A project report on

ANALYTIBOT: UNLEASHING LLM, LANGCHAIN, AI-ML AND OPENAI FOR THE ANALYSIS OF RESEARCH ARTICLES.

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering with Specialization in Cyber Physical Systems

By

ALEKYA RAMANI NALAGANDLA (20BPS1097)

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April, 2024

A project report on

ANALYTIBOT: UNLEASHING LLM, LANGCHAIN, AI-ML AND OPENAI FOR THE ANALYSIS OF RESEARCH ARTICLES.

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering with Specialization in Cyber Physical Systems

By

ALEKYA RAMANI NALAGANDLA (20BPS1097)

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April, 2024

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

DECLARATION

I hereby declare that the thesis entitled "**ANALYTIBOT: UNLEASHING LLM, LANGCHAIN, AI-ML AND OPENAI FOR THE ANALYSIS OF RESEARCH ARTICLES**" submitted by me, for the award of the degree of Bachelor of Technology in Computer Science and Engineering with Specialization in Cyber Physical Systems, Vellore Institute of Technology, Chennai is a record of bonafide work carried out by me under the supervision of Dr. T S Pradeep Kumar.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Chennai

Date:

Atekya · N

Signature of the Candidate

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

School of Computer Science and Engineering

CERTIFICATE

This is to certify that the report entitled **"AnalytiBot: Unleashing LLM, Langchain, AI-ML and OpenAI for the Research Analaysis of Articles"** is prepared and submitted by Alekya Ramani Nalagandla (20BPS1097) to Vellore Institute of Technology, Chennai, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science and Engineering with Specialization in Cyber Physical Systems** programme is a bonafide record carried out under my guidance. The project fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Signature of the Guide:

Name: Dr./Prof. T S Pradeep Kumar

Date:

Signature of the Internal Examiner	Signature of the External Examiner
Name:	Name:
Date:	Date:

Approved by the Head of Department,
B.Tech. CSE with Specialization in Cyber Physical Systems

Name: Dr. Renuka Devi S

Date:

ABSTRACT

The amalgamation of AI and ML has led to the discovery of today's boom - a world driven by robo-advisory, and a part of this is chatbots. As the amount of academic research increases rapidly, there’s a critical need for a more effective way to analyze it. This study introduces the AnalytiBot, a chatbot specially designed for researchers to streamline the analysis of unstructured data sources such as research papers (PDFs) and webpages (URLs) for researchers. AnalytiBot is a result of combination of powerful technologies like Langchain: facilitates the analysis of textual relationships within the data, LLM (Large Language Model): to process and analyze natural langauge found from the data sources, FAISS (Facebook AI Similarity Search): to search through vast amount of data, and OpenAI: creating embeddings, enables us access to language models. By integrating these technologies, the bot enables seamless interaction with and gain valuable insights from unstructured data sources. This significantly reduces the time researchers spend manually sifting through information, allowing them to dedicate their efforts to higher-level analysis and discovery.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Dr Pradeep Kumar T S, Professor in Computer Science and Engineer at VIT Chennai, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, for his constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavor. My association with him is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the fields of Artificial Intelligence and Machine Learning.

It is with gratitude that I would like to extend my thanks to the visionary leader Dr. G. Viswanathan our Honorable Chancellor, Mr. Sankar Viswanathan, Dr. Sekar Viswanathan, Dr. G V Selvam Vice Presidents, Dr. Sandhya Pentareddy, Executive Director, Ms. Kadhambari S. Viswanathan, Assistant Vice-President, Dr. V. S. Kanchana Bhaaskaran Vice-Chancellor, Dr. T. Thyagarajan Pro-Vice Chancellor, VIT Chennai and Dr. P. K. Manoharan, Additional Registrar for providing an exceptional working environment and inspiring all of us during the tenure of the course.

Special mention to Dr. Ganesan R, Dean, Dr. Parvathi R, Associate Dean Academics, Dr. Geetha S, Associate Dean Research, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai for spending their valuable time and efforts in sharing their knowledge and for helping us in every aspect.

In jubilant state, I express ingeniously my whole-hearted thanks to Dr. Renuka Devi S, Head of the Department, B.Tech. CSE with Specialization in Cyber Physical Systems and the Project Coordinators for their valuable support and encouragement to take up and complete the thesis.

My sincere thanks to all the faculties and staffs at Vellore Institute of Technology, Chennai who helped me acquire the requisite knowledge. I would like to thank my parents for their support. It is indeed a pleasure to thank my friends who encouraged me to take up and complete this task.

Place: Chennai

Date:

Alekya Ramani Nalagandla

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

1. AI	ARTIFICIAL INTELLIGENCE
2. ML	MACHINE LEARNING
3. FAISS	FACEBOOK AI SIMILARITY SEARCH
4. LLM	LARGE LANGUAGE MODEL
5. URL	UNIFORM RESOURCE LOCATOR
6. PDF	PORTABLE DOCUMENT FORMAT
7. GPT	GRENERATING PRETRAINING TRANSFORMER
8. VAE	VARIATIONAL AUTO ENCODER
9. GEMINI	GAGAN ENABLED MARINER'S INSTRUMENT FOR NAVIGATION AND INFORMATION
10. NLP	NATURAL LANGUAGE PROCESSING
11. CSV	COMMA SEPERATED VALUES

Chapter 1

Introduction

1.1 INTRODUCTION

In recent years, the amount of academic research that is being done has undergone a huge transformation, both in terms of the complexity and the volume. As of 2022, the number of published research papers per year has increased to about 5.14 million including essays, conference papers, book chapters, etc. This increase in volume has its own pros and cons, while it provides opportunities for betterment in the field of research through the vast amount of knowledge, the sheer volume of data and information poses an obstacle to accurate and effective analysis.

We live in an era where the knowledge that comes through the information plays a crucial role in any field, from a small business strategy to a very complex medical research. Traditionally, the process of going through the unstructured data, including research papers, news articles, etc. is considered as a very daunting and heavy manual labour which not only consumes a lot of time but also reduces the efficiency, which in return poses a threat of oversight and error.

Luckily, we have AI and ML, which not only provides us with innovative solutions for such problems but also produces accurate results. Through the integration of some of the most powerful technologies like LLM, OpenAI, Langchain and FAISS, I made a chatbot that's specifically tailored for the needs of researchers. AnalytiBot: a mini chatbot that helps with and provides a better way of interaction between the researchers and the unstructured data sources including URLs of the real time articles and research papers in the form of PDFs, Or datasets in the form of CSV files etc. while providing accurate answers for a given prompt with retaining the history of the questions and answers of a session.

1.2 OVERVIEW OF ANALYTIBOT

AnalytiBot, it represents the advancement made in the field of academic research, by offering a great solution for the challenges faced by the researchers in navigating and finding insights from a wide range of data sources. At its core, the AnalytiBot is a chatbot which is specifically crafted to meet and cater the unique needs of researchers, giving them a highly interactive platform to gain valuable insights and analysis from the huge amount of data that is processed and stored in the form of vector embedding's through the data sources.

At the core of the Chabot, it has multiple cutting-edge tools including LLM, Langchain, Streamlit, OpenAI, etc., which are integrated together to give a seamless experience and maximize the effectiveness and utility of the AnalytiBot.

Key Features:

1. **Interacting through Natural Language**: Whether it be summarizing the article or finding main insights, users can easily interact with the AnalytiBot using normal familiar and natual language, they don't need any kind of special training for interacting with the bot.
2. **Data Extraction**: It has the capability of extracting unstructured data from a wide range of sources, including URLs, CSV files and PDFs.
3. **AI-driven Answer Generation**: Through the power of Large Language Models, FAISS and OpenAI, AnalytiBot uses Keyword based search mechanisms, providing analysis and identifying trends to extract knowledge and intelligence that can used in real world. Additionally it also learns from what is being produced, i.e., a feedback learning model.
4. **Limit less Research**: While the existing chatbots has a limit or needs to be purchased for uploading pdfs, AnalytiBot has a limit less opportunities for research.

1.3 CHALLENGES PRESENT

In the field of research, a lot of academia face huge number of challenges that might affect and reduce the efficiency of the analysis. And most of these challenges stems from the volume of publications done today.

1. **Overwhelmingly high volume of Research Publications:**

Currently there are a lot of fields of study in which research work is being done, Apart from just academic universities which take up 15% of published research works, remaining 85% of that are published are through individual or other organisations, which shows the huge range and importance of research work that is being done, and this is increasing year by year. As the sheer volume of publications increases, it is becoming a very strenuous, time consuming and a daunting task to go through and identify required knowledge from this immense information. This in turn makes it challenging for researchers to keep up with.

2. **Time Consuming:**

Manual research is time consuming, from sifting through the data of research publications to noting down the required and relevant data points, not only is this a time consuming task but also it requires a certain level of expertise. Moreover, the repetitive process of going through and noting down the important and key findings is a very strenuous job, which might cause fatigue and is prone to errors while compromising the reliability, integrity and accuracy if the outcomes.

3. **Existing Difficulty in extracting insights from unstructured data sources:**

Unstructured data sources, like PDFs, and URLs are not easy to deal with. While we have exiting Chatbot's like ChatGPT and GEMINI, they don't provide us with the ability to go through an URL and get insights from them due to restrictions and various policies, similarly even though they have the ability to load the PDF and get insights from them, there is a limit up to of size and the number of documents uploaded. This in return poses a challenge of not being able to get insights from URLs and PDFs.

4. **Limitations of Existing Methods and Tools:**

Existing tools and methods like Chatbot's or manual research doesn't facilitate the analysis for the huge amount of data, i.e., there is a word limit up to which you can provide to the Chatbot and ask for by prompting questions and when it comes to manual research, it is prone to error. While you may be able to use these methods and tools for small-scale data, it is an ineffective way for dealing with large-scale data.

1.4 PROBLEM STATEMENT

While a lot of people face challenges when researching for a problem, the huge amount of publications and the data that is produced yearly makes it even harder for manual research and is highly prone to error and fatigue. And the existing tools and methods like chatgpt and Gemini might be useful for the analysis of small scale data but when it comes to huge large scale data which is what we generally deal with when going through various news articles and research works, they don't come handy. To overcome these ineffective ways, we introduce a better way which not only deals with large scale data but also gives accurate answers.

1.5 OBJECTIVES

1. **Provide an Easy-to-Use Interface:** AnalytiBot should have an easy-to-use interface that lets researchers to interact with it in an seamless and intuitive way, while promoting engagement that goes smoothly.
2. **Put Natural Language Processing (NLP) Skills into Practice:** Enhance the experience and accessibility of the model by integrating different algorithmic techniques which allows AnalytiBot to understand and interpret researchers question without any hurdle while saving the users the time of learning something new for interaction.

3. **Allow for the extraction of data from PDFs and URLs:** Providing algorithms and techniques to extract structured data from unstructured data sources, like URLs and PDFs, so that the data can be accurately retrieved and used.
4. **Use AI Technologies to Generate Insights:** AnalytiBot can detect different patterns, trends, and correlations in the extracted data by using AI and ML technologies, including Large Language Models (LLM) and OpenAI.
5. **Guarantee the scaling and Performance:** AnalytiBot is to be designed with scalability and performance in mind, which enables it to manage the large-scale data volumes and meet expanding user needs.
6. **Integrate Technologies and APIs:** To improve AnalytiBot's working and capacity for better retrieval, combine it with external technologies and APIs like Langchain, LLM, OpenAI and Facebook AI Similarity Search (FAISS).

1.6 SCOPE OF THE PROJECT

The project mainly lies with in the fields of Artificial Intelligence, Web Scrapping, and Natural Language Processing. I aim to achieve:

1. A seamless user interface for interaction between user and data sources.
2. A chatbot model that can successfully loads and extract the data from the provided unstructured data sources through web scrapping.
3. A chatbot that provides accurate answers with the sources of the answers while retaining and learning from the previously prompted questions and answers.

Chapter 2

Background

2.1INTRODUCTION

The earliest days of artificial intelligence study were when chatbots first emerged, with pioneering research published as early as the 1960s. But it wasn't until the development of cutting-edge machine learning methods and the accessibility of enormous volumes of data that chatbots started to show major advances in terms of usage and intelligence. The debut of ELIZA by Joseph Weizenbaum in the 1960s was an important turning point in the advancement of chatbots[1]. A simple program of natural language processing called ELIZA identified key phrases and produced responses that were pre-programmed to mimic an interaction with a psychotherapist. Although ELIZA was basic by today's norms, it pioneered the way for later chatbot versions [1].

In the modern era, DL algorithms and neural network algorithms have transformed chatbot technology, causing an upheaval in the field. Apple debuted Siri in 2011, an artificially intelligent assistant that can do tasks like sending messages, putting reminders, and responding to inquiries while also comprehending conversational commands [2]. With its initial release, Siri represented a major advancement in chatbot technology and demonstrated how artificial intelligence-driven interfaces for conversations may improve user experiences with digital devices.

A large number of chatbot systems each with its distinct characteristics and functions, appeared after Siri. Rivalling for leadership in the rapidly developing artificial intelligent assistant space, Amazon debuted Alexa, Google debuted Google Assistant, and Microsoft debuted Cortana.

Apart from artificially intelligent assistants, chatbots have become increasingly common in customer care usage, providing companies with an affordable way to perform repetitive queries. Customer support operations were transformed by platforms such as IBM Watson Assistant, Salesforce Einstein, and LivePerson, which helped companies to use AI-powered chatbot assistants that could quickly and effectively handle intricate customer interactions.

To put it simply, the advancement of chatbot technology has taken an upward trajectory of constant growth and development, driven by breakthroughs in AI, Ml, and NLP. While current chatbot models have improved user experiences and automated tasks, much can be done to enhance scalability, agility, and natural language understanding. Keeping this in light, the AnalytiBot is the result of an attempt to advance chatbot technology and solve the particular difficulties encountered and faced by researchers in navigating through the complexities of scholarly research.

2.2LITERATURE SURVEY

The development and high use of chatbot technology across a different and vast range of industries has been fueled by NLP and AI. Adamopoulou and Moussiades (2020) [3] provided a thorough overview of chatbot technology, highlighting the chatbot technologies application in marketing, education, customer assistance, cultural heritage, healthcare, and entertainment. In addition to discussing how societal prejudices affect chatbot design, the study also gives us a classification system based on the number of variables mentioned. It also discusses the general architecture and infrastructure of chatbots.

A critical evaluation of state-of-the-art chatbot designs and implementations is provided by Luo et al. (2022) [4], who look at developments in technology and useful implementations across an array of sectors. They highlight areas in need of further investigation and present fresh concepts for the creation and use of chatbots.

Fotheringham and Wiles (2023) [5] look into how artificial intelligence chatbot aides influence company value, especially for customer service positions. Their studies imply that enterprises using chatbot assistants for customer service have higher returns on stock, and that sentiment among investors looks to be affected by the machines' similarity to actual humans.

The development and use of chatbots makes use of large language models (LLMs) such as LangChain, as covered in works by Pandya and Holia (2023) and Jeong (2023) [6]. They show the effectiveness with which LLM-powered chatbots can be used to simplify customer assistance while improving generative AI services, accordingly

The use of LLMs for automated form filling is examined by Bucur (2023) [7], who additionally demonstrates how successfully LLM integration with information repositories may enhance the process of request form completion.

Mansurova et al. (2023) [8] propose incorporating ChatGPT with outside expertise sources to improve LLMs like ChatGPT and increase the way they perform in domain- particular tasks.

.Using tiny-LLMs connected with LangChain, Basit et al. (2024) [9] provide MedAide, an on-premise healthcare chatbot for effective edge-based medical diagnostics.

Nischal et al. (2020) [10] provide a system of chatbots for the banking sector designed to improve interaction with customers and money-related data access using LangChain and LLMs.

The use of generative AI services employing LLMs in commercial operations contexts is covered by Jeong (2023) [11] while focusing on document incorporation and fine-tuning techniques.

In the end, Bratić et al. (2024) [12] provide a hybrid approach that brings together a transformer architecture for faster educational content retrieval along with an LLM/chatbot user interface for correct replies from a centrally maintained database.

Collectively, these examples show the diverse applications of chatbot technology across several sectors and the transformative power of AI-powered conversational agents powered by LLMs and state-of-the-art frameworks such as LangChain.

Chapter 3

Proposed System

3.1 INTRODUCTION

I'll be implementing a mini-Chabot named AnalytiBot which takes in, loads, and extracts the data from unstructured data sources like PDFs, CSV files and URLs using LangChain technology, after loading the data, it is sent into text splitter for splitting the data into n equal chunks using the recursive text splitter module. These text chunks are sent into a language model which embeds the text to a vector and they are generally called vector embeddings and this is done through the OpenAI key which enables the model to access the language models of the OpenAI. Once the embedding is done they are sent into the vector database which in our case is a pickle file. We use a pickle file for faster data transfers. After this, the user is asked for the prompt and once a prompt is given it is also sent for embedding and the prompts vector embedding is compared with the chunks of embeddings present in the vector database using FAISS, what FAISS generally does is a similarity search it searches for chunks which has similar embedding to the prompt chunk and send it to the LLM and LLM learns and trains the model using these prompt and answer embeddings and send the appropriate output and the source from which the output is produced based on the temperature given for the Open AI model, i.e., the lesser the temperature the straight forward the answer is while the more the temperature the more creative the answer is. As the prompts and answers are generated the Chatbot learns more about the data i.e., it has a feedback loop going inside which enables the model to learn and train based on the prompts as well. Similarly when we load a CSV file it loads the data and presents us with various analysis models and statistical tests along woth all the variab;es present in the dataset for performiung and finding trends and patterns present in the depths of data. I also introduced one more module which helps with retaining the questions and answers on the user interface until the session is completed for better view and proper analysis of the user and this is done using the streamlit session module. All of this is implemented seamlessly through the streamlit web page which is implemented using the streamlit package.

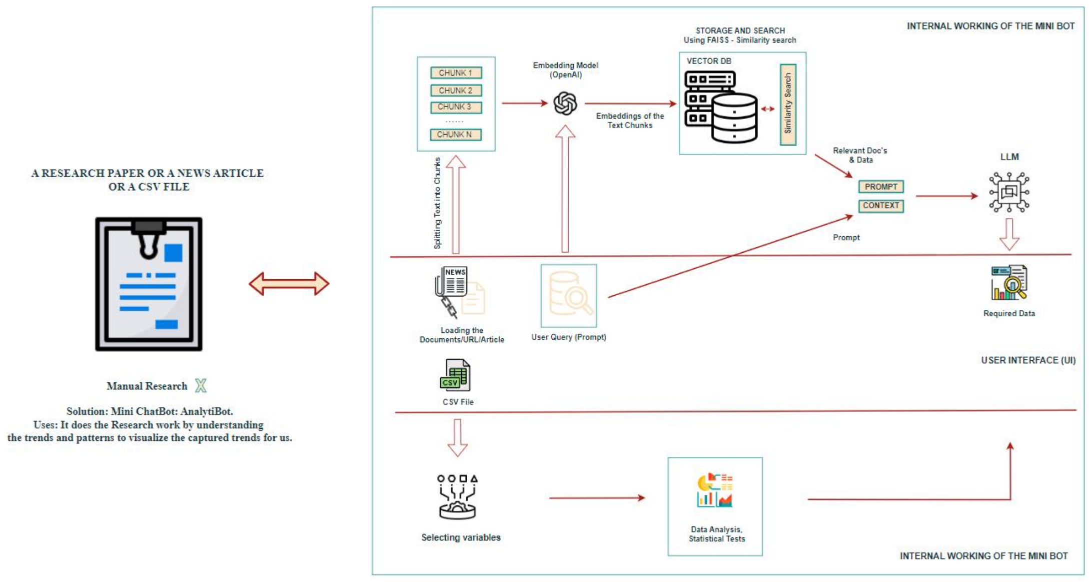

A RESEARCH PAPER OR A NEWS ARTICLE OR A CSV FILE
Manual Research
Solution: Mini ChatBot: AnalytiBot.
Uses: It does the Research work by understanding the trends and patterns to visualize the captured trends for us.
INTERNAL WORKING OF THE MINI BOT
STORAGE AND SEARCH Using FAISS - Similarity search
CHUNK 1
CHUNK 2
CHUNK 3
CHUNK N
Embedding Model (OpenAI)
Embeddings of the Text Chunks
VECTOR DB
Similarity Search
Splitting Text into Chunks
Relevant Doc's & Data
PROMPT
CONTEXT
Prompt
LLM
NEWS
Loading the Documents/URL/Article
User Query (Prompt)
Required Data
USER INTERFACE (UI)
CSV
CSV File
Selecting variables
Data Analysis, Statistical Tests
INTERNAL WORKING OF THE MINI BOT

Figure 1: Architecture of the AnalytiBot

3.2 MODULES

In this section we'll be discussing about different modules involved in the project, in in-depth explanation of each module and the working of each module will be thoroughly explained.

3.2.1. Data Input Module

Packages Used:

- Streamlit: For creating the web interface.
- Pandas: For handling dataframes (CSV file data).
- os: For interacting with the operating system.
- tempfile: For creating temporary files.
- PyPDFLoader (from langchain.document_loaders): For loading PDF files.
- UnstructuredURLLoader (from langchain.document_loaders): For loading data from URLs.

This module provides users with options to input data via URLs, PDF files, or CSV files through a Streamlit interface. Depending on the user's choice, the corresponding data loading mechanism is activated.

Figure 2: Loading the data from data sources

3.2.2. Data Processing Module

Packages Used:

- LangChain: A custom package for natural language processing and document handling.
- OpenAIEmbeddings (from langchain.embeddings): For converting text data into numerical vectors.
- RecursiveCharacterTextSplitter (from langchain.text_splitter): For splitting text data into chunks.
- FAISS (from langchain.vectorstores): For efficient storage and retrieval of vector embeddings.
- pickle: For serializing and deserializing Python objects.

This module processes the data loaded from URLs or PDF files. It involves splitting the text into smaller chunks, converting the text into numerical vectors (embeddings), and storing these embeddings in a vector database (FAISS index) using pickle for faster retrieval.

3.2.3. Text Processing Module

Here we do the text splitting using the text_splitter module from the langchain package, here we will be using the recursive text splitter library as it is the most efficient way for storing and gaining the maximum utilization of the chunk size and for our project.

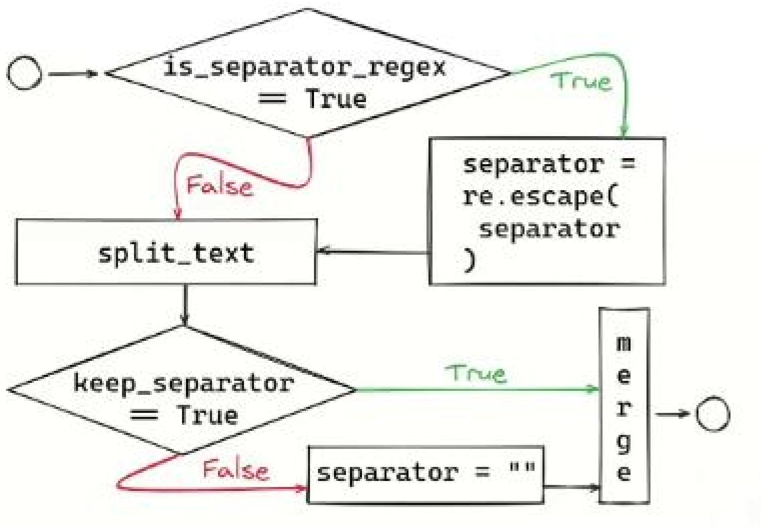
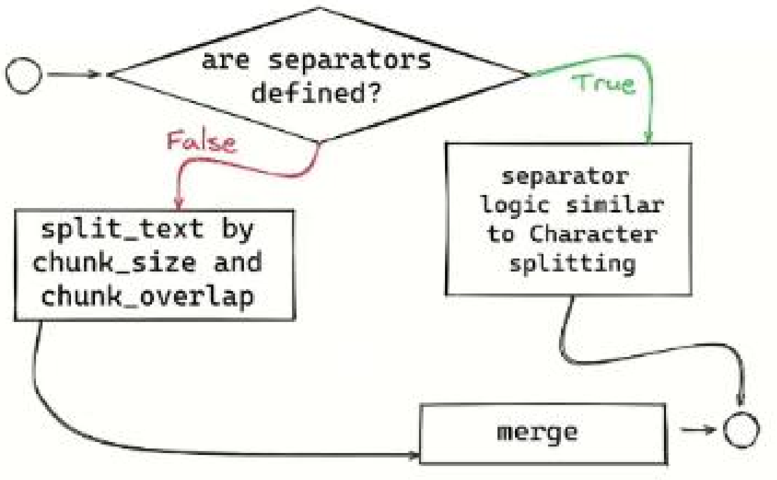


Figure 3: Flow diagram for the algorithm for Character splitting



are separators defined?
True
False
split_text by chunk_size and chunk_overlap
separator logic similar to Character splitting
merge

Figure 4: Flow diagram for the algorithm for Recursive Character splitting

Here you can see that we are asking for splitting near a space, or a comma, or a period or enter. This makes it more efficient and helps with better building of text chunks.

3.2.4. Vector Embedding

Vector embedding is one of the most crucial steps. Here in this module the textual data is converted into numerical vector form using the embedding language model of OpenAI language models.

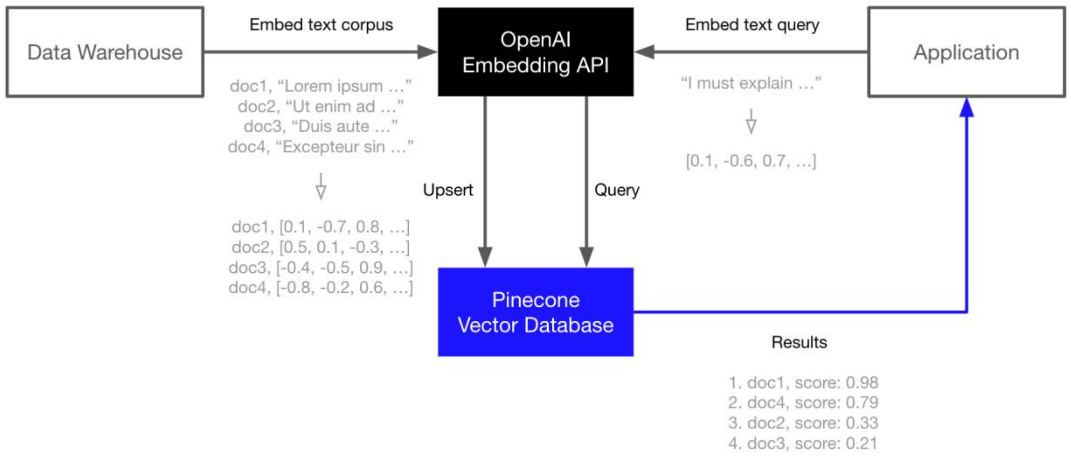


Figure 5: Architecture flow of how the vector embeddings are made

Steps involved in the Vector Embedding process:

1. **Access the OpenAI API:** Obtain an OpenAI API key by logging in and making an account (there will be a trial of three months for the newly created account), and this OpenAI API secret key grants access to OpenAI's language models and embedding abilities.

2. **Input the text:** Load the textual data into the model that you want to convert into embeddings. This can include chunks of data like sentences, paragraphs, or entire documents.

3. **Request to OpenAI API:** Sending a request to the OpenAI API, providing text input and any extra parameters if required for the embedding.

4. **Text Processing:** OpenAI's language model processes the input text, analyzing its semantic meaning, context, and relationships between words.

5. **Vector Embedding Generation:** The processed text is converted into a high-dimensional numerical vector representation, generally referred to as vector embedding. This embedding captures the semantic information encoded in the text, enabling mathematical operations and comparisons.

6. **Receive Embeddings:** Upon processing the text input, the OpenAI API returns the corresponding vector embeddings as output. These embeddings can be used for various applications, such as similarity search, content recommendation, or natural language understanding tasks.

Here an important point to remember is the trial version only lasts for 3 months and a voucher of $5 is given to use, with every embedding done and every language model usage there will be a significant reduction in the amount of remaining voucher money that can be used.

3.2.5. FAISS – Facebook AI Similarity Search

Here the entire vector embeddings stored in the vector database which in our case would be the .pkl file, will be compared to prompt vector embedding to find the most similar chunk of vector embeddings as the answer.

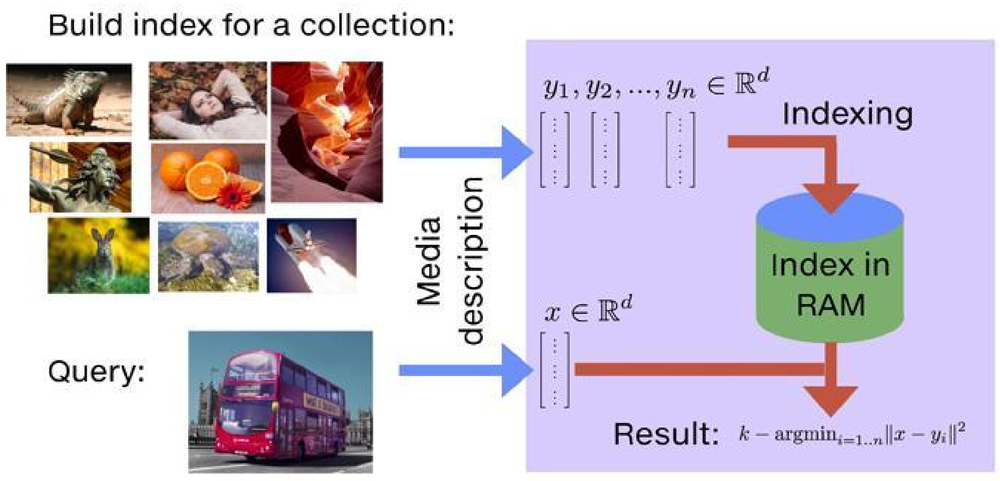

Build index for a collection:
$y_1, y_2, \ldots, y_n \in \mathbb{R}^d$
Indexing
Media description
Index in RAM
$x \in \mathbb{R}^d$
Query:
Result: $k - \mathrm{argmin}_{i=1..n} \|x - y_i\|^2$

Figure 7: Flow diagram for the FAISS architecture.

Here as you can see in the image we are building a vector database based on the loaded data by converting them into embeddings and then a query prompt is loaded and converted to embedding and sent into the RAM database which in our case is the pickle file for finding the most similar vector embedding from the chunks of already loaded data's embeddings.

3.2.6. LLM Module

Here we train the LLM model by giving it a chat/completion model of the OpenAI language models which can be used for chat-based models like chatbots, in this category we have 2 models existing, first is **gpt-3.5-**turbo and the second is **gpt-3.5-turbo-16k.** using these two models we'll be able to access and use the in-built functionalities and abilities of the OpenAI's chat language models.

Packages Used:

- LangChain: A custom package for natural language processing and document handling.
- ChatOpenAI (from langchain.chat_models): For generating responses to user queries using language models.

The LLM, powered by the ChatOpenAI model from LangChain, leverages advanced language models (such as GPT-3.5) to comprehend user queries and generate relevant responses. It utilizes state-of-the-art natural language processing techniques to provide accurate and contextually appropriate answers to user inquiries.

The LLM interacts with the processed data, retrieving relevant information from the vector database (FAISS index) and incorporating it into its responses. It adapts its answers based on the temperature parameter, controlling the level of creativity and relevance in the generated responses.

3.2.7. Retention of all Prompts and Answers

Packages Used:

- Streamlit: A Python library used for creating web applications with interactive user interfaces.
- Pandas: A powerful data manipulation library.

The Data Retention Module, integrated seamlessly within AnalytiBot's user interface using Streamlit, maintains a record of all questions asked and their corresponding answers throughout the session. This feature enhances user experience by providing a convenient way to review previous interactions and insights gained during the analytical process.

3.2.8. Statistical Analysis Module

Packages Used:

- Pandas: For data manipulation and analysis (CSV file data).
- statsmodels.api: For performing statistical analyses such as linear regression, ANOVA, etc.
- scipy.stats: For performing various statistical tests.
- seaborn: For creating visualizations.
- matplotlib.pyplot: For plotting graphs.

This module performs various statistical analyses on the data uploaded via CSV files. It includes linear regression, multiple regression, t-tests, ANOVA, Z-tests, correlation analysis, chi-square tests, moving average, exponential smoothing, autocorrelation, seasonality analysis, histogram, box plot, and Q-Q plot.

3.2.9. User Interface Module

Packages Used:

- Streamlit: For creating the web interface.

This module provides a user-friendly web interface using Streamlit. Users can input data, select analysis options, ask questions, and view analysis results within the same interface. Additionally, it stores past questions and answers for reference during the session

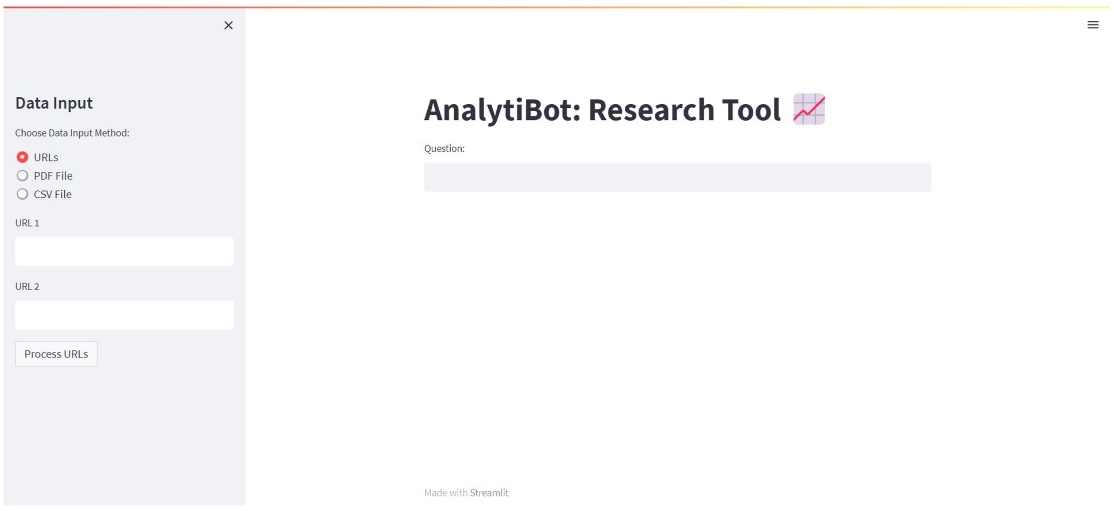


Figure 8: Representation of how streamlit package works through the web interface

Chapter 4

Implementation

4.1. PACKAGES REQUIREMENT

There are multiple packages that are required for implementing all the modules of this project. And we'll be completing the project in Python language version 3.9.0.

S.NO.	**Package Required**	**Uses**	**Package and its version for installing**
1	Streamlit	For building the Web Application.	streamlit==1.22.0
2	LangChain,	It is a framework that acts as a medium between data sources and LLM, Main Package which has all the other packages as sub packages inside it.	langchain==0.0.284
3	Dotenv	Used for loading the .env file, the file that contains the Open API key.	Python-dotenv==1.0.0
4	FAISS	Useful for Similarity search.	faiss-cpu==1.7.4
5	PyPDFLoader	Useful form loading the PDF	PyPDFLoader

		data file.	
6	Unstructured	Useful for loading the HTML unstructured URL.	unstructured==0.9.2
7	OpenAI	Enables to connect and use language models of OpenAI.	OpenAI == 0.28.0
8	Tiktoken, Pyhton-Magic, Libmagic	Other smaller packages for miscellaneous.	tiktoken==0.4.0, python-magic==0.4.27, python-magic-bin==0.4.14, libmagic==1.0
9.	Statsmodels, Scipy	comprehensive set of tools for statistical modeling, hypothesis testing. And various scientific computing functionalities	statsmodels=0.14.1, Scipy==1.13.0
10.	Seaborn	builds on top of matplotlib to create informative and aesthetically pleasing statistical visualizations	Seaborn==0.13. 2

Table 1: All the required packages for the project

For Installing these pacakges use the command ‘pip install package_name’ command, where package_name should be replaced by the packages required.

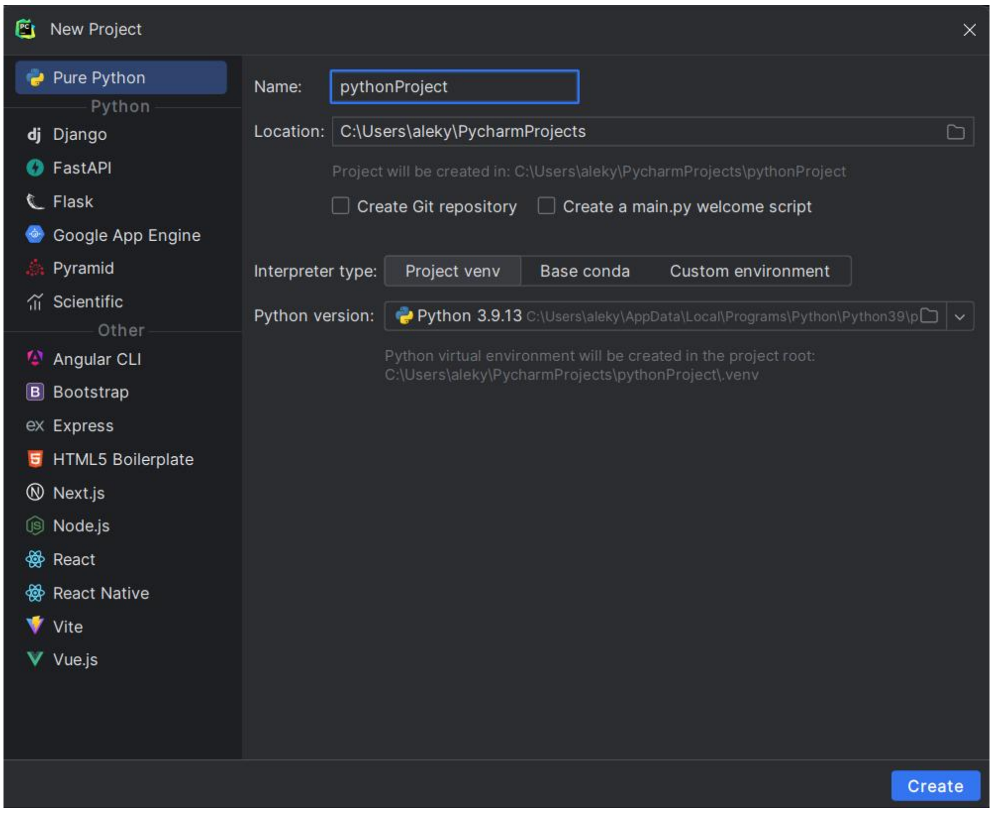
Similar to the packages required we also need library imports, some of the main library imports from the langchain package are:

1. RetrievalQSWithSourcesChain – for retrieving the answer with their sources.
2. OpenAIEmbeddings – for embedding the normal textual information into vectors.
3. RecursiveCharacterTextSplitter – for Splitting the texts recursively.
4. UnstructuredURLLoader – for loading the unstructured URL.
5. PyPDFLoader – for loading the PDF.
6. ChatOpenAI – for getting access and enabling the language models of OpenAI.
7. FAISS – for faster and efficient similarity search.
8. Statsmodels - comprehensive set of tools for statistical modeling, hypothesis testing, and econometrics
9. Scipy - provides various scientific computing functionalities
10. Seaborn - builds on top of matplotlib to create informative and aesthetically pleasing statistical visualizations

4.2. WEB IMPELMENTATION

Step by step procedure of the implementation:

1. Open Pycharm, create a new project with python version 3.9.0 and create a new python file inside the project and put the code in the file, while keeping the API key in another file with .env extension for better security reasons. Install all the required packages from the requirements file.



New Project

Name: pythonProject

Location: C:\Users\aleky\PycharmProjects

Project will be created in: C:\Users\aleky\PycharmProjects\pythonProject

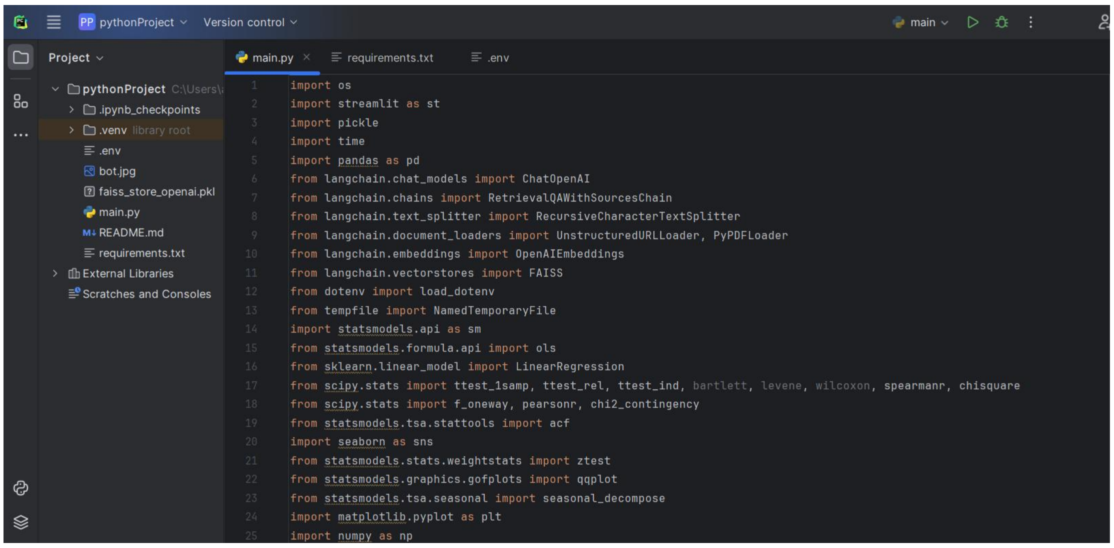
☐ Create Git repository ☐ Create a main.py welcome script

Interpreter type: Project venv | Base conda | Custom environment

Python version: Python 3.9.13 C:\Users\aleky\AppData\Local\Programs\Python\Python39\p

Python virtual environment will be created in the project root: C:\Users\aleky\PycharmProjects\pythonProject\.venv

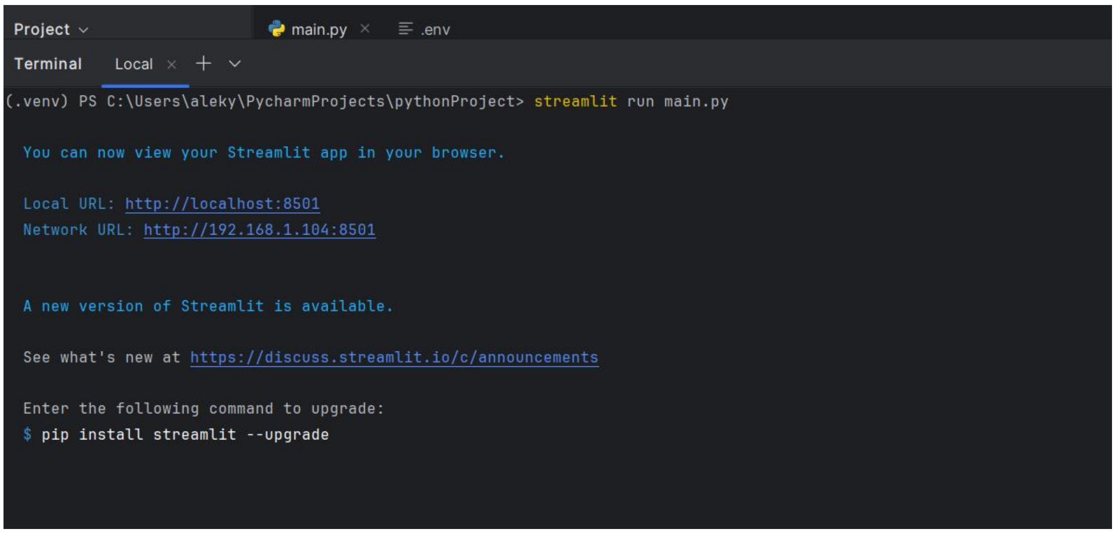
Figure 9: Creating a new project in the Py Charm.



```python
import os
import streamlit as st
import pickle
import time
import pandas as pd
from langchain.chat_models import ChatOpenAI
from langchain.chains import RetrievalQAWithSourcesChain
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.document_loaders import UnstructuredURLLoader, PyPDFLoader
from langchain.embeddings import OpenAIEmbeddings
from langchain.vectorstores import FAISS
from dotenv import load_dotenv
from tempfile import NamedTemporaryFile
import statsmodels.api as sm
from statsmodels.formula.api import ols
from sklearn.linear_model import LinearRegression
from scipy.stats import ttest_1samp, ttest_rel, ttest_ind, bartlett, levene, wilcoxon, spearmanr, chisquare
from scipy.stats import f_oneway, pearsonr, chi2_contingency
from statsmodels.tsa.stattools import acf
import seaborn as sns
from statsmodels.stats.weightstats import ztest
from statsmodels.graphics.gofplots import qqplot
from statsmodels.tsa.seasonal import seasonal_decompose
import matplotlib.pyplot as plt
import numpy as np
```

Figure 10: arranging the order of the files in the project space.

2. Open the terminal and run the code using ‘streamlit run main.py’ command.



```
(.venv) PS C:\Users\aleky\PycharmProjects\pythonProject> streamlit run main.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.1.104:8501


  A new version of Streamlit is available.

  See what's new at https://discuss.streamlit.io/c/announcements

  Enter the following command to upgrade:
  $ pip install streamlit --upgrade
```

Figure 11: python command to execute the code

Once you run the code it directly directs you to the web page where the user can upload the data files or source links.

3. The web page:

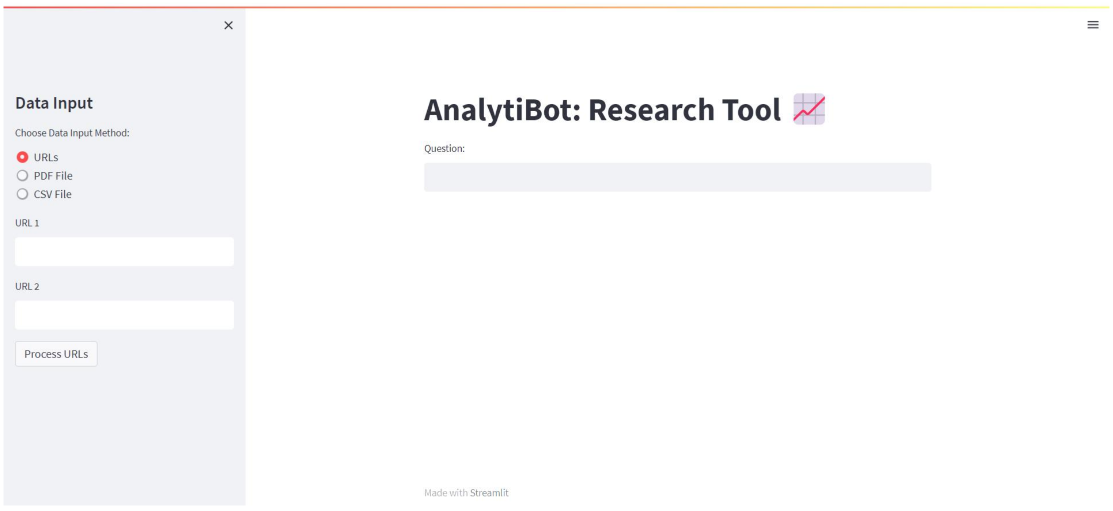
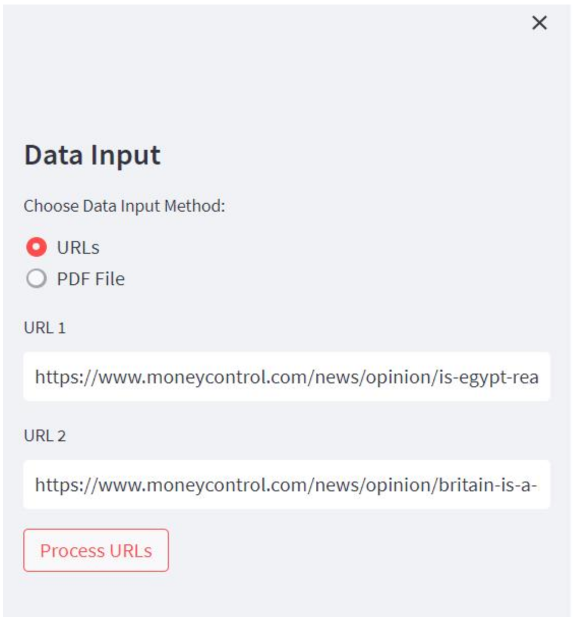


Figure 12: Representation of the User Web Interface

Here the user is provided with 2 choices i.e., to select URL for uploading URLs, or PDF for uploading a PDF.

4. Here firstly I’ll select the URLS option.



URL 1

https://www.moneycontrol.com/news/opinion/is-egypt-rea

URL 2

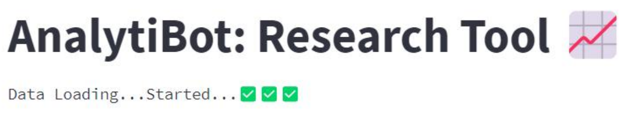
https://www.moneycontrol.com/news/opinion/britain-is-a-

Figure 13: The URL Data loading section, where the user can upload multiple URLs.

After uploading the URLs select the Process URLs button to process and extract the data from the URLs.

Here, there will be multiple stages representing the inside process will be displayed like text splitter done; Vector embeddings done, etc.

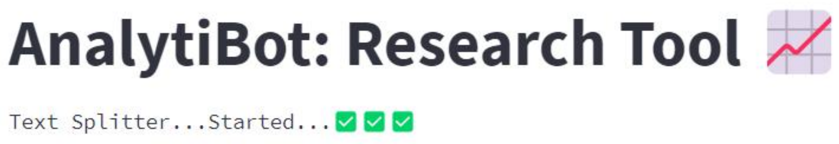
4.1. Data Loading started



Data Loading...Started...✅✅✅

Figure 14: Data Loading Started Internally

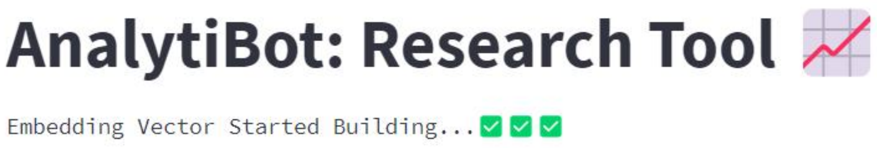
4.2. Text Splitter Started



Text Splitter...Started...✅✅✅

Figure 15: Text splitter started Internally

4.3. Vector Embeddings start building

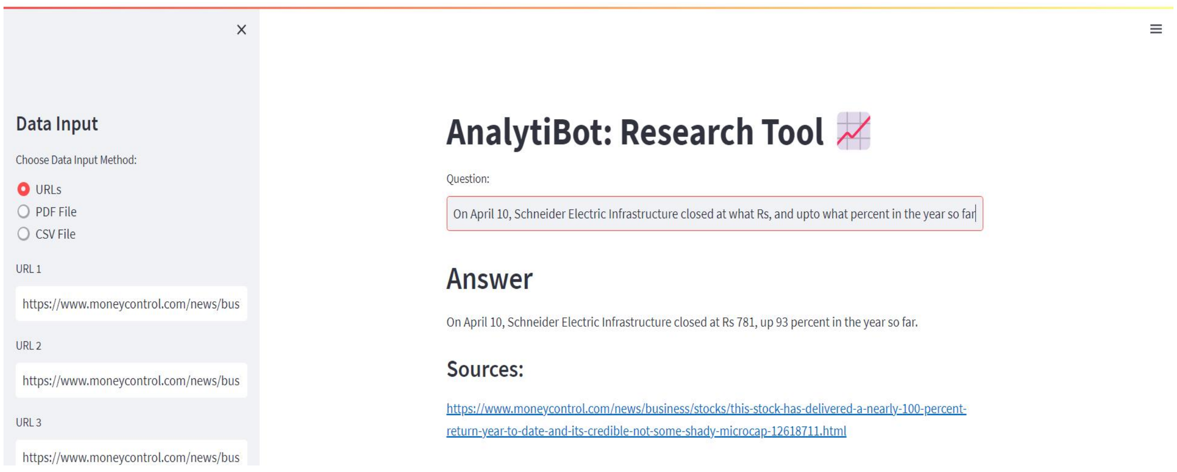

AnalytiBot: Research Tool 📈

Embedding Vector Started Building...✅✅✅

Figure 16: Embedded vectors started Internally

4.4. Now you can prompt multiple question from the URLs and check if we are getting accurate results.

Prompt 1 and its answer with verification:



Data Input

Choose Data Input Method:

[✓ URLs]
PDF File
CSV File

URL 1
https://www.moneycontrol.com/news/bus

URL 2
https://www.moneycontrol.com/news/bus

URL 3
https://www.moneycontrol.com/news/bus

AnalytiBot: Research Tool 📈

Question:
On April 10, Schneider Electric Infrastructure closed at what Rs, and upto what percent in the year so far

Answer

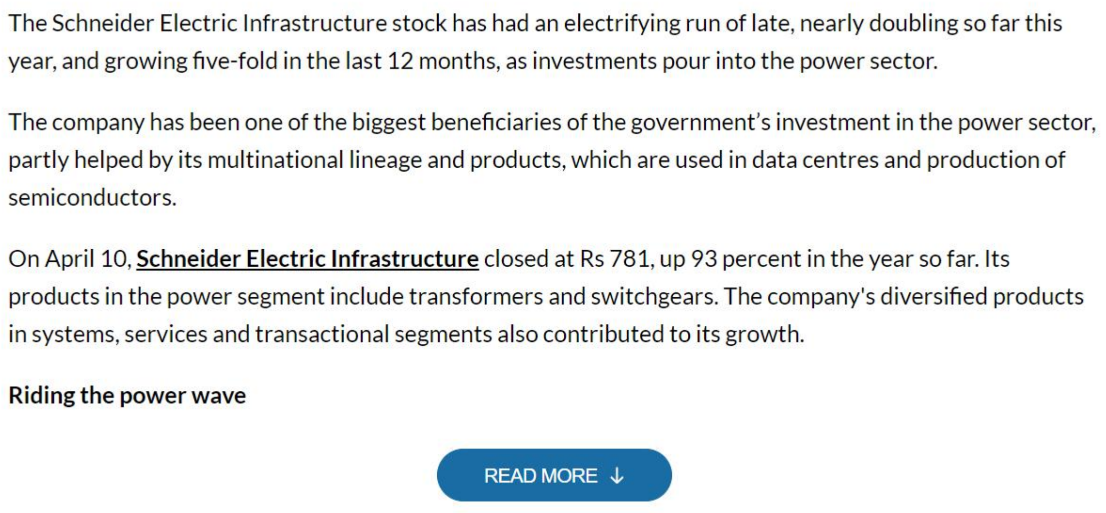
On April 10, Schneider Electric Infrastructure closed at Rs 781, up 93 percent in the year so far.

Sources:

https://www.moneycontrol.com/news/business/stocks/this-stock-has-delivered-a-nearly-100-percent-return-year-to-date-and-its-credible-not-some-shady-microcap-12618711.html

Figure 17: The Prompt 1 given, with its answer produced

Verification from the URL:


The Schneider Electric Infrastructure stock has had an electrifying run of late, nearly doubling so far this year, and growing five-fold in the last 12 months, as investments pour into the power sector.

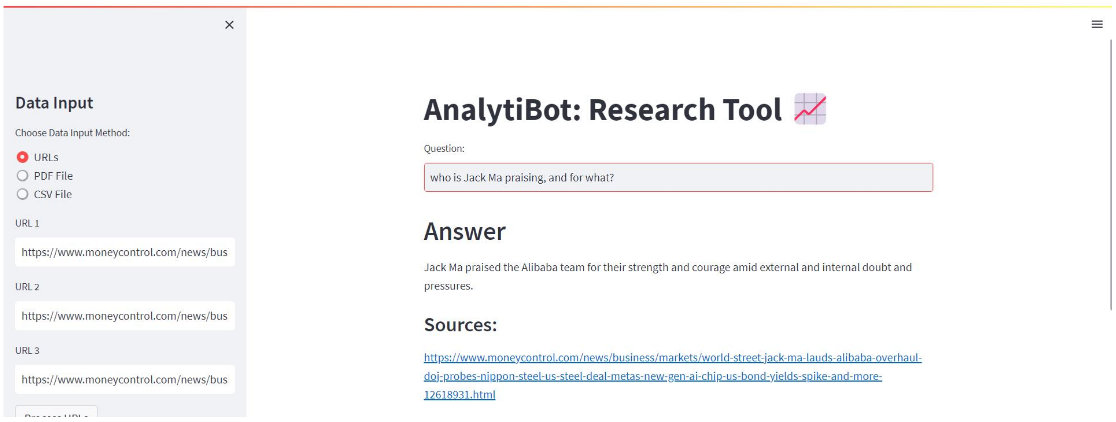
The company has been one of the biggest beneficiaries of the government's investment in the power sector, partly helped by its multinational lineage and products, which are used in data centres and production of semiconductors.

On April 10, **Schneider Electric Infrastructure** closed at Rs 781, up 93 percent in the year so far. Its products in the power segment include transformers and switchgears. The company's diversified products in systems, services and transactional segments also contributed to its growth.

Riding the power wave

Figure 18: Verifying the answer to prompt one with the data of the URL link

Prompt 2 with answer and verification:



URL 1
https://www.moneycontrol.com/news/bus

URL 2
https://www.moneycontrol.com/news/bus

URL 3
https://www.moneycontrol.com/news/bus

AnalytiBot: Research Tool

Question:
who is Jack Ma praising, and for what?

Answer

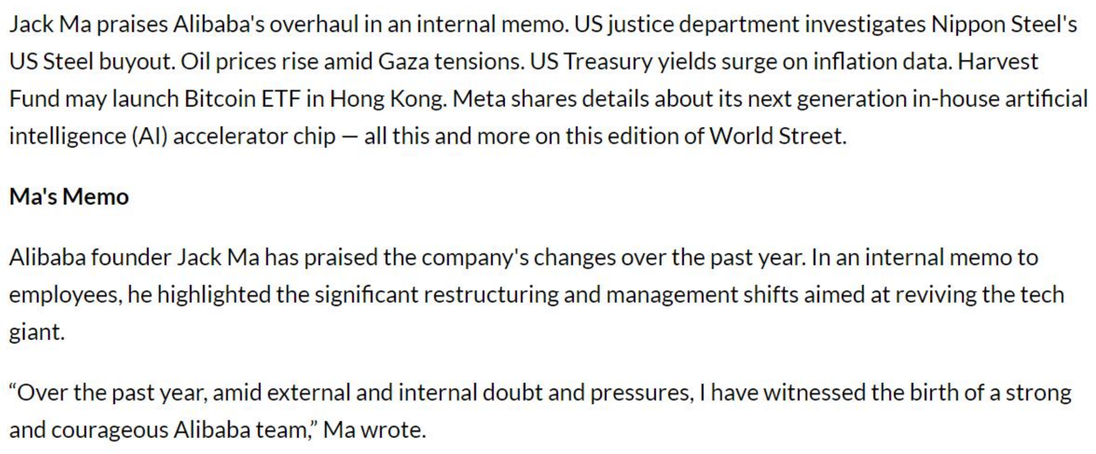
Jack Ma praised the Alibaba team for their strength and courage amid external and internal doubt and pressures.

Sources:

https://www.moneycontrol.com/news/business/markets/world-street-jack-ma-lauds-alibaba-overhaul-doj-probes-nippon-steel-us-steel-deal-metas-new-gen-ai-chip-us-bond-yields-spike-and-more-12618931.html

Figure 19: The Prompt 2 given, with its answer produced

Verification from the URL:


Jack Ma praises Alibaba's overhaul in an internal memo. US justice department investigates Nippon Steel's US Steel buyout. Oil prices rise amid Gaza tensions. US Treasury yields surge on inflation data. Harvest Fund may launch Bitcoin ETF in Hong Kong. Meta shares details about its next generation in-house artificial intelligence (AI) accelerator chip — all this and more on this edition of World Street.

Ma's Memo

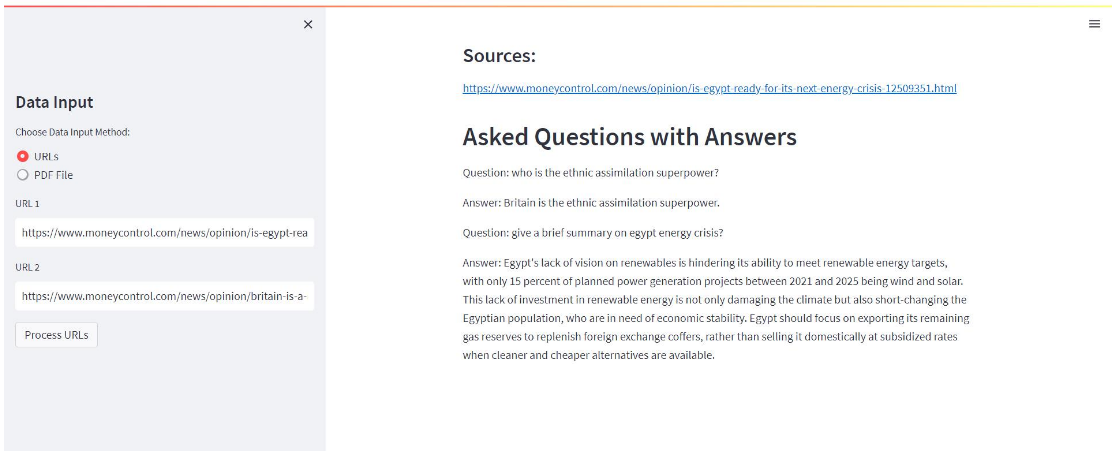
Alibaba founder Jack Ma has praised the company's changes over the past year. In an internal memo to employees, he highlighted the significant restructuring and management shifts aimed at reviving the tech giant.

"Over the past year, amid external and internal doubt and pressures, I have witnessed the birth of a strong and courageous Alibaba team," Ma wrote.

Figure 20: Verifying the answer to prompt two with the data of the URL link

As you can see it not only provides the accurate answer but also provides the source the answer belongs to and also retains the previously asked questions and answers until the session expires.

4.5. Retention of the questions and answers:



URL 1: https://www.moneycontrol.com/news/opinion/is-egypt-rea

URL 2: https://www.moneycontrol.com/news/opinion/britain-is-a-

Sources:

https://www.moneycontrol.com/news/opinion/is-egypt-ready-for-its-next-energy-crisis-12509351.html

Asked Questions with Answers

Question: who is the ethnic assimilation superpower?

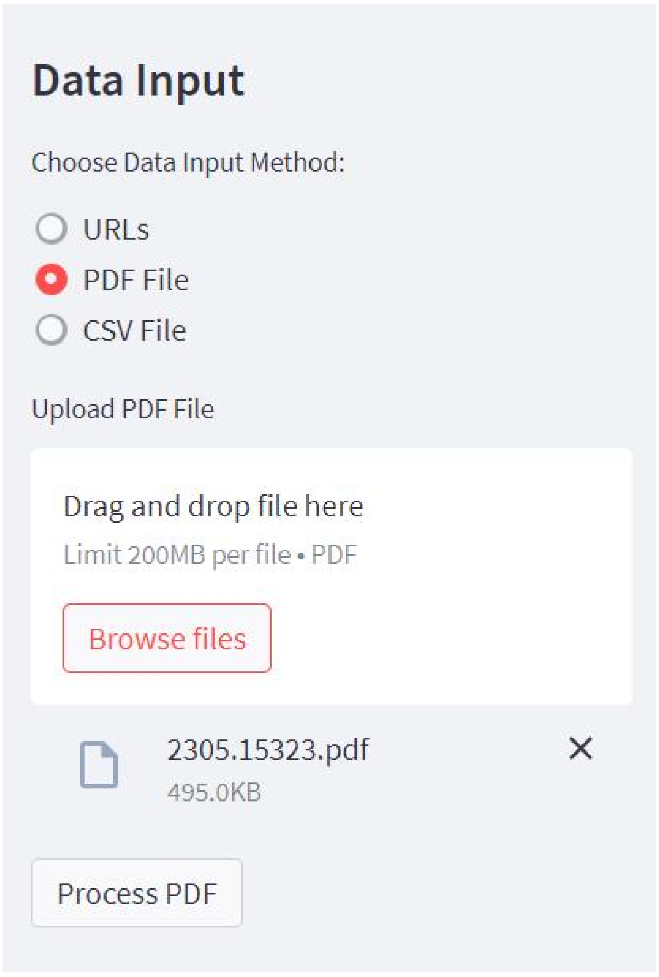
Answer: Britain is the ethnic assimilation superpower.

Question: give a brief summary on egypt energy crisis?

Answer: Egypt's lack of vision on renewables is hindering its ability to meet renewable energy targets, with only 15 percent of planned power generation projects between 2021 and 2025 being wind and solar. This lack of investment in renewable energy is not only damaging the climate but also short-changing the Egyptian population, who are in need of economic stability. Egypt should focus on exporting its remaining gas reserves to replenish foreign exchange coffers, rather than selling it domestically at subsidized rates when cleaner and cheaper alternatives are available.

Figure 21: Representation of the Data retention on the UI

5. Now I’ll try the same with the PDF file:


Data Input

Choose Data Input Method:

( ) URLs
(● PDF File)
( ) CSV File

Upload PDF File

Drag and drop file here
Limit 200MB per file • PDF

Browse files

2305.15323.pdf
495.0KB

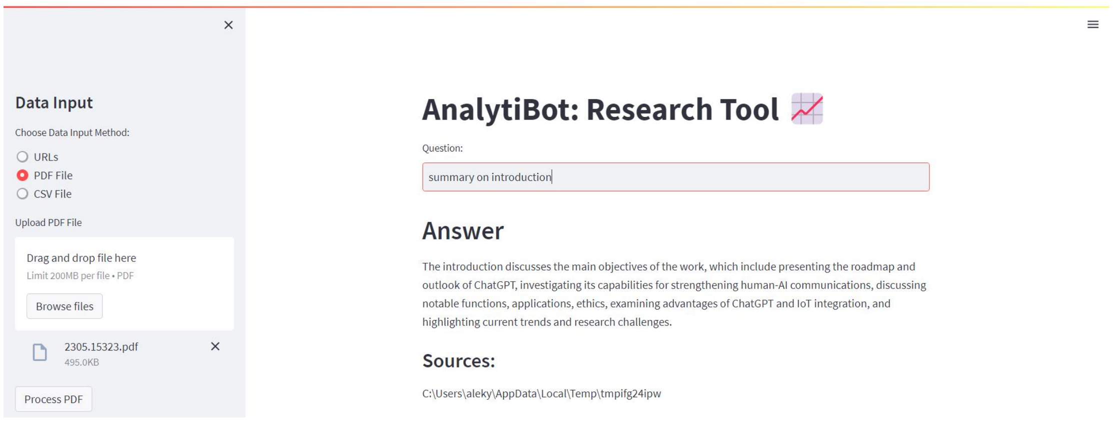
Process PDF

Figure 22: PDF data loading section.

Upload the PDF and process the PDF.

5.1. Similar to the URLs the PDF will also show the internal process and then prompt the question.

Prompt 1 of PDF:



AnalytiBot: Research Tool

Question:

summary on introduction

Answer

The introduction discusses the main objectives of the work, which include presenting the roadmap and outlook of ChatGPT, investigating its capabilities for strengthening human-AI communications, discussing notable functions, applications, ethics, examining advantages of ChatGPT and IoT integration, and highlighting current trends and research challenges.

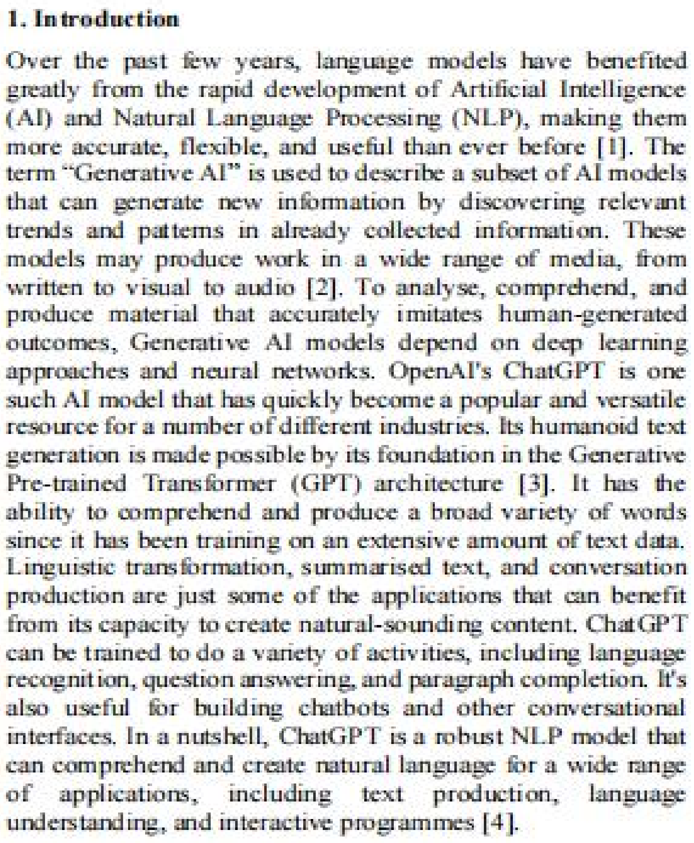
Sources:

C:\Users\aleky\AppData\Local\Temp\tmpifg24ipw

Figure 23: Giving prompt 1 for the uploaded PDF file, with the answer produced

Here it shows the source as a path of the file location as I am uploading the PDF file from my System.

Verifying the answer produced:

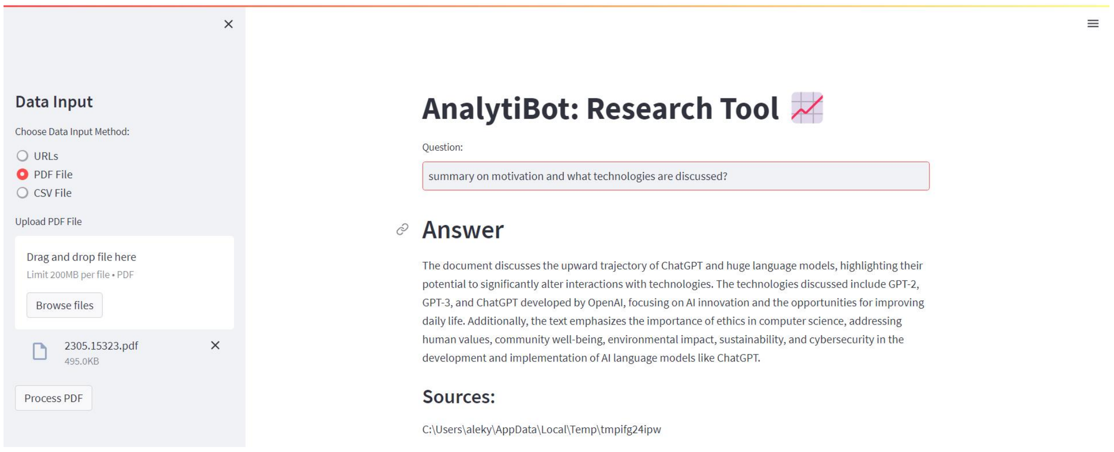


1. Introduction

Over the past few years, language models have benefited greatly from the rapid development of Artificial Intelligence (AI) and Natural Language Processing (NLP), making them more accurate, flexible, and useful than ever before [1]. The term "Generative AI" is used to describe a subset of AI models that can generate new information by discovering relevant trends and patterns in already collected information. These models may produce work in a wide range of media, from written to visual to audio [2]. To analyse, comprehend, and produce material that accurately imitates human-generated outcomes, Generative AI models depend on deep learning approaches and neural networks. OpenAI's ChatGPT is one such AI model that has quickly become a popular and versatile resource for a number of different industries. Its humanoid text generation is made possible by its foundation in the Generative Pre-trained Transformer (GPT) architecture [3]. It has the ability to comprehend and produce a broad variety of words since it has been training on an extensive amount of text data. Linguistic transformation, summarised text, and conversation production are just some of the applications that can benefit from its capacity to create natural-sounding content. ChatGPT can be trained to do a variety of activities, including language recognition, question answering, and paragraph completion. It's also useful for building chatbots and other conversational interfaces. In a nutshell, ChatGPT is a robust NLP model that can comprehend and create natural language for a wide range of applications, including text production, language understanding, and interactive programmes [4].

Figure 24: Verifying with the PDF file for prompt 1

Prompt 2 of PDF:



Data Input

Choose Data Input Method:

- URLs
- [✓ PDF File]
- CSV File

Upload PDF File

Drag and drop file here
Limit 200MB per file • PDF

Browse files

2305.15323.pdf 495.0KB

Process PDF

AnalytiBot: Research Tool

Question:

summary on motivation and what technologies are discussed?

Answer

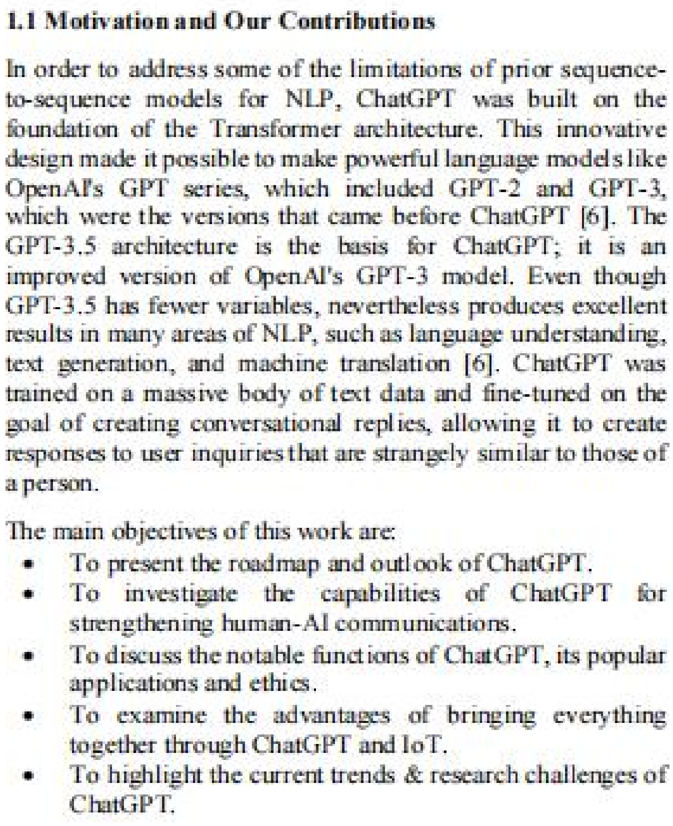
The document discusses the upward trajectory of ChatGPT and huge language models, highlighting their potential to significantly alter interactions with technologies. The technologies discussed include GPT-2, GPT-3, and ChatGPT developed by OpenAI, focusing on AI innovation and the opportunities for improving daily life. Additionally, the text emphasizes the importance of ethics in computer science, addressing human values, community well-being, environmental impact, sustainability, and cybersecurity in the development and implementation of AI language models like ChatGPT.

Sources:

C:\Users\aleky\AppData\Local\Temp\tmpifg24ipw

Figure 25: Giving prompt 2 for the uploaded PDF file, with the answer produced

Verifying it with the PDF content:



1.1 Motivation and Our Contributions

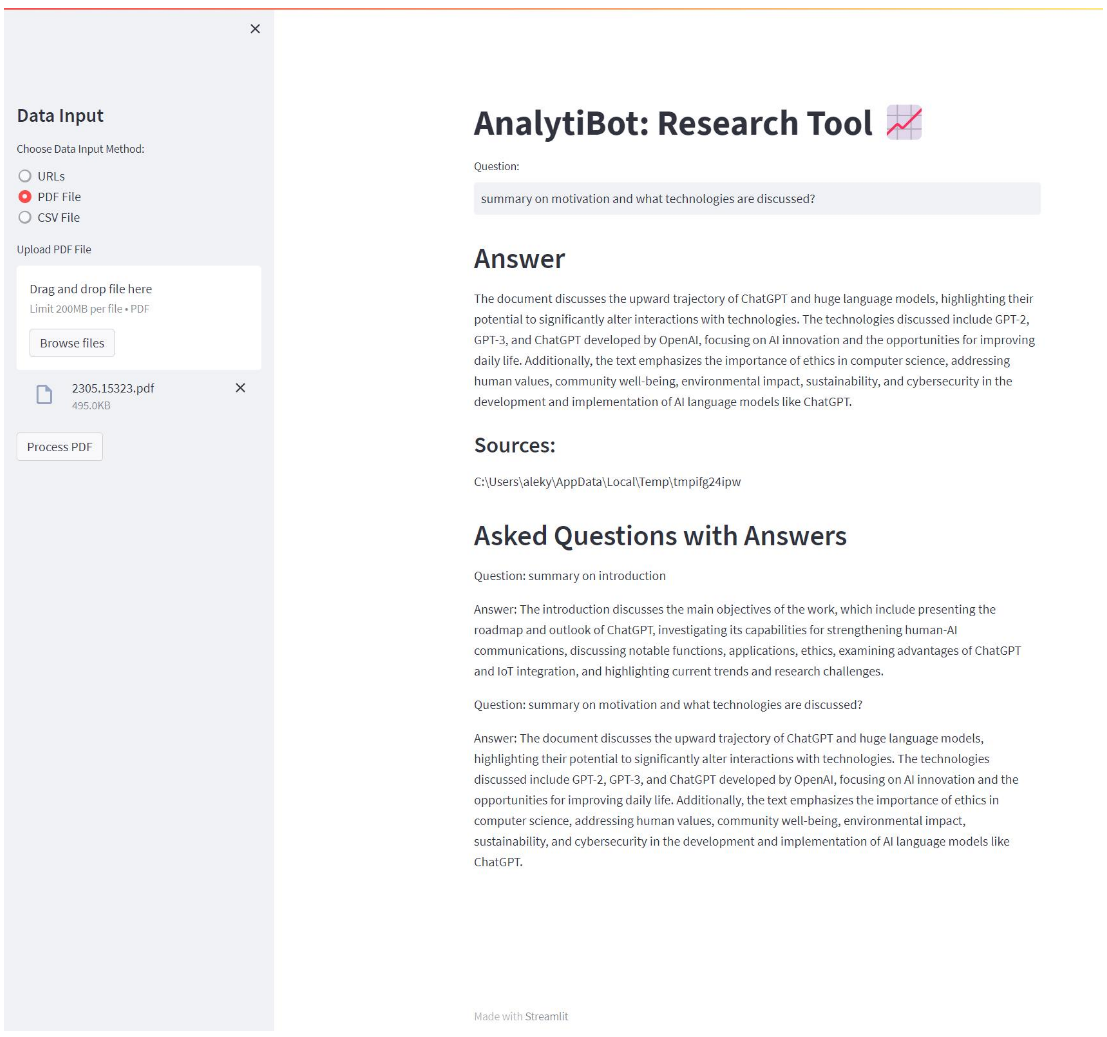
In order to address some of the limitations of prior sequence-to-sequence models for NLP, ChatGPT was built on the foundation of the Transformer architecture. This innovative design made it possible to make powerful language models like OpenAI's GPT series, which included GPT-2 and GPT-3, which were the versions that came before ChatGPT [6]. The GPT-3.5 architecture is the basis for ChatGPT; it is an improved version of OpenAI's GPT-3 model. Even though GPT-3.5 has fewer variables, nevertheless produces excellent results in many areas of NLP, such as language understanding, text generation, and machine translation [6]. ChatGPT was trained on a massive body of text data and fine-tuned on the goal of creating conversational replies, allowing it to create responses to user inquiries that are strangely similar to those of a person.

The main objectives of this work are:

- To present the roadmap and outlook of ChatGPT.
- To investigate the capabilities of ChatGPT for strengthening human-AI communications.
- To discuss the notable functions of ChatGPT, its popular applications and ethics.
- To examine the advantages of bringing everything together through ChatGPT and IoT.
- To highlight the current trends & research challenges of ChatGPT.

Figure 26: Verifying with the PDF file for prompt 2

6. All Prompts and their answers retention shown:



AnalytiBot: Research Tool

Question:

summary on motivation and what technologies are discussed?

Answer

The document discusses the upward trajectory of ChatGPT and huge language models, highlighting their potential to significantly alter interactions with technologies. The technologies discussed include GPT-2, GPT-3, and ChatGPT developed by OpenAI, focusing on AI innovation and the opportunities for improving daily life. Additionally, the text emphasizes the importance of ethics in computer science, addressing human values, community well-being, environmental impact, sustainability, and cybersecurity in the development and implementation of AI language models like ChatGPT.

Sources:

C:\Users\aleky\AppData\Local\Temp\tmpifg24ipw

Asked Questions with Answers

Question: summary on introduction

Answer: The introduction discusses the main objectives of the work, which include presenting the roadmap and outlook of ChatGPT, investigating its capabilities for strengthening human-AI communications, discussing notable functions, applications, ethics, examining advantages of ChatGPT and IoT integration, and highlighting current trends and research challenges.

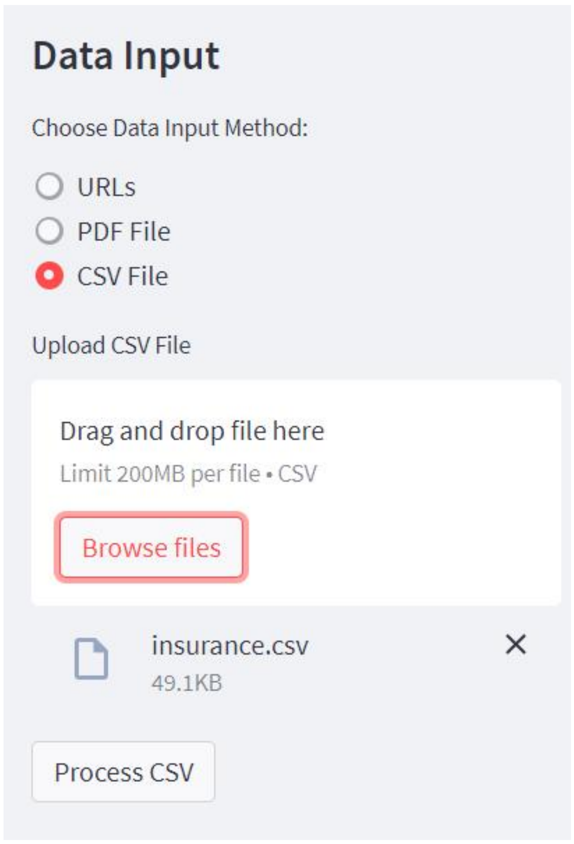
Question: summary on motivation and what technologies are discussed?

Answer: The document discusses the upward trajectory of ChatGPT and huge language models, highlighting their potential to significantly alter interactions with technologies. The technologies discussed include GPT-2, GPT-3, and ChatGPT developed by OpenAI, focusing on AI innovation and the opportunities for improving daily life. Additionally, the text emphasizes the importance of ethics in computer science, addressing human values, community well-being, environmental impact, sustainability, and cybersecurity in the development and implementation of AI language models like ChatGPT.

Figure 27: Complete view of the User web interface

And once the required analysis is done close it to close the session.

7. CSV File Input:


Data Input

Choose Data Input Method:

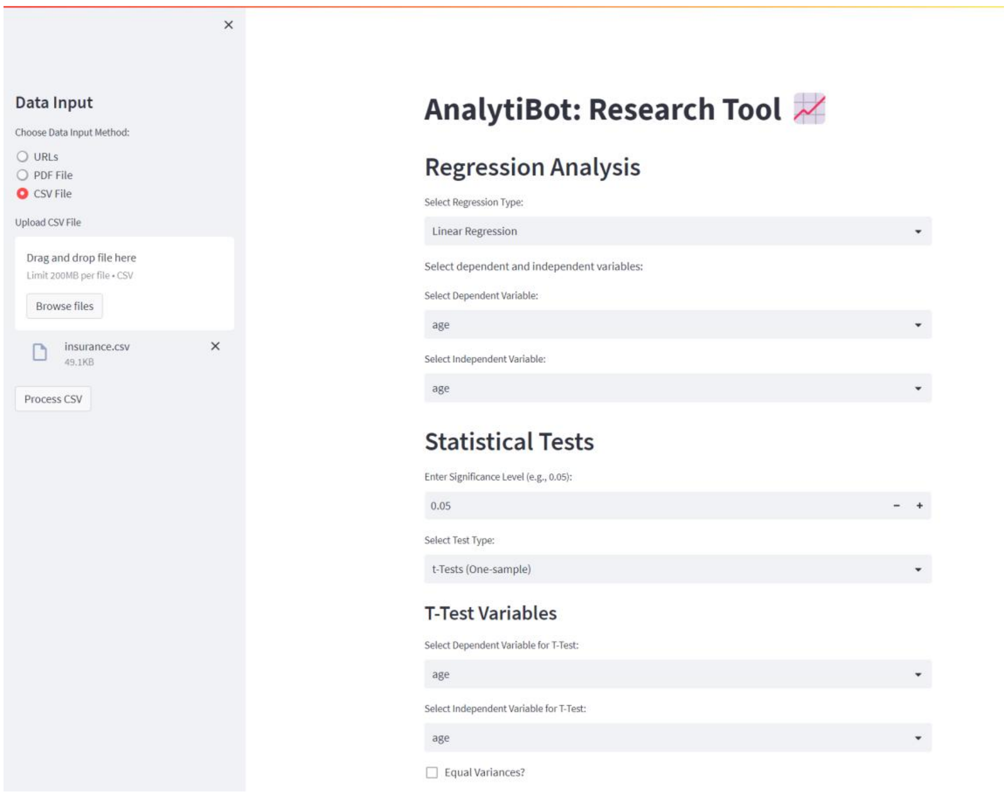
URLs
PDF File
[✓ CSV File]

Upload CSV File

Drag and drop file here
Limit 200MB per file • CSV

Figure 28: CSV data file loading section.

7.1. You will be provided with regression models and Statistical tests that you can perform on your data inside the CSV file:


Data Input

Choose Data Input Method:

- URLs
- PDF File
- [✓ CSV File]

Upload CSV File

Drag and drop file here

Limit 200MB per file • CSV

Browse files

insurance.csv 49.1KB

Process CSV

AnalytiBot: Research Tool

Regression Analysis

Select Regression Type: Linear Regression

Select dependent and independent variables:

Select Dependent Variable: age

Select Independent Variable: age

Statistical Tests

Enter Significance Level (e.g., 0.05): 0.05

Select Test Type: t-Tests (One-sample)

T-Test Variables

Select Dependent Variable for T-Test: age

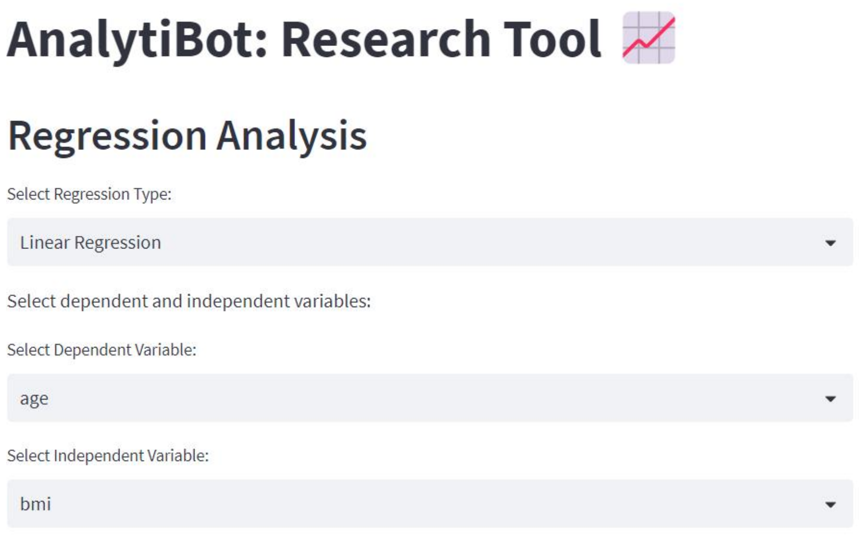
Select Independent Variable for T-Test: age

Equal Variances? [ ]

Figure 29: Regression and Statistical tests interface.

7.2. Performing Linear and multiple regression analysis on the variables.

7.2.1. Linear Regression


AnalytiBot: Research Tool

Regression Analysis

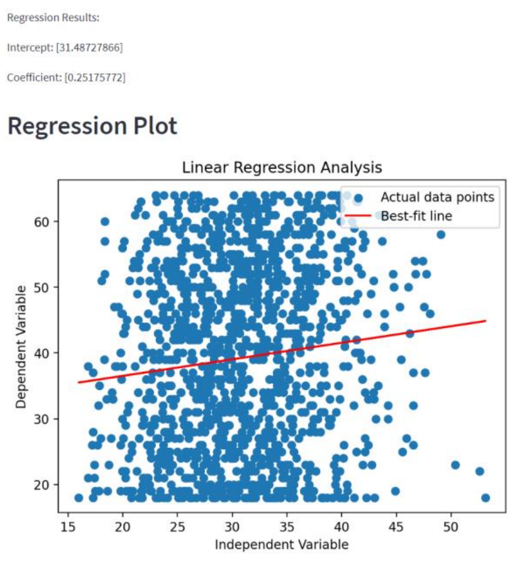
Select Regression Type: Linear Regression

Select dependent and independent variables:

Select Dependent Variable: age

Select Independent Variable: bmi

Figure 30: Linear Regression Input.


Regression Results:

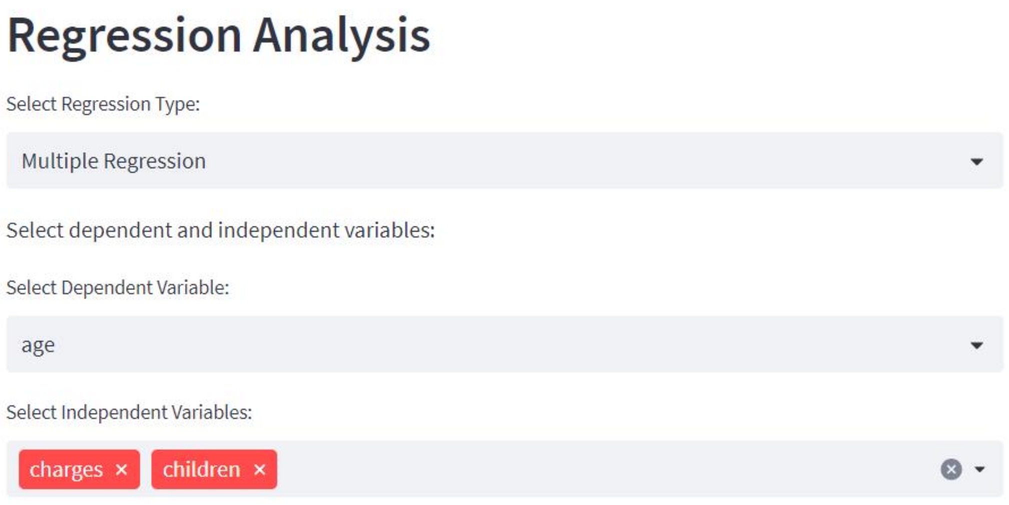
Intercept: [31.48727866]

Coefficient: [0.25175772]

Regression Plot

Figure 31: Linear Regression Output.

7.2.2 Input and Output for Multiple Regression:


Regression Analysis

Select Regression Type:

Multiple Regression

Select dependent and independent variables:

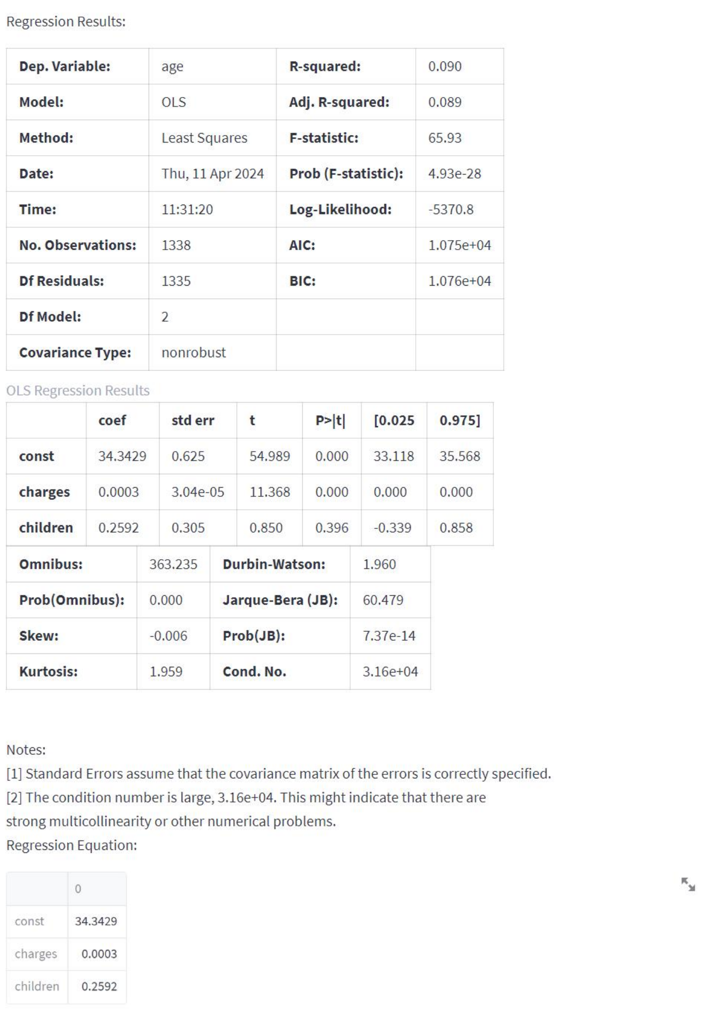
Select Dependent Variable:

age

Select Independent Variables:

charges × children ×

Figure 32: Multiple Regression Input.



Regression Results:

Dep. Variable:	age	**R-squared:**	0.090
Model:	OLS	**Adj. R-squared:**	0.089
Method:	Least Squares	**F-statistic:**	65.93
Date:	Thu, 11 Apr 2024	**Prob (F-statistic):**	4.93e-28
Time:	11:31:20	**Log-Likelihood:**	-5370.8
No. Observations:	1338	**AIC:**	1.075e+04
Df Residuals:	1335	**BIC:**	1.076e+04
Df Model:	2		
Covariance Type:	nonrobust		

OLS Regression Results

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	34.3429	0.625	54.989	0.000	33.118	35.568
charges	0.0003	3.04e-05	11.368	0.000	0.000	0.000
children	0.2592	0.305	0.850	0.396	-0.339	0.858

Omnibus:	363.235	**Durbin-Watson:**	1.960
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	60.479
Skew:	-0.006	**Prob(JB):**	7.37e-14
Kurtosis:	1.959	**Cond. No.**	3.16e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.16e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
Regression Equation:

	0
const	34.3429
charges	0.0003
children	0.2592

Regression Plot

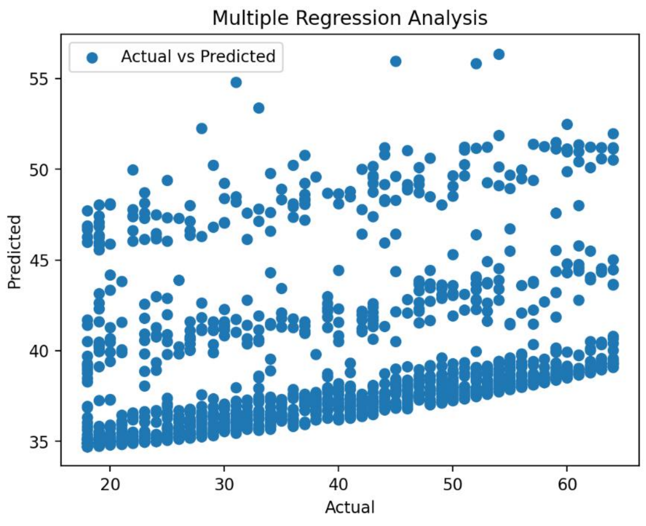
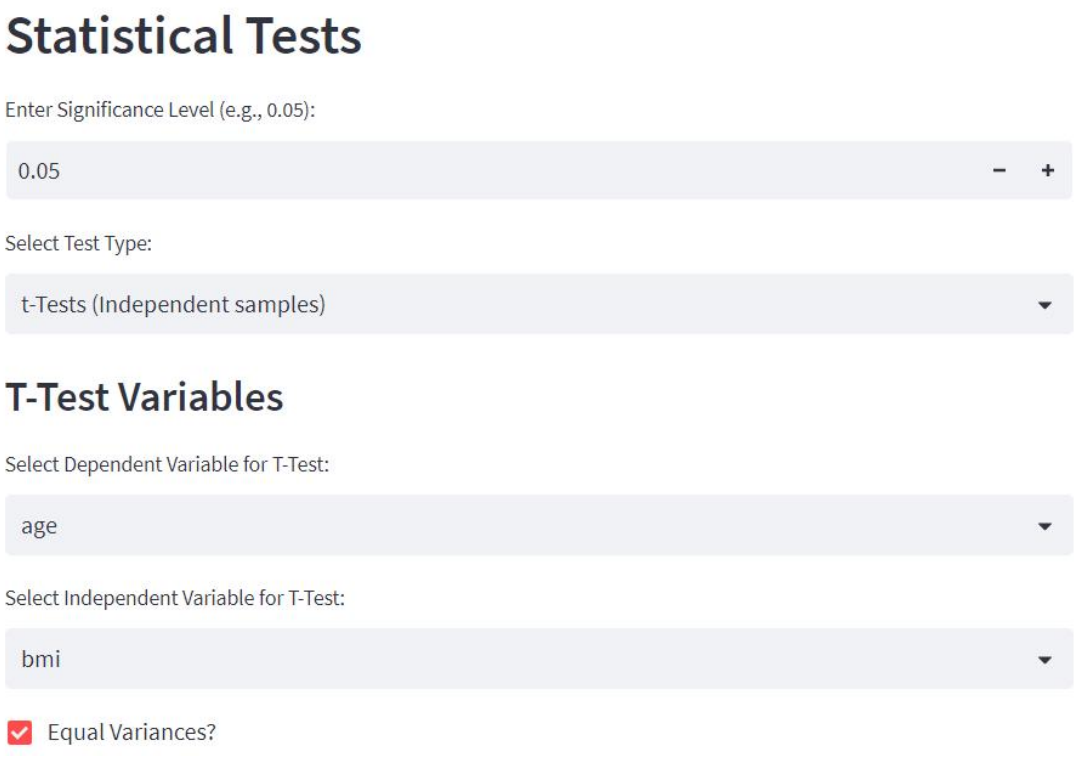


Figure 33: Multiple Regression Output.

7.3. Performing and giving one example for each statistical model using the variables from the uploaded CSV file.

7.3.1. Input section



Statistical Tests

Enter Significance Level (e.g., 0.05):

0.05

Select Test Type:

t-Tests (Independent samples)

T-Test Variables

Select Dependent Variable for T-Test:

age

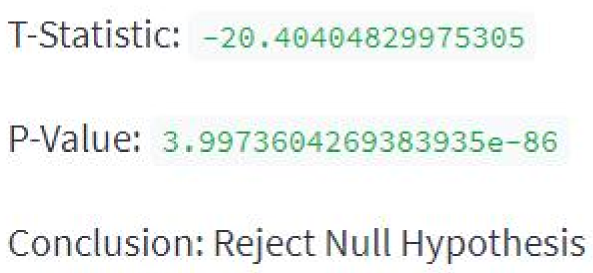
Select Independent Variable for T-Test:

bmi

☑ Equal Variances?

Figure 34: Input section for statistical tests section to choose the kind of test you want perform and the variables that the test is to be done on along with significance level.

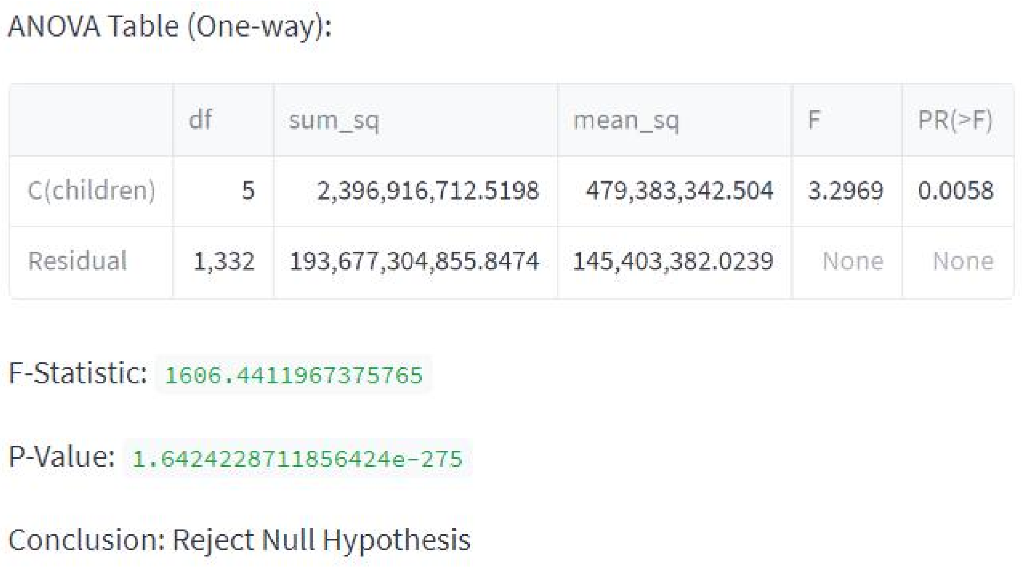
7.3.2. T-test Output:



```
T-Statistic: -20.40404829975305

P-Value: 3.9973604269383935e-86

Conclusion: Reject Null Hypothesis
```

Figure 35: T-test Output.

7.3.3. One way ANOVA Output:



ANOVA Table (One-way):

	df	sum_sq	mean_sq	F	PR(>F)
C(children)	5	2,396,916,712.5198	479,383,342.504	3.2969	0.0058
Residual	1,332	193,677,304,855.8474	145,403,382.0239	None	None

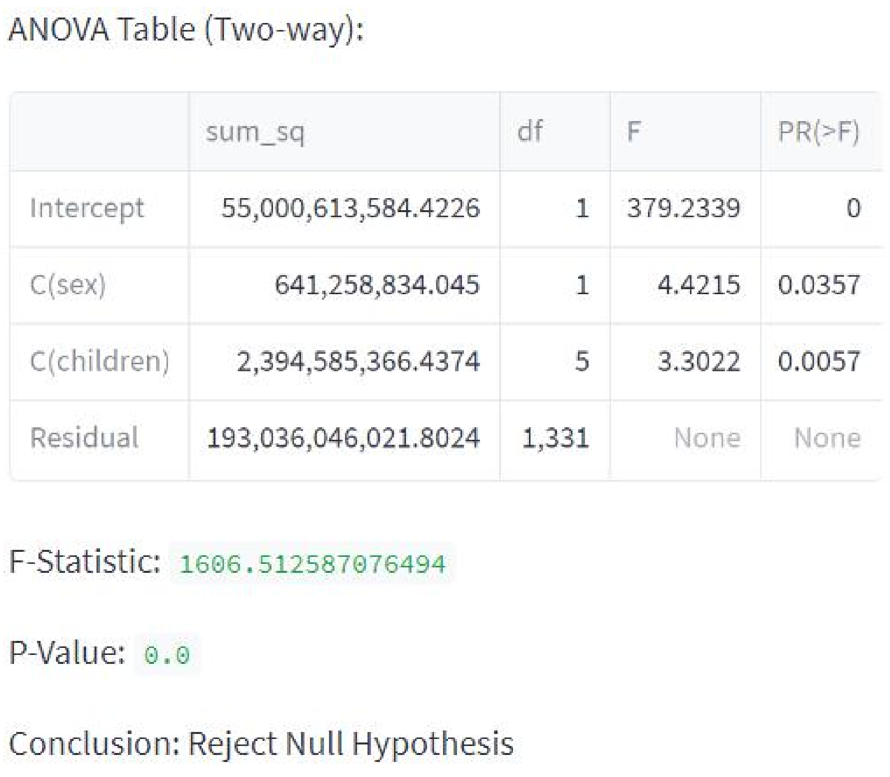
F-Statistic: `1606.4411967375765`

P-Value: `1.6424228711856424e-275`

Conclusion: Reject Null Hypothesis

Figure 36: One way ANOVA Output.

7.3.4. Two way ANOVA Output:



ANOVA Table (Two-way):

	sum_sq	df	F	PR(>F)
Intercept	55,000,613,584.4226	1	379.2339	0
C(sex)	641,258,834.045	1	4.4215	0.0357
C(children)	2,394,585,366.4374	5	3.3022	0.0057
Residual	193,036,046,021.8024	1,331	None	None

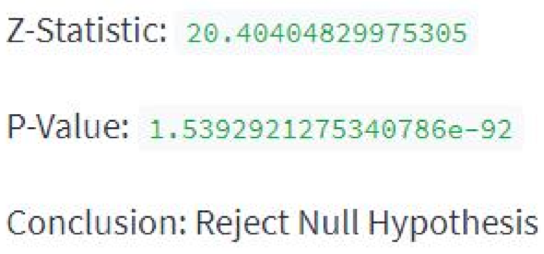
F-Statistic: `1606.512587076494`

P-Value: `0.0`

Conclusion: Reject Null Hypothesis

Figure 37: Two way ANOVA Output.

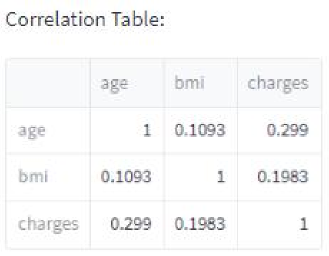
7.3.5. Z test Output:



```
Z-Statistic: 20.40404829975305

P-Value: 1.5392921275340786e-92

Conclusion: Reject Null Hypothesis
```

Figure 38: Z-test Output.

7.3.6. Correlation Analysis Output:



Correlation Table:

	age	bmi	charges
age	1	0.1093	0.299
bmi	0.1093	1	0.1983
charges	0.299	0.1983	1

Correlation Matrix Plot:

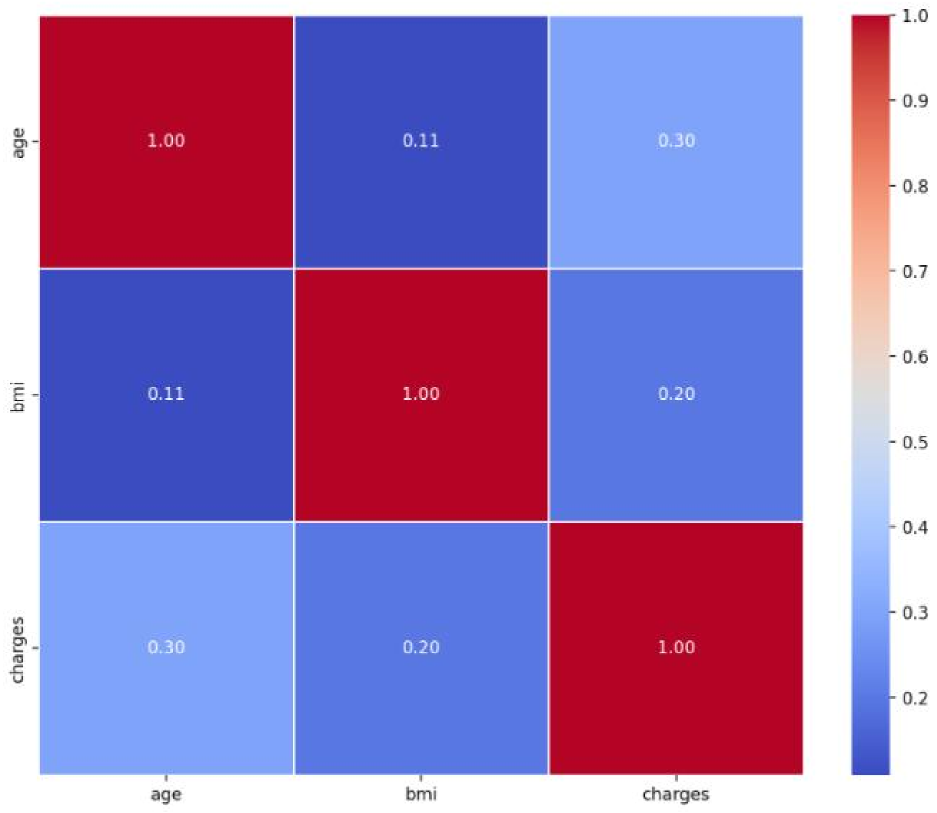
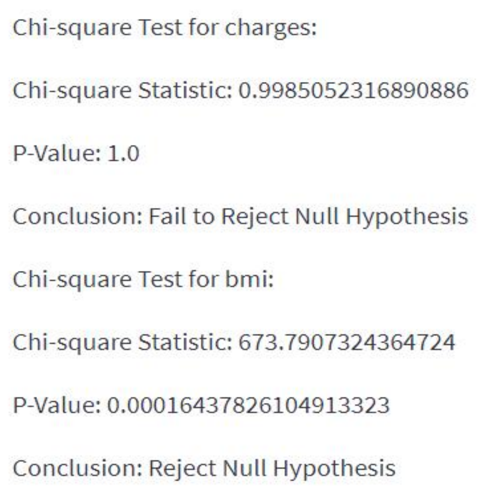


Figure 39: Correlation Analysis Output.

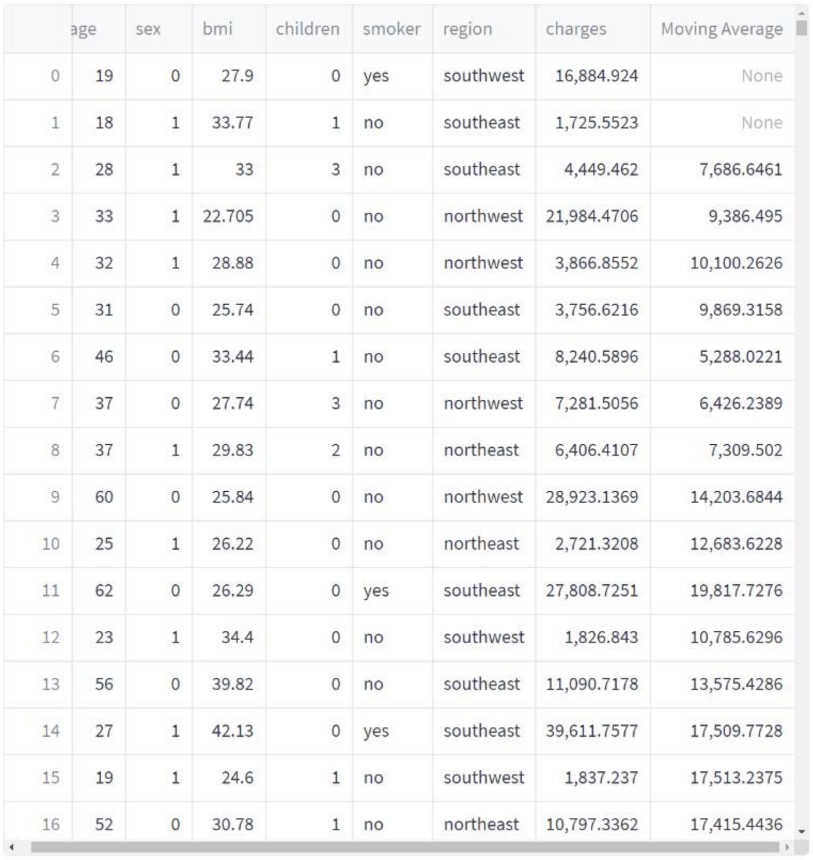
7.3.7. Chi - Square test Output:



```
Chi-square Test for charges:

Chi-square Statistic: 0.9985052316890886

P-Value: 1.0

Conclusion: Fail to Reject Null Hypothesis

Chi-square Test for bmi:

Chi-square Statistic: 673.7907324364724

P-Value: 0.00016437826104913323

Conclusion: Reject Null Hypothesis
```

Figure 40: Chi-square test Output.

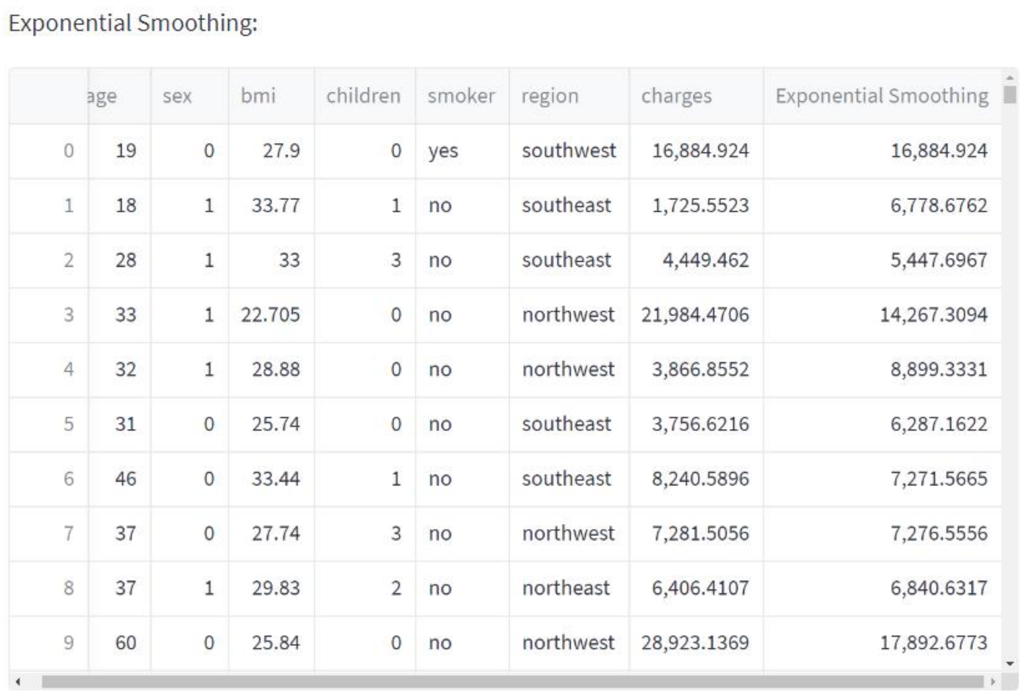
7.3.8. Moving Average Output:



	age	sex	bmi	children	smoker	region	charges	Moving Average
0	19	0	27.9	0	yes	southwest	16,884.924	None
1	18	1	33.77	1	no	southeast	1,725.5523	None
2	28	1	33	3	no	southeast	4,449.462	7,686.6461
3	33	1	22.705	0	no	northwest	21,984.4706	9,386.495
4	32	1	28.88	0	no	northwest	3,866.8552	10,100.2626
5	31	0	25.74	0	no	southeast	3,756.6216	9,869.3158
6	46	0	33.44	1	no	southeast	8,240.5896	5,288.0221
7	37	0	27.74	3	no	northwest	7,281.5056	6,426.2389
8	37	1	29.83	2	no	northeast	6,406.4107	7,309.502
9	60	0	25.84	0	no	northwest	28,923.1369	14,203.6844
10	25	1	26.22	0	no	northeast	2,721.3208	12,683.6228
11	62	0	26.29	0	yes	southeast	27,808.7251	19,817.7276
12	23	1	34.4	0	no	southwest	1,826.843	10,785.6296
13	56	0	39.82	0	no	southeast	11,090.7178	13,575.4286
14	27	1	42.13	0	yes	southeast	39,611.7577	17,509.7728
15	19	1	24.6	1	no	southwest	1,837.237	17,513.2375
16	52	0	30.78	1	no	northeast	10,797.3362	17,415.4436

Figure 41: Moving Average Output.

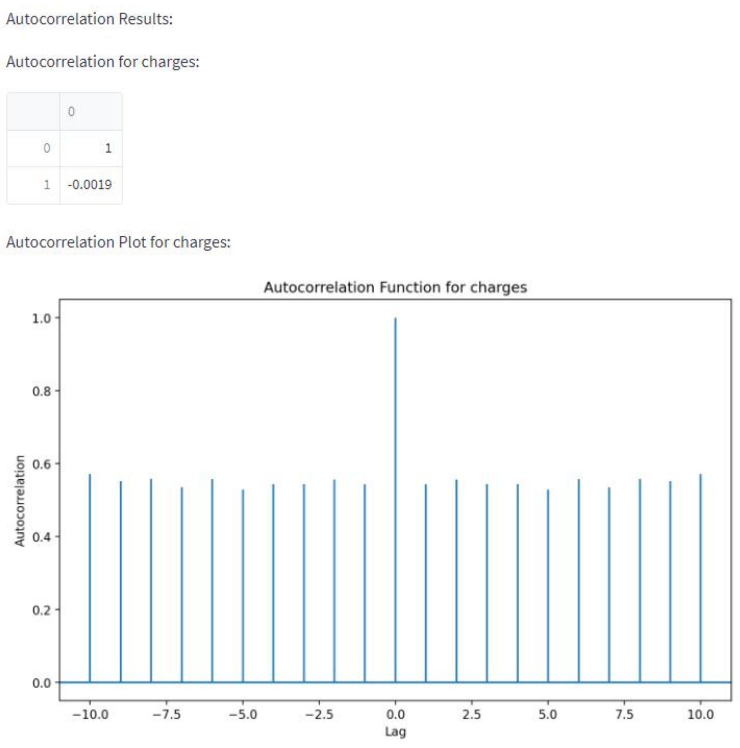
7.3.9. Exponential Smothering Output:


Exponential Smoothing:

	age	sex	bmi	children	smoker	region	charges	Exponential Smoothing
0	19	0	27.9	0	yes	southwest	16,884.924	16,884.924
1	18	1	33.77	1	no	southeast	1,725.5523	6,778.6762
2	28	1	33	3	no	southeast	4,449.462	5,447.6967
3	33	1	22.705	0	no	northwest	21,984.4706	14,267.3094
4	32	1	28.88	0	no	northwest	3,866.8552	8,899.3331
5	31	0	25.74	0	no	southeast	3,756.6216	6,287.1622
6	46	0	33.44	1	no	southeast	8,240.5896	7,271.5665
7	37	0	27.74	3	no	northwest	7,281.5056	7,276.5556
8	37	1	29.83	2	no	northeast	6,406.4107	6,840.6317
9	60	0	25.84	0	no	northwest	28,923.1369	17,892.6773

Figure 42: Exponential Smothering Output.

7.3.10. Auto-correlation Output:


Autocorrelation Results:

Autocorrelation for charges:

	0
0	1
1	-0.0019

Autocorrelation Plot for charges:

Figure 43: Auto Correlation Output.

7.3.11. Seasonality Analysis Output:

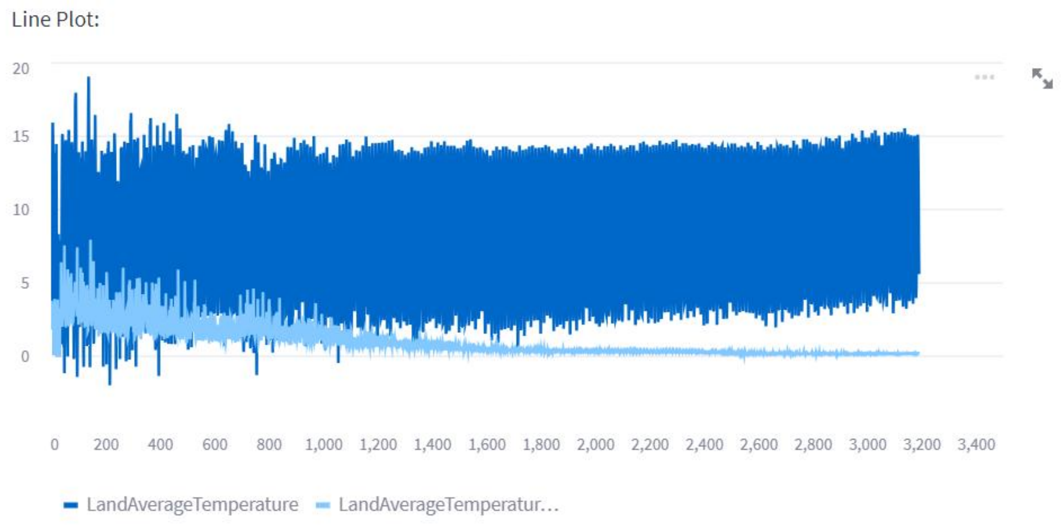


Figure 44: Seasonality Analysis Output.

7.3.12. Histogram Output:

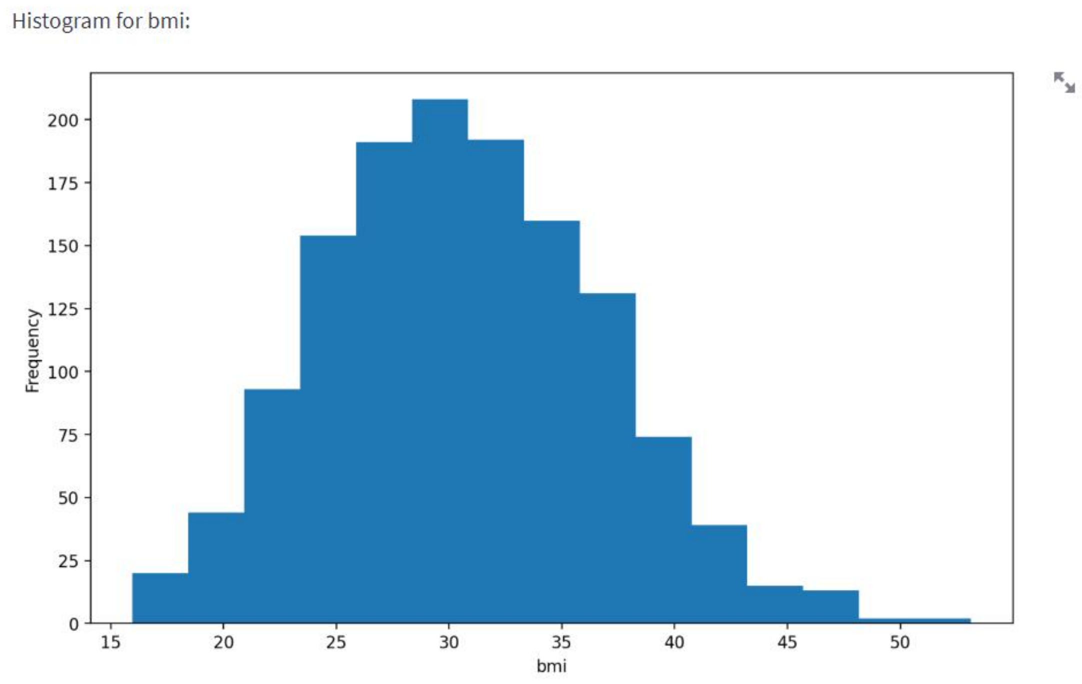


Figure 45: Histogram Output.

7.3.13. Box-plot Output:

Box Plot:

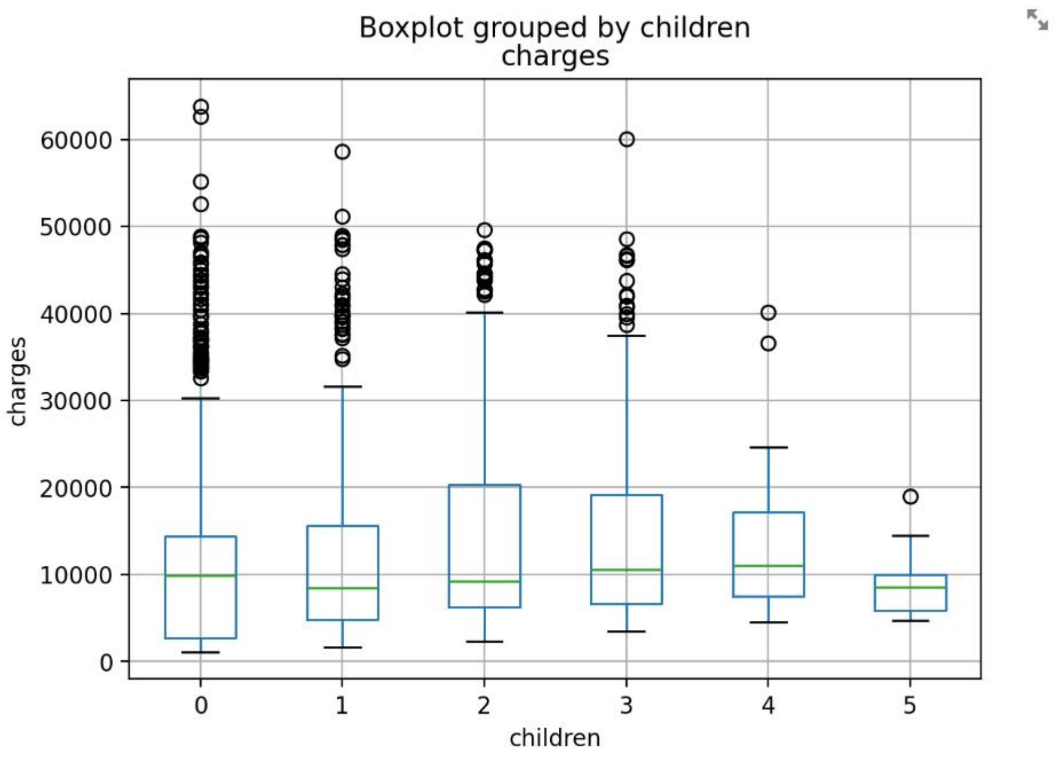


Figure 46: Box-plot by grouping Output.

Box Plot:

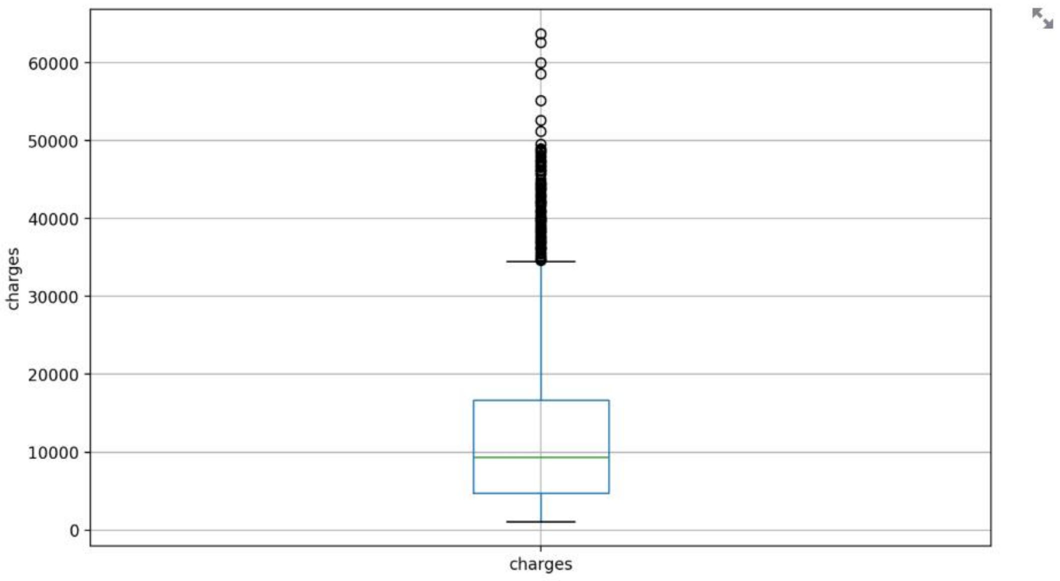


Figure 47: Box-plot Output.

7.3.14. Q-Q plot Output:

Q-Q Plot:

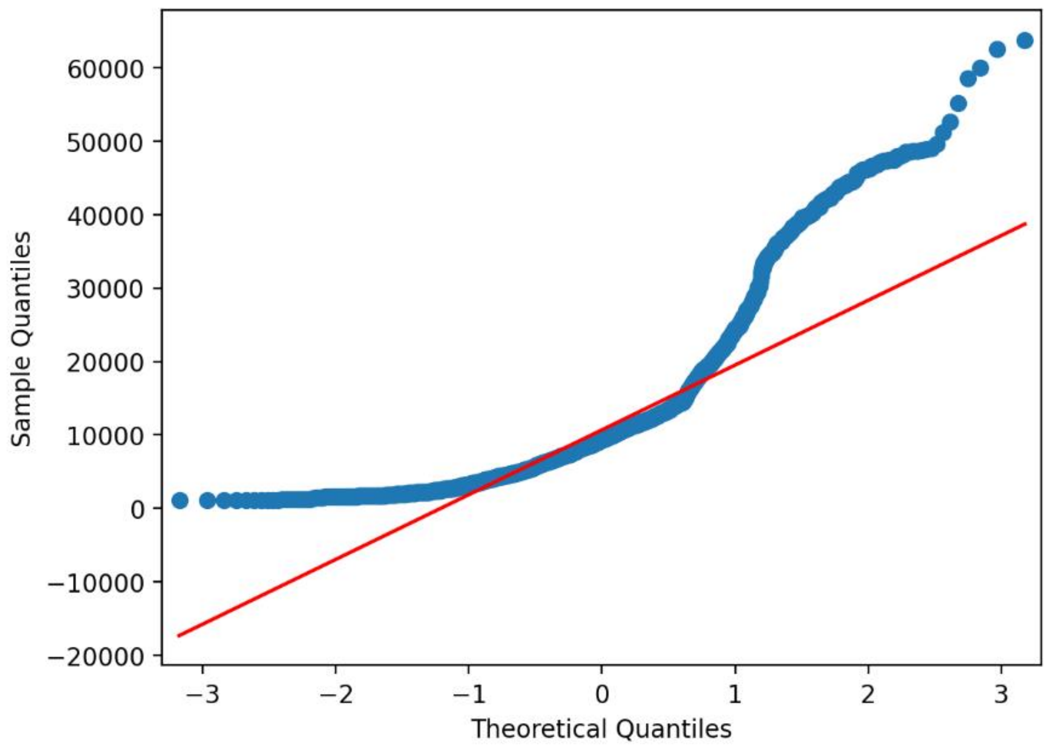


Figure 48: Q-Q plot Output.

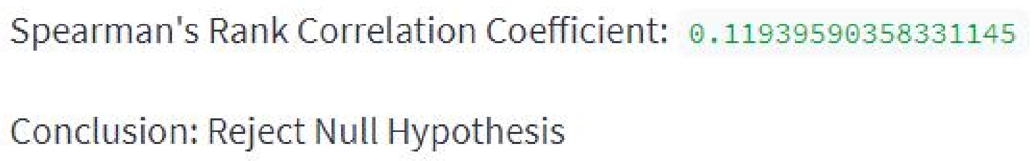
7.3.15. Pearson's Correlation coefficient Output:

Pearson Correlation Coefficient: `0.19834096883362876`

Conclusion: Reject Null Hypothesis

Figure 49: Pearson's Correlation coefficient Output.

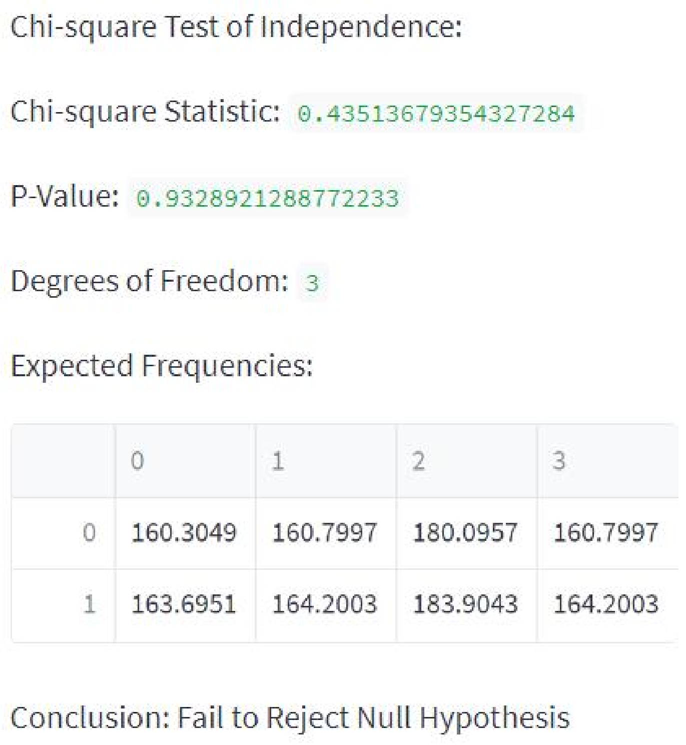
7.3.16. Spear-man Correlation coefficient Output:



```
Spearman's Rank Correlation Coefficient: 0.11939590358331145

Conclusion: Reject Null Hypothesis
```

Figure 50: Spear-man Correlation coefficient Output.

7.3.17. Chi square test of independence Output:



```
Chi-square Test of Independence:

Chi-square Statistic: 0.43513679354327284

P-Value: 0.9328921288772233

Degrees of Freedom: 3

Expected Frequencies:
```

	0	1	2	3
0	160.3049	160.7997	180.0957	160.7997
1	163.6951	164.2003	183.9043	164.2003

```
Conclusion: Fail to Reject Null Hypothesis
```

Figure 51: Chi square test of independence Output.

Chapter 5

Conclusion and Future Work

5.1 CONCLUSION

In conclusion, the proposed model of chatbot is exclusively made for the advancement in the research field, by offering the users a better interface to communicate with the data. The suggested Research Bot offers scholars an advanced and user-friendly platform for going through the vast quantity of research papers and web articles, which is an important advancement in the field of research in academia analysis. AnalytiaBot simplifies the process of extracting data, analyzing it, and providing insights from unstructured data sources by utilizing cutting-edge technology like Large Language Models (LLMs), natural language processing (NLP) techniques, and creative frameworks like LangChain. Due to its intuitive interface and sophisticated capabilities, Research Bot enables users to effectively sift through enormous records of research material, find hidden information, and make sensible decisions in their respective fields of study. Research Bot speeds the rate of discovery in academic research, promotes collaboration, and increases efficiency by automating time-consuming processes along with providing useful insights.

Looking back on Research Bots creation journey, it is clear that there's a lot of room for additional improvements and developments. Research Bot has the ability to significantly transform the field of academic investigation analysis and information generation by always evolving to adapt to constantly evolving technology and customer needs.

5.2. FUTURE SCOPE

The future scope of the technology used for natural language processing (NLP) is highly promising in terms of expanding its range of abilities and applications, particularly in the field of text-to-vector embedding utilizing OpenAI's API. More new versions of the OpenAI API will provide us with more advanced embedding language model approaches in the years to come. These approaches should be able to recognize contextual and finer semantic subtleties in text. In addition, fresh techniques for text embedding that integrate multiple modal input modules—such as pictures and audio—may come along as a result of breakthroughs in artificial intelligence (AI) and machine learning research, greatly enhancing the representation of textual material.

Furthermore, future advancements may enhance the expansion, effectiveness, and availability of text embedding services, allowing effortless integration into an array of networks and applications, as the need for natural language processing (NLP) solutions rises across industry sectors.

Appendices

APPENDIX 1

Large Language Model (LLM):

LLM plays a crucial role in this project. Here is an introduction to it with its internal working.

LLM is a part and a type of DL (Deep Learning), which is designed to extract, process, analyse and understand human language. LLMs are generally fed with a huge and vast amount of data, they are trained on this data, which allows them to learn and analyse the statistical patterns, trends and correlations present in the natural human language. LLM are widely famous and known for their ability to produce contextual relevant data for any given prompt, not only does it provide us with accurate answers but also with natural language processing abilities like sentiment analysis, text translation and etc.

One of the most famous LLM existing today is OpenAI GPT and its series, which includes versions like GPT – 2, GPT – 3, 3.5 and the latest of them GPT – 4. They are specifically designed for sequence to sequence based tasks which make them the best choice for generating prompts and texts.

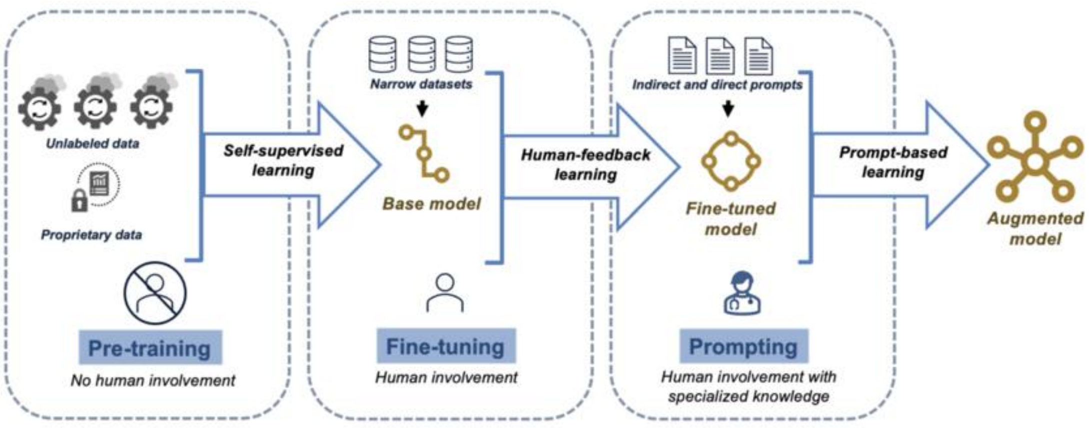


Figure 52: Architecture view of how the LLM works

Here's how the LLM work:

Step1: Tokenization

The text that is sent in as the input is transformed into smaller chunks of tokens. i.e., bigger sentences are broken down into smaller text chunks and then converted into numerical vector representations which are known as embedding's, while keeping each and every one of them unique.

Step2: Transformer Architecture

LLM is created based on the transformer architecture. This generally consists of numerous layers of various mechanisms and neural networks. One of them is self-attention mechanism this allows the model to be able to accurately weigh each words weight-age in the sentence based on different situations and also based on their relation with each other.

Step3: Encoding

The input text that is tokenised is send into the transformer layers and undergoes encoding. At each and every layer the model analyses, learns and extracts new features and insights from the input text.

Step4: Contextualisation

One Key feature of any LLM is their ability to contextualize the meaning behind the words and sentences. For example, the GPTs when prompted a question it understand the context of the question and stores it in the form of meaningful numerical vectors. And based on these they generate the answers to any type of prompts.

Step5: Decoding

Once encoding is done and using any one od the similarity search algorithms like FAISS or VAE, etc. the answers vectors are sent back to the LLM where it gets decoded back into the meaningful textual format.

Step6: Training

Generally LLM is trained up on vast amount of text data using unsupervised learning techniques, which allows the model to learn on its own and finds patterns, trends and correlations present in the data. When questions are prompted and as the user prompts questions the LLM learns from it and stores the correlations of the prompts and generated answers. This is often called feedback learning.

Step7: Fine Tuning

When required the LLM might go through fine tuning based on the domain or task specific data. For example let's say in the GPT I'm prompting a question about any topic from medical field and when this is processed and encoded, during the next step while finding the similar chunks of vectors the model fine tunes itself to only check the chunks which are or are related to medical field. By doing this it improves the model performance and also saves time.

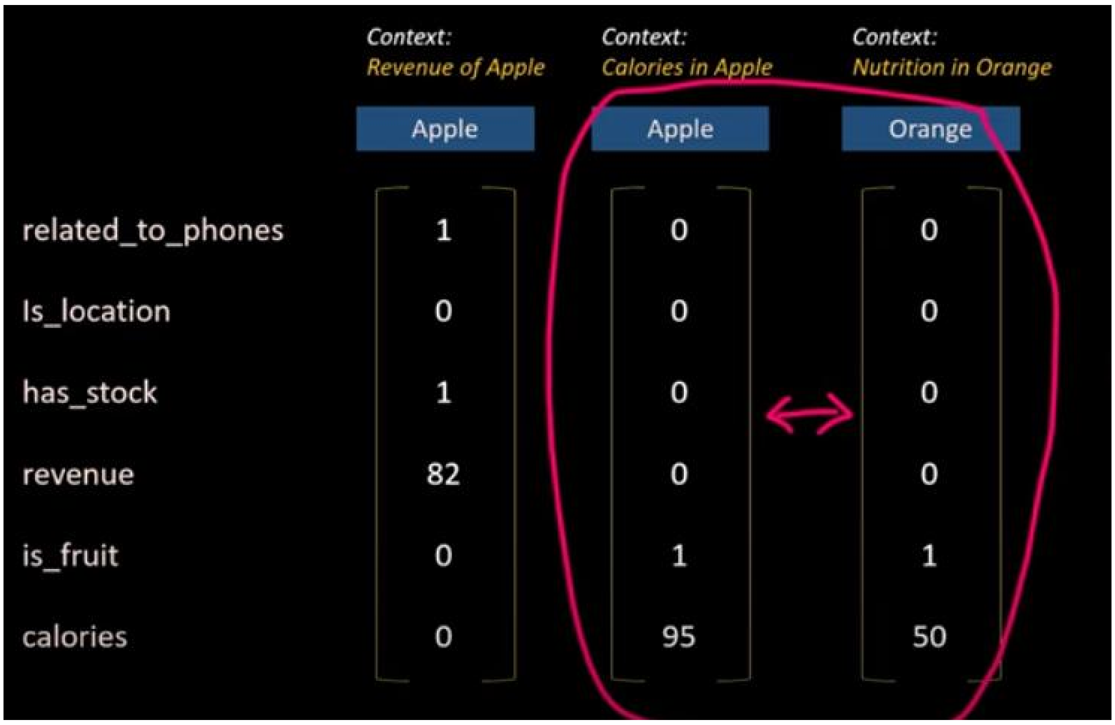
APPENDIX 2

Vector Embedding:

Vector embeddings is one of the main steps involved along side with LLM in thos project. The definition and its internal working are discussed below.

Definition:

The method of expressing and representing words or textual data in the form of numerical vector space is referred to as word embedding or text embedding, or vector embedding. The main goal of a vector embedding is to express the semantic textual content and relationships between words so that the ML algorithms can pre-process and analyse them effectively.



	Context: Revenue of Apple	Context: Calories in Apple	Context: Nutrition in Orange
	Apple	Apple	Orange
related_to_phones	1	0	0
Is_location	0	0	0
has_stock	1	0	0
revenue	82	0	0
is_fruit	0	1	1
calories	0	95	50

Figure 53: Representation of vector embeddings created based on context

Here's a rundown of the vector embedding procedure:

- **Tokenization**: The text data is represented by smaller chunks of tokens, words, sentences and sub-sentences. After that, each token is given a unique statistical numerical identification.

- **Characteristic Representation**: Each and every token is represented by a numerical vector, with each dimension signifying a token's contextual meaning. These vectors are generally initialized by using either already existing pre-trained embeddings or randomly initialized embeddings.
- **Training or Pre-trained Embeddings**: Depending on the situation and strategy, vector embeddings can be trained from scratch on large text corpora using techniques like Word2Vec, GloVe, or FastText.
- **Semantic linkages**: Based on the co-occurrence patterns of keywords in the text corpus, the embeddings are improved during training or pre-training to capture semantic links between words. In the vector space, words with comparable contexts or meanings show up as adjacent vectors.

For example, From the figure 29, we can see how apple from the sentence revenue of the apple has different embedding compared to the apple from the contextual sentence calories in apple. And also the apple and orange which are in the context of fruits has similar embeddings. This is how the embeddings of the contexts are divided and formed based on the context.

APPENDIX 3

```
import os
import streamlit as st
import pickle
import time
import pandas as pd
from langchain.chat_models import ChatOpenAI
from langchain.chains import RetrievalQAWithSourcesChain
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.document_loaders import UnstructuredURLLoader,
PyPDFLoader
from langchain.embeddings import OpenAIEmbeddings
from langchain.vectorstores import FAISS
from dotenv import load_dotenv
from tempfile import NamedTemporaryFile
import statsmodels.api as sm
from statsmodels.formula.api import ols
from sklearn.linear_model import LinearRegression
from scipy.stats import ttest_1samp, ttest_rel, ttest_ind, bartlett,
levene, wilcoxon, spearmanr, chisquare
from scipy.stats import f_oneway, pearsonr, chi2_contingency
from statsmodels.tsa.stattools import acf
import seaborn as sns
from statsmodels.stats.weightstats import ztest
from statsmodels.graphics.gofplots import qqplot
from statsmodels.tsa.seasonal import seasonal_decompose
import matplotlib.pyplot as plt
import numpy as np

load_dotenv()  # take environment variables from .env (especially
openai api key)

st.title("AnalytiBot: Research Tool ■")
st.sidebar.title("Data Input")

# Allow user to choose between URL or PDF file
data_input_choice = st.sidebar.radio("Choose Data Input Method:",
("URLs", "PDF File", "CSV File"))
```

```
if data_input_choice == "URLs":
    urls = []
    for i in range(1):
        url = st.sidebar.text_input(f"URL {i + 1}")
        urls.append(url)

    process_data_clicked = st.sidebar.button("Process URLs")

elif data_input_choice == "PDF File":
    uploaded_file = st.sidebar.file_uploader("Upload PDF File",
type="pdf")

    process_data_clicked = st.sidebar.button("Process PDF")

elif data_input_choice == "CSV File":
    uploaded_csv = st.sidebar.file_uploader("Upload CSV File",
type="csv")
    if uploaded_csv:
        st.header("Regression Analysis")
        regression_type = st.selectbox("Select Regression Type:",
("Linear Regression", "Multiple Regression"))

        if regression_type == "Linear Regression":
            st.write("Select dependent and independent variables:")
            data = pd.read_csv(uploaded_csv)
            columns =
data.select_dtypes(include='number').columns.tolist()  # Select only
numeric columns
            dependent_variable = st.selectbox("Select Dependent
Variable:", columns)
            independent_variable = st.selectbox("Select Independent
Variable:", columns)

        elif regression_type == "Multiple Regression":
            st.write("Select dependent and independent variables:")
            data = pd.read_csv(uploaded_csv)
            columns =
data.select_dtypes(include='number').columns.tolist()  # Select only
numeric columns
            dependent_variable = st.selectbox("Select Dependent
Variable:", columns)
            independent_variables = st.multiselect("Select
Independent Variables:", columns)

        all_columns = data.columns.tolist()

        # Add inputs for statistical tests
        st.header("Statistical Tests")
        significance_level = st.number_input("Enter Significance
Level (e.g., 0.05):", value=0.05, step=0.01)
        test_type = st.selectbox("Select Test Type:", (
            "t-Tests (One-sample)", "t-Tests (Independent samples)",
"t-Tests (Paired samples)",
            "Analysis of Variance (One-way)", "Analysis of Variance
(Two-way)", "Z-test",
            "Correlation Analysis", "Chi-square Test",
            "Moving Average", "Exponential Smoothing",
"Autocorrelation", "Seasonality Analysis",
            "Histogram", "Box Plot", "Q-Q Plot",
            "Pearson Correlation Coefficient", "Spearman's Rank
Correlation Coefficient",
            "Chi-square Test of Independence"))

        if test_type in ("t-Tests (One-sample)", "t-Tests
(Independent samples)", "t-Tests (Paired samples)"):
            st.subheader("T-Test Variables")
            dependent_variable_ttest = st.selectbox("Select
Dependent Variable for T-Test:", columns,
key="ttest_dependent_variable")
            independent_variable_ttest = st.selectbox("Select
Independent Variable for T-Test:", columns,
key="ttest_independent_variable")
            equal_variances = st.checkbox("Equal Variances?")
        elif test_type in ("Analysis of Variance (One-way)",
"Analysis of Variance (Two-way)"):
            st.subheader("ANOVA Variables")
            dependent_variable_anova = st.selectbox("Select
Dependent Variable for ANOVA:", columns,

key="anova_dependent_variable")
            # One-way ANOVA selections (if applicable)
            if test_type == "Analysis of Variance (One-way)":
                independent_variable_anova = st.selectbox("Select
Independent Variable for ANOVA:", all_columns,

key="anova_independent_variable")
            # Two-way ANOVA selections (if applicable)
            elif test_type == "Analysis of Variance (Two-way)":
                independent_variable1_anova = st.selectbox("Select
1st Independent Variable for ANOVA:", all_columns,

key="anova_independent_variable1")
                independent_variable2_anova = st.selectbox("Select
2nd Independent Variable for ANOVA:",

all_columns,

key="anova_independent_variable2")  # Ensure unique variables

        elif test_type == "Z-test":
            st.subheader("Z-test")
            variable1_ztest = st.selectbox("Select Variable 1 for Z-
test:", columns, key="ztest_variable1")
            variable2_ztest = st.selectbox("Select Variable 2 for Z-
test:", columns, key="ztest_variable2")

        elif test_type == "Correlation Analysis":
            st.subheader("Correlation Analysis")
            selected_variables_corr = st.multiselect("Select
Variables for Correlation Analysis:", columns)

        elif test_type == "Chi-square Test":
            st.subheader("Chi-square Test")
            # Allow selecting multiple variables
            selected_variables = st.multiselect("Select Variables
for Chi-square Test:", all_columns)



        elif test_type == "Moving Average":
            st.subheader("Moving Average")
            variable1_Movi_Avg = st.selectbox("Select Variable for
Performing Moving Avg:", columns,

key="Movi_Avg_variable1")
            window_size_ma = st.number_input("Enter Window Size for
Moving Average:", min_value=1, max_value=len(data),
                                                 value=10)
        elif test_type == "Exponential Smoothing":
            st.subheader("Exponential Smoothing")
            variable1_expo_smooth = st.selectbox("Select Variable
for Performing Moving Avg:", columns,

key="Expo_Smooth_variable1")
            smoothing_level = st.number_input("Enter Smoothing Level
for Exponential Smoothing:", min_value=0.01,
                                                  max_value=1.0,
value=0.5)
        elif test_type == "Autocorrelation":
            st.subheader("Autocorrelation")
            selected_variables_acf = st.multiselect("Select
Variable(s) for Autocorrelation:", columns)
            lag_acf = st.number_input("Enter Lag for
Autocorrelation:", min_value=0, max_value=len(data) - 1, value=1)

        elif test_type == "Seasonality Analysis":
            st.subheader("Seasonality Analysis")
            time_series_data = st.file_uploader("Upload Time Series
Data (CSV file)", type="csv")
            seasonality_frequency = st.selectbox("Select Seasonality
Frequency:", ["Daily", "Weekly", "Monthly"])
            visualization_method = st.selectbox("Select
Visualization Method:", ["Line Plot", "Seasonal Decomposition"])

        elif test_type == "Histogram":
            st.subheader("Histogram")
            variable_for_histogram = st.selectbox("Select Variable
for Histogram:", columns)
            num_bins = st.number_input("Number of Bins:",
min_value=1, value=10)

        elif test_type == "Box Plot":
            st.subheader("Box Plot")
            variable_for_box_plot = st.selectbox("Select Variable
for Box Plot:", columns)
            group_by_variable = st.selectbox("Group Data by Variable
(Optional):", [None] + columns)

        elif test_type == "Q-Q Plot":
            st.subheader("Q-Q Plot")
            variable_for_qq_plot = st.selectbox("Select Variable for
Q-Q Plot:", columns)
            distribution_to_compare = st.selectbox("Distribution to
Compare Against:", ["Normal", "Uniform"])


        elif test_type == "Pearson Correlation Coefficient":
            st.subheader("Pearson Correlation Coefficient")
            variable1_pearson = st.selectbox("Select Variable 1 for
Pearson Correlation Coefficient:", columns,
                                              key="pearson_variable1")
            variable2_pearson = st.selectbox("Select Variable 2 for
Pearson Correlation Coefficient:", columns,
                                                     key="pearson_variable2")

        elif test_type == "Spearman's Rank Correlation Coefficient":
            st.subheader("Spearman's Rank Correlation Coefficient")
            variable1_spearman = st.selectbox("Select Variable 1 for
Spearman's Rank Correlation Coefficient:", columns,

key="spearman_coeff_variable1")
            variable2_spearman = st.selectbox("Select Variable 2 for
Spearman's Rank Correlation Coefficient:", columns,

key="spearman_coeff_variable2")

        elif test_type == "Chi-square Test of Independence":
            st.subheader("Chi-square Test of Independence")
            variable1_chi_square = st.selectbox("Select Variable 1
for Chi-square Test of Independence:", all_columns,

key="chi_square_independence_variable1")
            variable2_chi_square = st.selectbox("Select Variable 2
for Chi-square Test of Independence:", all_columns,

key="chi_square_independence_variable2")

    process_data_clicked = st.sidebar.button("Process CSV")

elif data_input_choice == "Text File":
    uploaded_txt = st.sidebar.file_uploader("Upload Text File",
type="txt")

    process_data_clicked = st.sidebar.button("Process Text File")

file_path = "faiss_store_openai.pkl"

main_placeholder = st.empty()
llm = ChatOpenAI(model_name="gpt-3.5-turbo-16k", temperature=0.3)
```

```
# Use session_state to store previous prompts and questions
if "question_answers" not in st.session_state:
    st.session_state["question_answers"] = {}

if process_data_clicked:
    if data_input_choice == "URLs":
        # load data from URLs
        loader = UnstructuredURLLoader(urls=urls)
        main_placeholder.text("Data Loading...Started...✅✅✅")
        data = loader.load()

        # split data
        text_splitter = RecursiveCharacterTextSplitter(
            separators=['\n\n', '\n', '.', ','],
            chunk_size=1000
        )
        main_placeholder.text("Text Splitter...Started...✅✅✅")
        docs = text_splitter.split_documents(data)

        # create embeddings and save it to FAISS index
        embeddings = OpenAIEmbeddings()
        vectorstore_openai = FAISS.from_documents(docs, embeddings)
        main_placeholder.text("Embedding Vector Started
Building...✅✅✅")
        time.sleep(2)

        # Save the FAISS index to a pickle file
        with open(file_path, "wb") as f:
            pickle.dump(vectorstore_openai, f)


    elif data_input_choice == "PDF File":
        # load data from PDF file
        if uploaded_file:
            bytes_data = uploaded_file.read()
            with NamedTemporaryFile(delete=False) as tmp:  # open a
named temporary file
```

```
                tmp.write(bytes_data)  # write data from the
uploaded file into it
                data = PyPDFLoader(tmp.name).load()  # <---- now it
works!
            os.remove(tmp.name)

            # split data
            text_splitter = RecursiveCharacterTextSplitter(
                separators=['\n\n', '\n', '.', ','],
                chunk_size=1000
            )
            main_placeholder.text("Text Splitter...Started...✅✅✅")
            docs = text_splitter.split_documents(data)

            # create embeddings and save it to FAISS index
            embeddings = OpenAIEmbeddings()
            vectorstore_openai = FAISS.from_documents(docs,
embeddings)
            main_placeholder.text("Embedding Vector Started
Building...✅✅✅")
            time.sleep(2)

            # Save the FAISS index to a pickle file
            with open(file_path, "wb") as f:
                pickle.dump(vectorstore_openai, f)

query = main_placeholder.text_input("Question: ")
if query:
    if os.path.exists(file_path):
        with open(file_path, "rb") as f:
            vectorstore = pickle.load(f)
            chain = RetrievalQAWithSourcesChain.from_llm(llm=llm,
retriever=vectorstore.as_retriever())
            result = chain({"question": query},
return_only_outputs=True)
            # result will be a dictionary of this format -->
{"answer": "", "sources": [] }
```

```
            st.header("Answer")
            st.write(result["answer"])

            # Store the question and its answer in the session_state
            st.session_state["question_answers"][query] =
result["answer"]

            # Display sources, if available
            sources = result.get("sources", "")
            if sources:
                st.subheader("Sources:")
                sources_list = sources.split("\n")  # Split the
sources by newline
                for source in sources_list:
                    st.write(source)

# Display all asked questions with their answers
if st.session_state["question_answers"]:
    st.header("Asked Questions with Answers")
    for question, answer in
st.session_state["question_answers"].items():
        st.write(f"Question: {question}")
        st.write(f"Answer: {answer}")

# Descriptive Analysis and Regression Analysis
if process_data_clicked and data_input_choice == "CSV File":
    st.header("Descriptive Analysis")
    st.write("Summary Statistics:")
    st.write(data.describe())

    if regression_type == "Linear Regression" and dependent_variable
and independent_variable:
        X = data[independent_variable].values.reshape(-1, 1)
        y = data[dependent_variable].values.reshape(-1, 1)

        model = LinearRegression()
        model.fit(X, y)
        predictions = model.predict(X)

        st.write("Regression Results:")
        st.write(f"Intercept: {model.intercept_}")
        st.write(f"Coefficient: {model.coef_[0]}")

        # Plotting the best-fit line
        st.header("Regression Plot")
        plt.scatter(X, y, label='Actual data points')
        plt.plot(X, predictions, color='red', label='Best-fit line')
        plt.xlabel('Independent Variable')
        plt.ylabel('Dependent Variable')
        plt.title('Linear Regression Analysis')
        plt.legend()
        st.pyplot(plt)

    elif regression_type == "Multiple Regression" and
dependent_variable and independent_variables:
        X = data[independent_variables]
        y = data[dependent_variable]
        X = sm.add_constant(X)  # Add constant for intercept
        model = sm.OLS(y, X).fit()
        predictions = model.predict(X)

        st.write("Regression Results:")
        st.write(model.summary())

        st.write("Regression Equation:")
        st.write(model.params)

        # Plotting the best-fit line
        st.header("Regression Plot")
        plt.scatter(y, predictions, label='Actual vs Predicted')
        plt.xlabel('Actual')
        plt.ylabel('Predicted')
        plt.title('Multiple Regression Analysis')
        plt.legend()
        st.pyplot(plt)


    # Perform statistical tests
    if test_type == "t-Tests (One-sample)":
        if dependent_variable_ttest and independent_variable_ttest:
            t_statistic, p_value =
ttest_1samp(data[independent_variable_ttest],
data[dependent_variable_ttest].mean())
            st.write("T-Statistic:", t_statistic)
            st.write("P-Value:", p_value)
            st.write("Conclusion:",
                     "Reject Null Hypothesis" if p_value <
significance_level else "Fail to Reject Null Hypothesis")

    elif test_type == "t-Tests (Independent samples)":
        if dependent_variable_ttest and independent_variable_ttest:
            t_statistic, p_value =
ttest_ind(data[independent_variable_ttest],
data[dependent_variable_ttest],

equal_var=equal_variances)
            st.write("T-Statistic:", t_statistic)
            st.write("P-Value:", p_value)
            st.write("Conclusion:",
                     "Reject Null Hypothesis" if p_value <
significance_level else "Fail to Reject Null Hypothesis")

    elif test_type == "t-Tests (Paired samples)":
        if dependent_variable_ttest and independent_variable_ttest:
            t_statistic, p_value =
ttest_rel(data[independent_variable_ttest],
data[dependent_variable_ttest])
            st.write("T-Statistic:", t_statistic)
            st.write("P-Value:", p_value)
            st.write("Conclusion:",
                     "Reject Null Hypothesis" if p_value <
significance_level else "Fail to Reject Null Hypothesis")

    if test_type == "Analysis of Variance (One-way)":

        if dependent_variable_anova and independent_variable_anova:
            model = ols(f'{dependent_variable_anova} ~
C({independent_variable_anova})',
                        data).fit()  # Use categorical encoding for
one-way ANOVA
            anova_table = sm.stats.anova_lm(model, typ=1)
            test_result = f_oneway(data[independent_variable_anova],
data[dependent_variable_anova])
            st.write("ANOVA Table (One-way):")
            st.write(anova_table)
            st.write("F-Statistic:", test_result[0])
            st.write("P-Value:", test_result[1])
            st.write("Conclusion:", "Reject Null Hypothesis" if
test_result[

1] < significance_level else "Fail to Reject Null Hypothesis")


    elif test_type == "Analysis of Variance (Two-way)":
        if dependent_variable_anova and independent_variable1_anova
and independent_variable2_anova:
            model = ols(
                f'{dependent_variable_anova} ~
C({independent_variable1_anova}) + C({independent_variable2_anova})',
                data).fit()  # Use categorical encoding for two-way
ANOVA
            anova_table = sm.stats.anova_lm(model, typ=3)  # typ=3
for two-way interaction
            test_result = f_oneway(data[dependent_variable_anova],
data[independent_variable1_anova],
                                   data[independent_variable2_anova])
# Include all factors for two-way test
            st.write("ANOVA Table (Two-way):")
            st.write(anova_table)
            st.write("F-Statistic:", test_result[0])
            st.write("P-Value:", test_result[1])
            st.write("Conclusion:", "Reject Null Hypothesis" if
test_result[

1] < significance_level else "Fail to Reject Null Hypothesis")

    # Perform Z-test
    if test_type == "Z-test":
        if variable1_ztest and variable2_ztest:
            # Extract data for selected variables
            data1 = data[variable1_ztest]
            data2 = data[variable2_ztest]

            # Perform Z-test (assuming data1 and data2 are normally
distributed)
            z_statistic, p_value = ztest(data1, data2)

            # Print Z-test results
            st.write("Z-Statistic:", z_statistic)
            st.write("P-Value:", p_value)

            # Interpret Z-test results
            if p_value < significance_level:
                st.write("Conclusion: Reject Null Hypothesis")
            else:
                st.write("Conclusion: Fail to Reject Null
Hypothesis")

    elif test_type == "Correlation Analysis":
        if selected_variables_corr:
            # Compute correlation coefficients
            correlation_matrix = data[selected_variables_corr].corr()
            # Display correlation table
            st.write("Correlation Table:")
            st.write(correlation_matrix)
            # Generate correlation matrix plot
            st.write("Correlation Matrix Plot:")
            plt.figure(figsize=(10, 8))
            sns.heatmap(correlation_matrix, annot=True,
cmap='coolwarm', fmt=".2f", linewidths=0.5)
            st.pyplot(plt)


    elif test_type == "Chi-square Test":
        if selected_variables:
            for variable in selected_variables:
                # Get the observed frequencies for each category
                observed_values =
data[variable].value_counts().values

                # Assuming a uniform distribution as the expected
distribution (adjust if needed)
                expected_values = np.array([len(data) /
len(data[variable].unique())] * len(
                    data[variable].unique()))

                # Perform Chi-square test
                chi2_stat, p_value = chisquare(observed_values,
expected_values)

                #  Display results
                st.write(f"Chi-square Test for {variable}:")
                st.write(f"Chi-square Statistic: {chi2_stat}")
                st.write(f"P-Value: {p_value}")
                st.write("Conclusion:",
                        "Reject Null Hypothesis" if p_value <
significance_level else "Fail to Reject Null Hypothesis")

    elif test_type == "Moving Average":
        if window_size_ma:
            data['Moving Average'] =
data[variable1_Movi_Avg].rolling(window=window_size_ma).mean()
```

```
            st.write("Moving Average:")
            st.write(data)

    elif test_type == "Exponential Smoothing":
        if smoothing_level:
            data['Exponential Smoothing'] =
data[variable1_expo_smooth].ewm(alpha=smoothing_level).mean()
            st.write("Exponential Smoothing:")
            st.write(data)

    elif test_type == "Seasonality Analysis":
        if time_series_data:
            time_series = pd.read_csv(time_series_data)
            if visualization_method == "Line Plot":
                st.write("Line Plot:")
                st.line_chart(time_series)
            elif visualization_method == "Seasonal Decomposition":
                decomposition = seasonal_decompose(time_series,
model='additive', freq=seasonality_frequency.lower())
                st.write("Seasonal Decomposition:")
                st.write(decomposition.plot())
                st.write(decomposition.seasonal)

    elif test_type == "Autocorrelation":
        if selected_variables_acf and lag_acf:
            st.write("Autocorrelation Results:")
            for variable in selected_variables_acf:
                acf_result = acf(data[variable], nlags=lag_acf)
                st.write(f"Autocorrelation for {variable}:")
                st.write(acf_result)

                # Plot autocorrelation function
                st.write(f"Autocorrelation Plot for {variable}:")
                plt.figure(figsize=(10, 6))
                # Plotting the Autocorrelation plot.
                plt.acorr(data[variable], maxlags=10)
                plt.xlabel("Lag")
                plt.ylabel("Autocorrelation")
                plt.title(f"Autocorrelation Function for {variable}")
                st.pyplot(plt)

    elif test_type == "Histogram":
        if variable_for_histogram:
            st.write("Histogram for {variable_for_histogram}:")
            plt.figure(figsize=(10, 6))
            plt.hist(data[variable_for_histogram], bins=num_bins)
            plt.xlabel(variable_for_histogram)
            plt.ylabel('Frequency')
            st.pyplot(plt)

    elif test_type == "Box Plot":
        if variable_for_box_plot:
            st.write("Box Plot:")
            plt.figure(figsize=(10, 6))
            if group_by_variable:
                data.boxplot(column=variable_for_box_plot,
by=group_by_variable)
            else:
                data.boxplot(column=variable_for_box_plot)
            plt.ylabel(variable_for_box_plot)
            st.pyplot(plt)

    elif test_type == "Q-Q Plot":
        if variable_for_qq_plot:
            st.write("Q-Q Plot:")
            plt.figure(figsize=(10, 6))
            if distribution_to_compare == "Normal":
                qqplot(data[variable_for_qq_plot], line='s')
            elif distribution_to_compare == "Uniform":
                qqplot(data[variable_for_qq_plot], line='q')
            st.pyplot(plt)

    elif test_type == "Pearson Correlation Coefficient":
        if variable1_pearson and variable2_pearson:
            correlation_coefficient =
pearsonr(data[variable1_pearson], data[variable2_pearson])[0]
            st.write("Pearson Correlation Coefficient:",
correlation_coefficient)
            st.write("Conclusion:", "Reject Null Hypothesis" if abs(
                correlation_coefficient) > significance_level else
"Fail to Reject Null Hypothesis")

    elif test_type == "Spearman's Rank Correlation Coefficient":
        if variable1_spearman and variable2_spearman:
            correlation_coefficient =
spearmanr(data[variable1_spearman], data[variable2_spearman])[0]
            st.write("Spearman's Rank Correlation Coefficient:",
correlation_coefficient)
            st.write("Conclusion:", "Reject Null Hypothesis" if abs(
                correlation_coefficient) > significance_level else
"Fail to Reject Null Hypothesis")

    elif test_type == "Chi-square Test of Independence":
        if variable1_chi_square and variable2_chi_square:
            contingency_table =
pd.crosstab(data[variable1_chi_square], data[variable2_chi_square])
            chi2, p, dof, expected =
chi2_contingency(contingency_table)
            st.write("Chi-square Test of Independence:")
            st.write("Chi-square Statistic:", chi2)
            st.write("P-Value:", p)
            st.write("Degrees of Freedom:", dof)
            st.write("Expected Frequencies:")
            st.write(expected)
            st.write("Conclusion:",
                     "Reject Null Hypothesis" if p <
significance_level else "Fail to Reject Null Hypothesis")
```

REFERENCES

[1]. Adamopoulou, E., & Moussiades, L. (2020). Chatbots: History, technology, and applications. *Machine Learning with applications*, *2*, 100006.

[2]. Zemčík, M. T. (2019). A brief history of chatbots. *DEStech Transactions on Computer Science and Engineering*, *10*.

[3]. Adamopoulou, E., & Moussiades, L. (2020). An overview of chatbot technology. In IFIP international conference on artificial intelligence applications and innovations (pp. 373-383). Springer, Cham.

[4]. Luo, B., Lau, R. Y., Li, C., & Si, Y. W. (2022). A critical review of state-of-the-art chatbot designs and applications. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 12(1), e1434.

[5]. Fotheringham, D., & Wiles, M. A. (2023). The effect of implementing chatbot customer service on stock returns: An event study analysis. Journal of the Academy of Marketing Science, 51(4), 802-822.

[6]. Pandya, K., & Holia, M. (2023). Automating Customer Service using LangChain: Building custom open-source GPT Chatbot for organizations. arXiv preprint arXiv:2310.05421.

[7]. Jeong, C. (2023). A Study on the Implementation of Generative AI Services Using an Enterprise Data-Based LLM Application Architecture. arXiv preprint arXiv:2309.01105.

[8]. Bucur, M. (2023). Exploring Large Language Models and Retrieval Augmented Generation for Automated Form Filling (Bachelor's thesis, University of Twente).

[9]. Mansurova, A., Nugumanova, A., & Makhambetova, Z. (2023). Development of a question answering chatbot for blockchain domain. Scientific Journal of Astana IT University, 27-40.

[10]. Basit, A., Hussain, K., Hanif, M. A., & Shafique, M. (2024). MedAide: Leveraging Large Language Models for On-Premise Medical Assistance on Edge Devices. arXiv preprint arXiv:2403.00830.

[11]. Nischal, C. N., Sachin, T., Vivek, B. K., & Taranath, K. G. (2020). Developing a chatbot using machine learning. International Journal of Research in Engineering, Science and Management, 3(8), 40-43.

[12]. Jeong, C. (2023). Generative AI service implementation using LLM application architecture: based on RAG model and LangChain framework. Journal of Intelligence and Information Systems, 29(4), 129-164.

[13]. Bratić, D., Šapina, M., Jurečić, D., & Žiljak Gršić, J. (2024). Centralized Database Access: Transformer Framework and LLM/Chatbot Integration-Based Hybrid Model. Applied System Innovation, 7(1), 17.

[14]. Følstad, A., Araujo, T., Law, E. L. C., Brandtzaeg, P. B., Papadopoulos, S., Reis, L., ... & Luger, E. (2021). Future directions for chatbot research: an interdisciplinary research agenda. *Computing*, *103*(12), 2915-2942.

[15]. Bernard, J., & Bernard, J. (2016). Python data analysis with pandas. Python Recipes Handbook: A Problem-Solution Approach, 37-48.

[16]. Alekseev, D., Shagalova, P., & Sokolova, E. (2021). Development of a chatbot using machine learning algorithms to automate educational processes. In Proceedings of the 31st International Conference on Computer Graphics and Vision (GraphiCon 2021)–C (pp. 1104-1113).

[17]. Ahrens, J., Geveci, B., Law, C., Hansen, C., & Johnson, C. (2005). 36-paraview: An end-user tool for large-data visualization. The visualization handbook, 717, 50038-1.

[18]. Mauri, M., Elli, T., Caviglia, G., Uboldi, G., & Azzi, M. (2017, September). RAWGraphs: a visualisation platform to create open outputs. In Proceedings of the 12th biannual conference on Italian SIGCHI chapter (pp. 1-5).

[19]. Yau, N. (2011). Visualize this: the FlowingData guide to design, visualization, and statistics. John Wiley & Sons.

[20]. Tamrakar, R., & Wani, N. (2021). Design and development of CHATBOT: A review. ResearchGate, Apr.