API Key
- An Outlook account (Email & Password)

Setup Instructions

1. **Install Dependencies**

```bash
pip install -r requirements.txt
```

2. **Environment Variables**

- Create a `.env` file in the root directory.
- Add your keys:

```
GROQ_API_KEY=gsk_your_groq_key_here
```
```

OUTLOOK_EMAIL=your_email@outlook.com

OUTLOOK_PASSWORD=your_password

```

- **Note**: If you have 2-Factor Authentication enabled on Outlook, you must generate an **App Password** and use that instead of your regular password.

## Running the Application

```bash

streamlit run app.py

```

Usage

1. The application will open in your browser.

2. Paste the **Job Description** into the text area.

3. Upload your **Resume (PDF)**.

4. Click **Next: Generate Email**.

- The AI will analyze the resume and job description.

5. Review the generated email.

6. Click **Send Email via Outlook**.

Troubleshooting

- **SMTP Error**: If sending fails, check your internet connection and ensure your Outlook credentials in `.env` are correct.

- **Blocked Sign-in**: Microsoft might block the sign-in if it looks suspicious. Check your email for security alerts or use an App Password.

34. Linear-Regression-Model

■ https://github.com/Muhammad-Muzammil-Shah/Linear-Regression-Model

README

Linear Regression Model

This project demonstrates linear regression using Python in a Jupyter notebook. It covers data loading, model training, prediction, and visualization of results for regression analysis.

■ Features

- Load and preprocess data

- Train a linear regression model

- Predict and evaluate results

- Visualize regression line and data points

- Step-by-step Jupyter notebook tutorial

■■ Installation

1. Clone the repository:
```

```bash
git clone https://github.com/Muhammad-Muzammil-Shah/Linear-Regression-Model.git
cd Linear-Regression-Model
```

2. (Optional) Create a virtual environment:

```bash
python -m venv venv
venv\Scripts\activate # Windows
# Or
source venv/bin/activate # macOS/Linux
```

3. Install dependencies:

```bash
pip install numpy pandas matplotlib scikit-learn
```

■ Usage

1. Open `Linear Regression Model.ipynb` in Jupyter or Colab.
2. Run the notebook step-by-step to:
- Load and preprocess data
- Train and evaluate the regression model
- Visualize results
3. Review the interpretation and key findings at the end of the notebook.

■ Technologies

- Python
- Jupyter Notebook
- numpy, pandas, matplotlib, scikit-learn

■ License

Open source. Feel free to use, modify, and distribute.

35. MAAAKS-ACADEMIC-PARK

■ https://github.com/Muhammad-Muzammil-Shah/MAAAKS-ACADEMIC-PARK

README

Preschool Website Template

A modern, responsive website template designed specifically for preschools, kindergartens, and early childhood education centers. This template provides a complete solution for educational institutions

looking to establish a professional online presence.

■ Features

- **Fully Responsive Design** - Works seamlessly on desktop, tablet, and mobile devices
- **Modern UI/UX** - Clean, child-friendly design with vibrant colors
- **Bootstrap Framework** - Built with Bootstrap for consistent styling and responsiveness
- **Multiple Pages** - Complete website structure with all essential pages
- **Interactive Elements** - Smooth animations and user-friendly navigation
- **Cross-Browser Compatible** - Works on all modern web browsers
- **Easy to Customize** - Well-organized code structure for easy modifications

■ Pages Included

- **Home Page** (`index.html`) - Welcome page with hero section and overview
- **About Us** (`about.html`) - Information about the preschool and its mission
- **Classes** (`classes.html`) - Available programs and age groups
- **Facilities** (`facility.html`) - School facilities and amenities
- **Teachers** (`team.html`) - Staff profiles and qualifications
- **Testimonials** (`testimonial.html`) - Parent and student feedback
- **Appointments** (`appointment.html`) - Schedule visits and meetings
- **Contact** (`contact.html`) - Contact information and location
- **Call to Action** (`call-to-action.html`) - Enrollment and inquiry section
- **404 Error** (`404.html`) - Custom error page

■■ Technologies Used

- **HTML5** - Semantic markup for better SEO and accessibility
- **CSS3** - Modern styling with custom properties and animations
- **Bootstrap 5** - Responsive grid system and components
- **JavaScript** - Interactive features and smooth animations
- **SCSS** - Enhanced CSS with variables and mixins

■ Libraries & Frameworks

- **Bootstrap** - Responsive framework
- **Animate.css** - CSS animations
- **Owl C

... (truncated)

36. Muhammad-Muzammil-Shah

■ https://github.com/Muhammad-Muzammil-Shah/Muhammad-Muzammil-Shah

README

<!-- ======= HERO HEADER ======= -->

<p align="center">

■ Hey there! I'm M. Muzammil Shah

AI Engineer passionate about **Microsoft Copilot Studio**, **Power Platform**, and building production-ready **AI agents**, **RAG pipelines**, and **automation workflows**.

■ **Core Expertise**

- Microsoft Copilot Studio: Multi-topic agents, triggers, flows, Adaptive Cards

- Power Platform: Power Automate, Power Apps, Dataverse integrations

- Azure AI: Semantic search, embeddings, Azure AI Search + Blob Storage

- AI/ML: LLMs, RAG (LangChain), NLP, Computer Vision (OpenCV/YOLO)

■ Currently focused on **enterprise-grade voice/chat agents** and **scalable knowledge base copilots**.

■ Tech Stack

<img src="https

... (truncated)

37. mychatbot

■ https://github.com/Muhammad-Muzammil-Shah/mychatbot

README

MyChatbot

This project demonstrates a simple chatbot implementation using Python. It provides a basic conversational interface and can be extended for custom use cases.

■ Features

- Basic chatbot logic in Python
- Easy to customize and extend
- Simple conversational flow

■■ Installation

1. Clone the repository:

```bash
git clone https://github.com/Muhammad-Muzammil-Shah/mychatbot.git
cd mychatbot
```

2. (Optional) Create a virtual environment:

```bash
python -m venv venv
venv\Scripts\activate # Windows
# Or
source venv/bin/activate # macOS/Linux
```

3. Install dependencies (if any):

```bash
pip install -r requirements.txt
```

■ Usage

1. Run the chatbot script:

```bash
python chatbot.py
```

2. Interact with the chatbot in the terminal or extend for web/app use.

■ Technologies

- Python

■ License

Open source. Feel free to use, modify, and distribute.

38. My_Portfolio

■ https://github.com/Muhammad-Muzammil-Shah/My_Portfolio

README

My Portfolio

This project is a personal portfolio website showcasing skills, projects, and experience. It is designed for easy customization and can be used to present your professional profile online.

■ Features

- Modern, responsive design

- Sections for skills, projects, experience, and contact

- Easy to customize content and style

■■ Installation

1. Clone the repository:

```bash

git clone https://github.com/Muhammad-Muzammil-Shah/My_Portfolio.git

cd My_Portfolio

```

2. Open `index.html` in your browser to view the portfolio.

■ Usage

1. Edit HTML/CSS files to add your own information, projects, and contact details.

2. Optionally deploy to GitHub Pages or any static hosting service.

■ Technologies

- HTML, CSS, JavaScript

■ License

Open source. Feel free to use, modify, and distribute.

39. NLP

■ https://github.com/Muhammad-Muzammil-Shah/NLP

README

NLP Projects

This repository contains Jupyter notebooks demonstrating various Natural Language Processing (NLP) techniques and workflows using Python. Topics include data cleaning, tokenization, stemming, lemmatization, vectorization, regular expressions, and pipelines.

■ Features

- Data cleaning and preprocessing
- Tokenization and stemming
- Lemmatization
- Count Vectorization and TF-IDF
- N-Grams
- Regular expressions for text analysis
- NLP pipelines
- Example datasets included

■■ Installation

1. Clone the repository:

```bash
git clone https://github.com/Muhammad-Muzammil-Shah/NLP.git
cd NLP
```

2. (Optional) Create a virtual environment:

```bash
python -m venv venv
venv\Scripts\activate # Windows
# Or
source venv/bin/activate # macOS/Linux
```

3. Install dependencies:

```bash
pip install numpy pandas scikit-learn nltk
```

■ Usage

1. Open any notebook (e.g., `Read_&_clean_Data-02.ipynb`, `Lemmatizer-06.ipynb`, `TF-IDF-09.ipynb`) in Jupyter or Colab.
2. Run the notebook step-by-step to:
   - Clean and preprocess text data
   - Apply NLP techniques
   - Visualize and interpret results
3. Review the interpretation and key findings at the end of each notebook.

■ Technologies

- Python

- Jupyter Notebook

- numpy, pandas, scikit-learn, nltk

■ License

Open source. Feel free to use, modify, and distribute.

40. PAK-PALESTINE-FLAGE-DETECTION-Using-Yolov11

■ https://github.com/Muhammad-Muzammil-Shah/PAK-PALESTINE-FLAGE-DETECTION-Using-Yolov11

README

PAK-PALESTINE Flag Detection Using YOLOv11

This project demonstrates flag detection for Pakistan and Palestine using the YOLOv11 object detection model. The Jupyter notebook walks through training, inference, and evaluation steps for flag detection in images.

■ Features

- Detect Pakistan and Palestine flags in images

- YOLOv11-based object detection workflow

- Step-by-step Jupyter notebook tutorial

- Example dataset included

■■ Installation

1. Clone the repository:

```bash

git clone https://github.com/Muhammad-Muzammil-Shah/PAK-PALESTINE-FLAGE-DETECTION-Using-Yolov11.git

cd PAK-PALESTINE-FLAGE-DETECTION-Using-Yolov11

```

2. (Optional) Create a virtual environment:

```bash

python -m venv venv

venv\Scripts\activate # Windows

# Or

source venv/bin/activate # macOS/Linux

```

3. Install dependencies:

```bash

pip install numpy pandas opencv-python torch yolov11

```

■ Usage

1. Open `PAK&PALESTINE FLAGE DETECTION M.Muzammil BAI-22S-028.ipynb` in Jupyter or Colab.

2. Run the notebook step-by-step to:

- Train and evaluate YOLOv11 model

- Detect flags in sample images

- Visualize detection results

3. Review the interpretation and key findings at the end of the notebook.

■ Technologies

- Python

- Jupyter Notebook

- YOLOv11, numpy, pandas, opencv-python, torch

■ License

Open source. Feel free to use, modify, and distribute.

41. Phishing-Email-Detection

■ https://github.com/Muhammad-Muzammil-Shah/Phishing-Email-Detection

README

■ ROBO Phishing Email Detector

Detect phishing emails using Natural Language Processing and a machine learning classifier trained on real-world email datasets. This project offers both single-email and bulk detection modes via a user-friendly **Streamlit web app**.

■ Features

* ■ **Phishing Detection**: Accurately classify emails as *Phishing* or *Not Phishing*.

* ■ **Model Evaluation**: Performance metrics including classification report, ROC curve, confusion matrix, and cross-validation accuracy.

* ■ **ML Model**: Trained using **Multinomial Naive Bayes** with **Bag of Words** feature extraction.

* ✉■ **Interactive UI**: Web interface built using **Streamlit** for real-time email detection.

* ■ **CSV Upload Support**: Detect phishing in bulk from uploaded email datasets.

■ Project Structure

```

■ phishing-email-detector/

■■■ phishing_email.csv # Dataset

■■■ spam_model.pkl # Trained ML model (MultinomialNB)
```

■■■ vectorizer.pkl # Trained CountVectorizer
■■■ streamlit_gui.py # Streamlit app
■■■ requirements.txt # Project dependencies
■■■ README.md # Project documentation
```
---
## ■ Dataset
* **Source**: `/content/phishing_email.csv`
* **Columns Used**:
* `text_combined` → Renamed to `email`
* `label` → Renamed to `Spam` (1 = phishing, 0 = not phishing)
---
## ■■ Preprocessing Steps
1. **Cleaning**: Removed NaN values, punctuation, and stopwords.
2. **Tokenization**: Counted tokens using `nltk.word_tokenize`.
3. **Stemming**: Applied `PorterStemmer`.
4. **Vectorization**: Used `CountVectorizer` to convert emails to numerical form.
---
## ■ Model Training
* **Algorithm**: Multinomial Naive Bayes
* **Evaluation**:
* Accuracy via 5-fold Cross-Validation
* Classification Report (Precision, Recall, F1-score)
* Confusion Matrix
* ROC Curve
---
## ■ Sample Prediction
```python
email = "Congratulations! You've won an i
... (truncated)

## 42. Python

■ https://github.com/Muhammad-Muzammil-Shah/Python

***README***

Basic Web Scraper Task Objective:

Objective:

Develop a web scraperto extractinformation from a website.

Details:

Write a scriptto scrape data (e.g., headlines, product prices)from a specified website using libraries like BeautifulSoup orScrapy.
Implementfunctionality to handle differentHTML structures and extractrelevant data.
Store the scraped data in a CSVfile or database.
Simple Chatbot Task Objective:
Build a basic rule-based chatbot
Details:
Create a command-line or GUI-based chatbotthatresponds to predefined keywords or phrases.
Implementfunctionality for handling differenttypes of userinputs and generating appropriate responses.
Optionally, use a library like ChatterBotfor more advanced features.
Linear Regression Model Task Objective:
Implement a simple linearregression model
Details:
Use a dataset(e.g., house prices, student scores)to predict a continuous variable.
Implementthe linearregression algorithm using Python libraries such asNumPy and scikit-learn.
Visualize the data and the regression line using Matplotlib.
Spam Email Classifier Task:
Create a spam email classifier usingNatural Language Processing (NLP).
Details:
Use a dataset oflabeled emails (spam or not spam).
Preprocess the text data (e.g.,tokenization,removing stop words) usingNLTK.
Implement aNaive Bayes classifier using scikit-learn to classify the emails.

## 43. py_to_dll

■ https://github.com/Muhammad-Muzammil-Shah/py_to_dll

***README***

# py_to_dll

## 44. Real_Estate_chatbot

■ https://github.com/Muhammad-Muzammil-Shah/Real_Estate_chatbot

### *README*

# ■ Real Estate Chatbot

Interact with your real estate CSV data using natural language! This Streamlit app leverages LangChain, HuggingFace embeddings, and Llama-2 for conversational Q&A on property datasets.

---

## ■ Features

* ■ **CSV Upload**: Upload any real estate CSV file and chat with your data.

* ■ **Conversational AI**: Uses Llama-2 (via CTransformers) for natural language answers.

* ■ **Semantic Search**: Embeddings via HuggingFace and FAISS for relevant document retrieval.

* ■ **Chat History**: Maintains session history for context-aware responses.

---

## ■■ Setup Instructions

1. **Python 3.8+ required**

2. Create a virtual environment:

```bash
python -m venv .venv
.\.venv\Scripts\activate
```

3. Install dependencies:

```bash
pip install -r req.txt
```

4. Launch the app:

```bash
streamlit run app.py
```

■ How It Works

* Upload a CSV file (e.g., `zameenkarachi.csv`).

* The app loads your data, creates embeddings, and builds a FAISS vector store.

* Ask questions about your data (e.g., "Show properties in Karachi under 1 crore").

* The chatbot uses Llama-2 to answer based on your CSV content.

■ Main Libraries Used

* Streamlit

* LangChain

* HuggingFace Embeddings

* FAISS

* CTransformers (Llama-2)

■ Project Structure

```
Real_Estate_chatbot/
■■■ app.py # Streamlit app
■■■ req.txt # Requirements
■■■ zmeenkarachi.csv # Example dataset
■■■ README.md # Documentation
```

■ Author

Syed Muhammad Muzammil Shah

■ License

Open-source under the MIT License.

45. Rename-Images

■ https://github.com/Muhammad-Muzammil-Shah/Rename-Images

README

■■ Image Renamer Script

This Python script renames image files in a specified directory to sequential numbers (e.g., `1.jpg`, `2.png`). Useful for organizing images with consistent naming for ML, web, or personal projects.

■ Features

* Detects and renames image files in a folder

* Sequentially numbers images based on alphabetical order

* Supports common formats (`.jpg`, `.png`, etc.)

■■ How It Works

```python
import os
def rename_images_to_numbers(directory, extension=".jpg"):
files = [f for f in os.listdir(directory) if f.endswith(extension)]
files.sort()
for index, filename in enumerate(files, start=1):
old_path = os.path.join(directory, filename)
```

new_name = f"{index}{extension}"

new_path = os.path.join(directory, new_name)

os.rename(old_path, new_path)

print(f"Renamed {len(files)} images successfully.")

Example usage

rename_images_to_numbers("pic", extension=".jpg")

```

---

## ■ Usage

1. Place your images in the `pic` folder (or change the path in the script).

2. Run the notebook or script:

```bash

python rename_images.py

```

3. Images will be renamed to `1.jpg`, `2.jpg`, ...

■■ Notes

* Backup your images before running the script.

* Renaming order is alphabetical.

■ License

MIT License. See `LICENSE` for details.

■ Contributions

Fork, create pull requests, or report issues!

46. School-Uniform-detection

■ https://github.com/Muhammad-Muzammil-Shah/School-Uniform-detection

README

School-Uniform-detection

Overview

This project demonstrates object detection for school uniforms using YOLOv5 and YOLO11 models. It includes Jupyter notebooks for training, evaluating, and running inference on custom datasets, with step-by-step instructions and code examples.

How It Works

- **Custom Dataset Preparation:** Uses Roboflow to collect, annotate, and export images for training.

- **Model Training:** Notebooks show how to train YOLOv5 and YOLO11 models on the dataset, including hyperparameter tuning and augmentation.

- **Evaluation:** Visualizes training results, confusion matrix, and sample predictions.
```

- **Inference:** Runs object detection on images and videos, saving results for review.

- **Deployment:** Instructions for deploying models using Roboflow and running inference on edge devices.

Main Files

- `train_yolov5_object_detection_on_custom_data.ipynb`: Step-by-step guide for YOLOv5 training and evaluation.

- `YOLO11_Uniform_Detection.ipynb`: Step-by-step guide for YOLO11 training and inference.

- `MergedImages.png`: Example image for inference.

- `notebooks/`: Additional notebooks and resources.

Usage

1. Prepare your dataset using Roboflow and export in YOLO format.

2. Follow the notebooks to train and evaluate your model.

3. Run inference on new images or videos to detect uniforms.

4. Deploy your model using Roboflow or on local/edge devices.

Technologies

- Python, Jupyter Notebook

- YOLOv5, YOLO11 (Ultralytics)

- Roboflow (dataset management)

This repo is ideal for learning and experimenting with custom object detection workflows.

47. Simple-Chatbot

■ https://github.com/Muhammad-Muzammil-Shah/Simple-Chatbot

README

README not found.

48. spam_or_not_spam

■ https://github.com/Muhammad-Muzammil-Shah/spam_or_not_spam

README

Spam or Not Spam

This project demonstrates spam detection using Python and machine learning. It includes a Jupyter notebook for data analysis, model training, and evaluation on a labeled email dataset.

■ Features

- Load and preprocess email dataset

- Train and evaluate spam detection model

- Visualize results and metrics

- Step-by-step Jupyter notebook tutorial

■■ Installation

1. Clone the repository:

```bash
git clone https://github.com/Muhammad-Muzammil-Shah/spam_or_not_spam.git
cd spam_or_not_spam
```

2. (Optional) Create a virtual environment:

```bash
python -m venv venv
venv\Scripts\activate # Windows
# Or
source venv/bin/activate # macOS/Linux
```

3. Install dependencies:

```bash
pip install numpy pandas scikit-learn matplotlib
```

■ Usage

1. Open `spam_or_not_spam.ipynb` in Jupyter or Colab.
2. Run the notebook step-by-step to:
- Load and preprocess the dataset
- Train and evaluate the spam detection model
- Visualize results
3. Review the interpretation and key findings at the end of the notebook.

■ Technologies

- Python
- Jupyter Notebook
- numpy, pandas, scikit-learn, matplotlib

■ License

Open source. Feel free to use, modify, and distribute.

49. syed0

■ https://github.com/Muhammad-Muzammil-Shah/syed0

README

README not found.

50. web-scraping-with-Python

■ https://github.com/Muhammad-Muzammil-Shah/web-scraping-with-Python

README

Web Scraping with Python

This project demonstrates web scraping using Python. It includes a Jupyter notebook for scraping headlines, saving results to CSV, and analyzing the data.

■ Features

- Scrape headlines from web pages

- Save scraped data to CSV files

- Analyze and visualize results

- Step-by-step Jupyter notebook tutorial

■■ Installation

1. Clone the repository:

```bash

git clone https://github.com/Muhammad-Muzammil-Shah/web-scraping-with-Python.git

cd web-scraping-with-Python

```

2. (Optional) Create a virtual environment:

```bash

python -m venv venv

venv\Scripts\activate # Windows

# Or

source venv/bin/activate # macOS/Linux

```

3. Install dependencies:

```bash

pip install requests beautifulsoup4 pandas

```

■ Usage

1. Open `web scraping with Python.ipynb` in Jupyter or Colab.

2. Run the notebook step-by-step to:

- Scrape headlines from web pages

- Save and analyze results

3. Review the interpretation and key findings at the end of the notebook.

■ Technologies

- Python

- Jupyter Notebook

- requests, beautifulsoup4, pandas

■ License

Open source. Feel free to use, modify, and distribute.

51. Winter-Internship-Program-at-ITSOLERA-Pvt.-Ltd

■ https://github.com/Muhammad-Muzammil-Shah/Winter-Internship-Program-at-ITSOLERA-Pvt.-Ltd

README

README not found.

52. Word-file-Handling

■ https://github.com/Muhammad-Muzammil-Shah/Word-file-Handling

README

Word-file-Handling