Submitted By

Name	**Utkarsh Chaurasia**
Role	**CV Engineer Intern**
Branch	**Information Technology**
Semester	**7th**
College	**NIT Raipur**
Submission Date	**15-11-2023**

Problem Statement:

Objective:
To develop an application that can perform statistical analysis of CSV files using the Prompt and Llama-2 model, and generate plots based on the results.

Assignment:
Develop an application that can perform statistical analysis of CSV files using the Prompt and LLM model, and generate plots based on the results. The application should be able to:

- Read and parse CSV files
- Perform basic statistical analysis on the data, such as calculating the mean, median, mode, standard deviation, and correlation coefficient
- Generate plots of the data, such as histograms, scatter plots, and line plots etc.,
- Answer questions about the data in a comprehensive and informative way

Requirements:
The application should be developed using the following technologies. Feel free to use any method.

- Python, Prompt
- Any LLM
- Matplotlib
- Pandas

Deliverables:

- The deliverables for this project include:
- A working application that can perform statistical analysis of CSV files and generate plots
- A demonstration of the application in action
- A report that describes the design and implementation of the application

Solution:

“DataStatsViz - Statistical Analysis and Visualization Application”

I. Introduction

Project Title: DataStatsViz

Objective:

DataStatsViz is a comprehensive statistical analysis and visualization application designed to empower users in extracting valuable insights from their datasets. Leveraging advanced technologies such as Python, Prompt, Llama-2 model, Matplotlib, and Pandas, this application provides a seamless and intuitive platform for data exploration.

Dataset Overview

1 Introduction to the Iris Flower Dataset

The Iris dataset is a classic dataset in the field of machine learning and statistics, introduced by the British biologist and statistician Ronald A. Fisher in 1936. It comprises measurements of four features of iris flowers: sepal length, sepal width, petal length, and petal width. Each observation in the dataset corresponds to a specific species of iris: setosa, versicolor, or virginica.

2 Dataset Structure

The Iris dataset is structured in tabular form, with rows representing individual flower samples and columns corresponding to different features. The key features include:

- **Sepal Length (Cm):** The length of the iris flower's sepal.
- **Sepal Width (Cm):** The width of the iris flower's sepal.
- **Petal Length (Cm):** The length of the iris flower's petal.
- **Petal Width (Cm):** The width of the iris flower's petal.
- **Species:** The species of the iris flower, classified into three categories: setosa, versicolor, and virginica.

3 Characteristics of the Iris Flower Dataset

3.1 Size and Dimensions

The dataset typically contains 150 samples, with each species contributing 50 observations. Each sample includes measurements for the four features mentioned above.

3.2 Target Variable (Species)

The 'Species' column serves as the target variable, making this a supervised learning dataset. The goal of various analyses is often to understand the relationships between the features and predict the species based on these measurements.

4 Significance of the Iris Dataset

The Iris dataset is widely used in academia and industry for educational purposes, testing algorithms, and benchmarking model performance. Its simplicity and clarity make it an excellent starting point for exploring various statistical and machine learning concepts.

II. Design Overview

1. Data Processing

1.1 Reading and Parsing CSV Files

The application utilizes Pandas to efficiently read and parse CSV files, ensuring the dataset is ready for subsequent analysis.

```
import pandas as pd

# Reading the CSV file
df = pd.read_csv("Iris.csv")
```

1.2 Checking Data Integrity

Data integrity is maintained by checking for missing values and handling duplicates.

```
# Checking missing values
df.isnull().sum()

# Checking duplicates
data = df.drop_duplicates(subset="Species")
```

2. Statistical Analysis

2.1 Descriptive Statistics

Statistical analysis includes calculating mean, median, mode, standard deviation, and correlation coefficient.

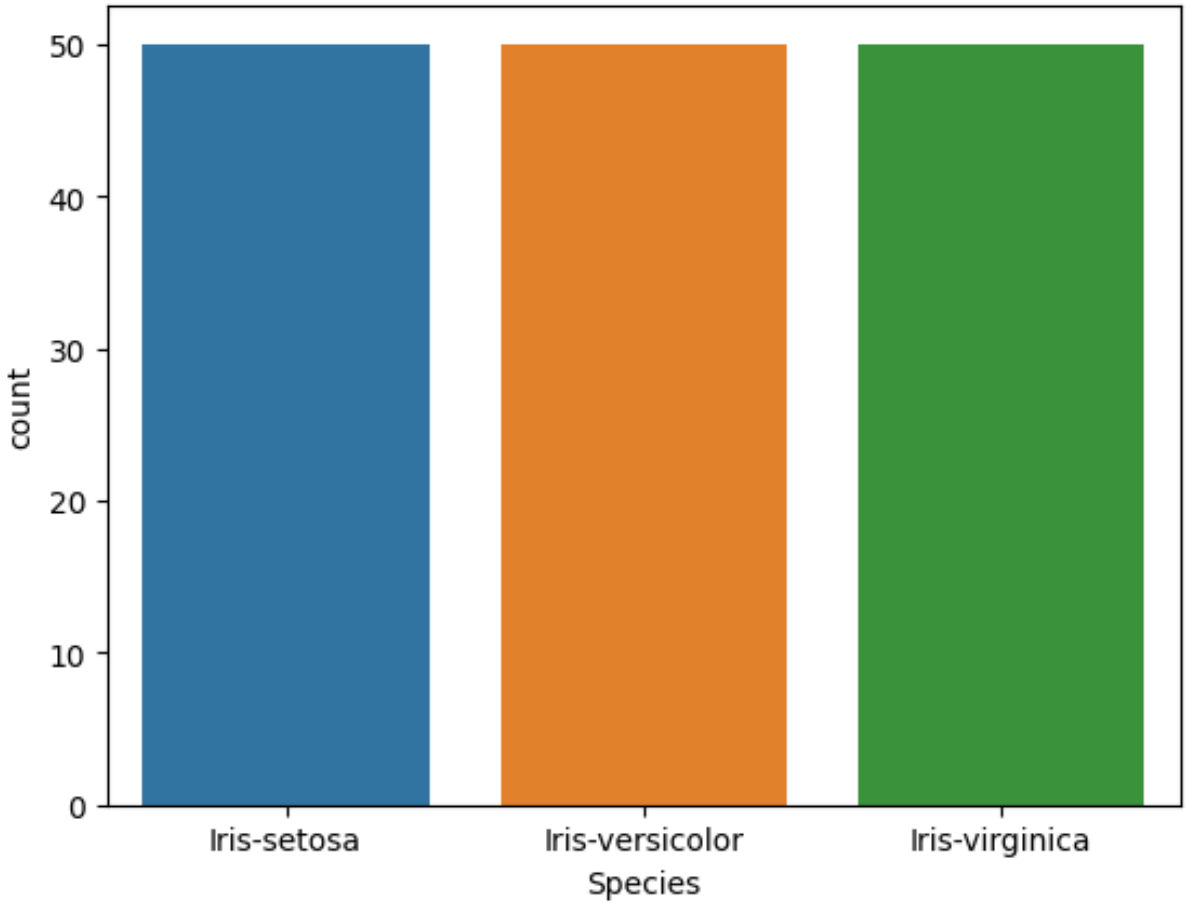
```
# Calculate the mean, median, mode, std dev, and correlation
mean = df.mean()
median = df.median()
mode = df.mode()
std_dev = df.std()
corr = df.corr()
```

3. Data Visualization

3.1 Count Plot

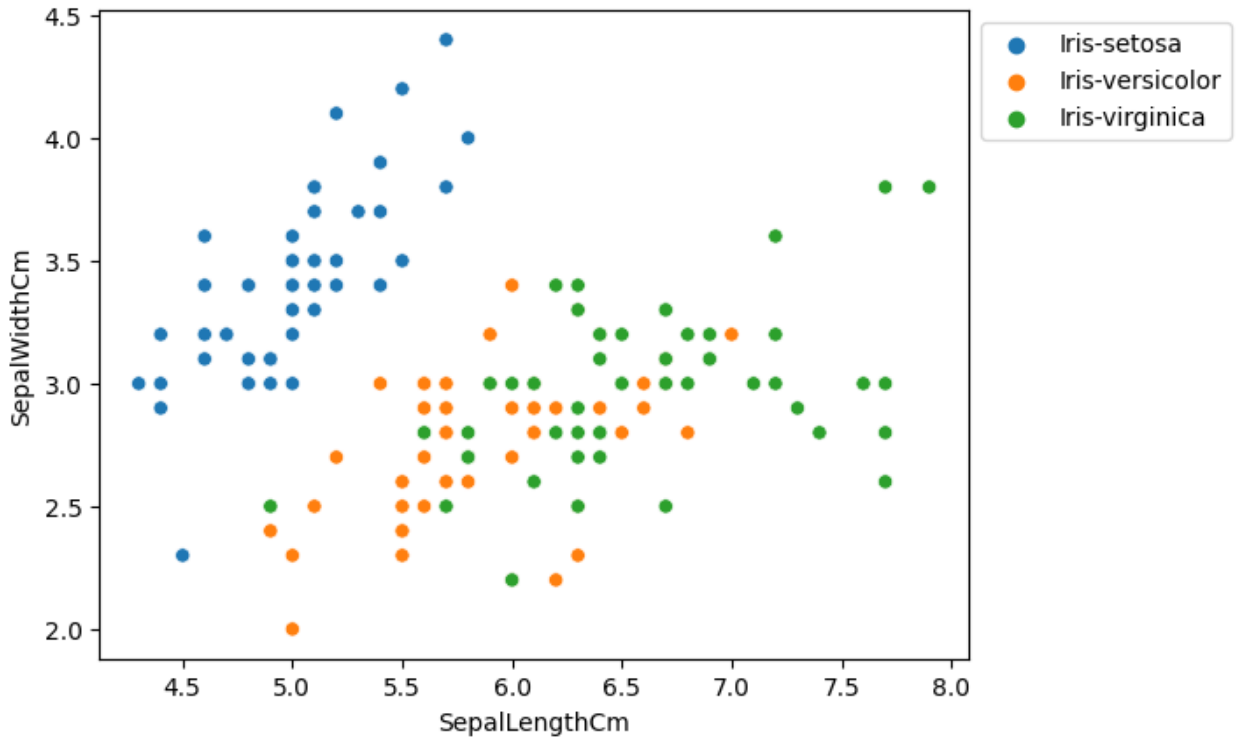
```
import seaborn as sns
import matplotlib.pyplot as plt

# Count plot of Species
sns.countplot(x='Species', data=df)
plt.show()
```



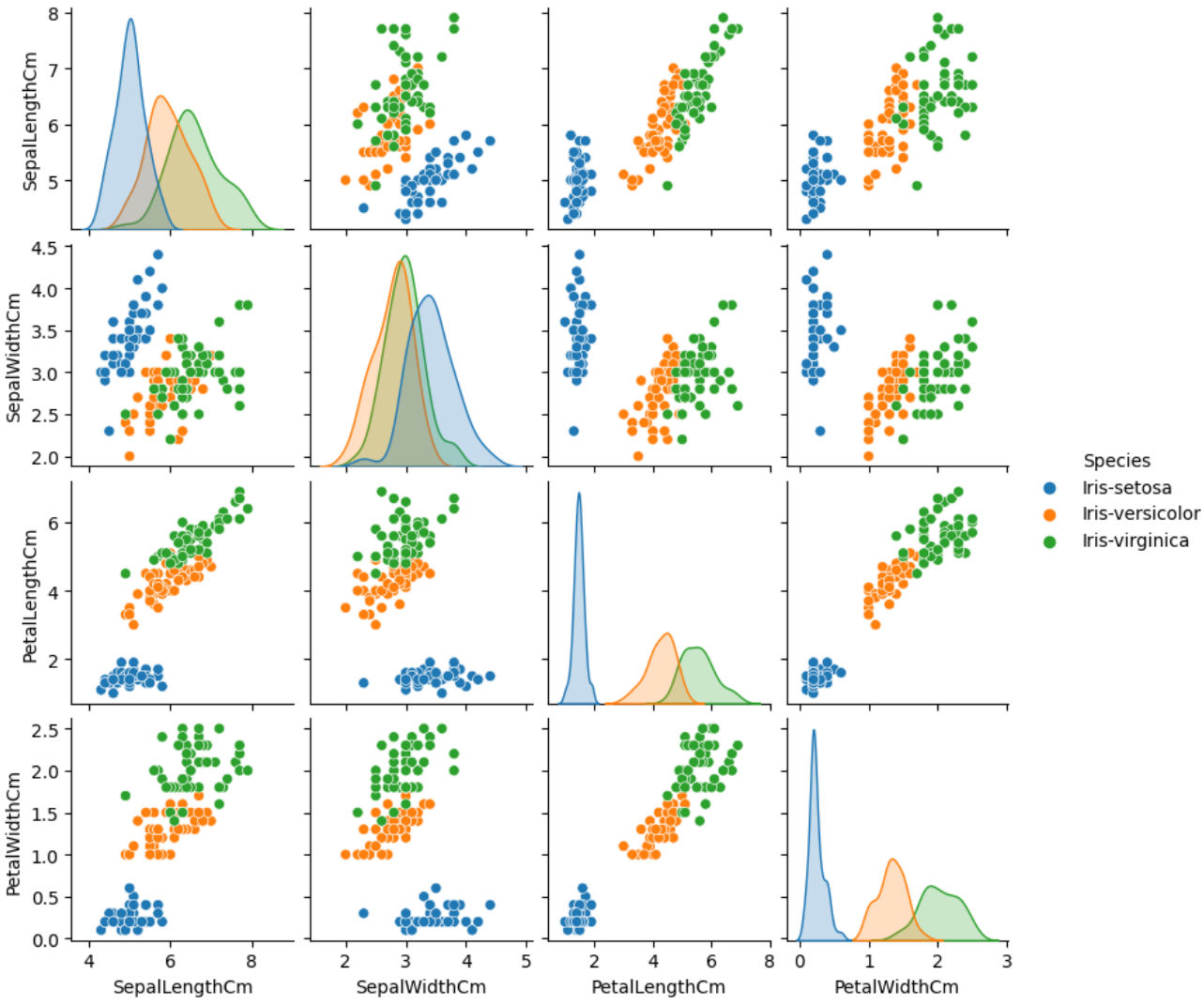
3.2 Scatter Plot

```
# Scatter plot of Sepal Length and Sepal Width
sns.scatterplot(x='SepalLengthCm', y='SepalWidthCm', hue='Species', data=df)
plt.legend(bbox_to_anchor=(1, 1), loc=2)
plt.show()
```



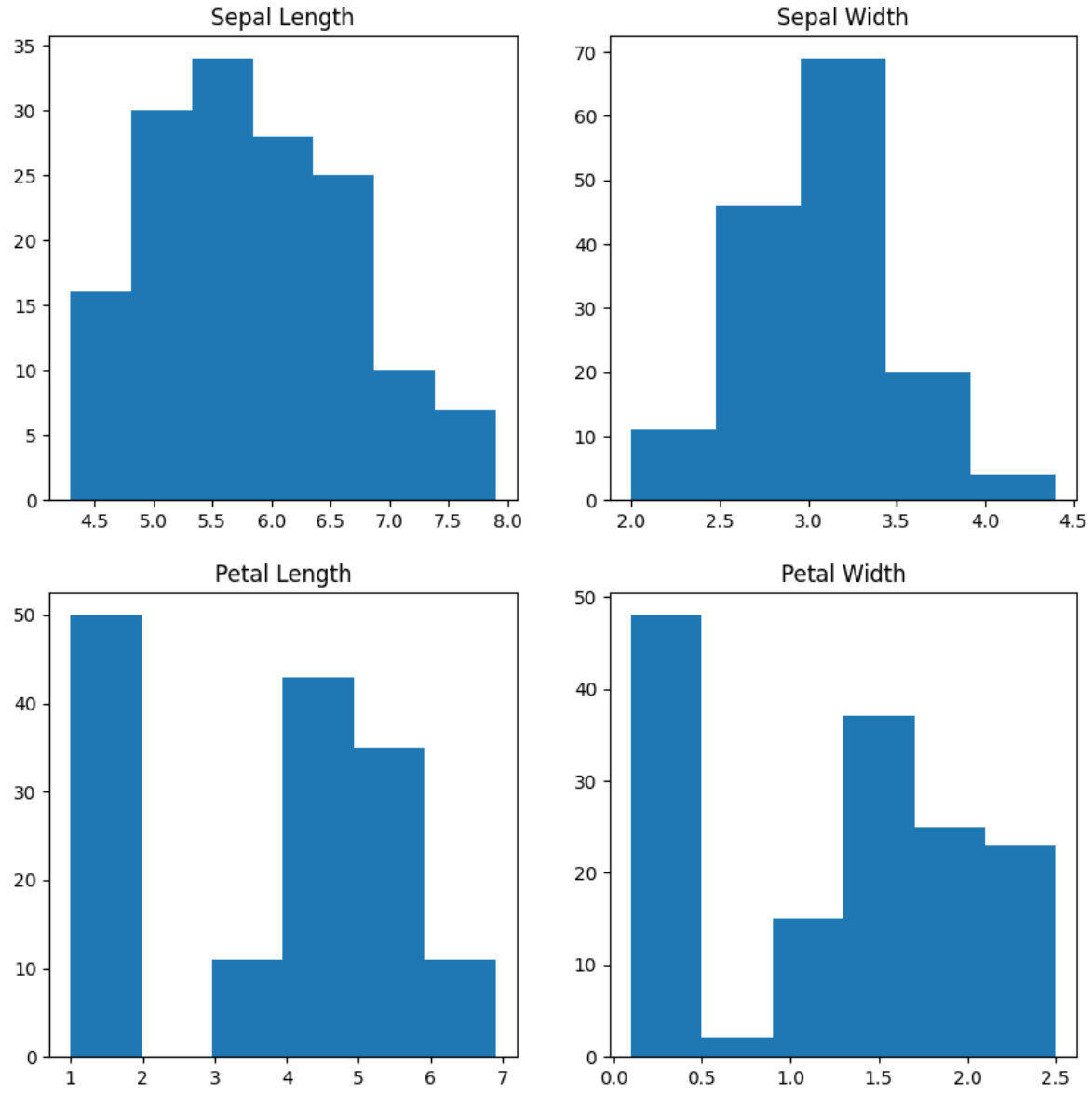
3.3 Pair Plot

```
# Pair plot of all features
sns.pairplot(df.drop(['Id'], axis=1), hue='Species', height=2)
plt.show()
```



3.4 Histograms with Distplot Plot

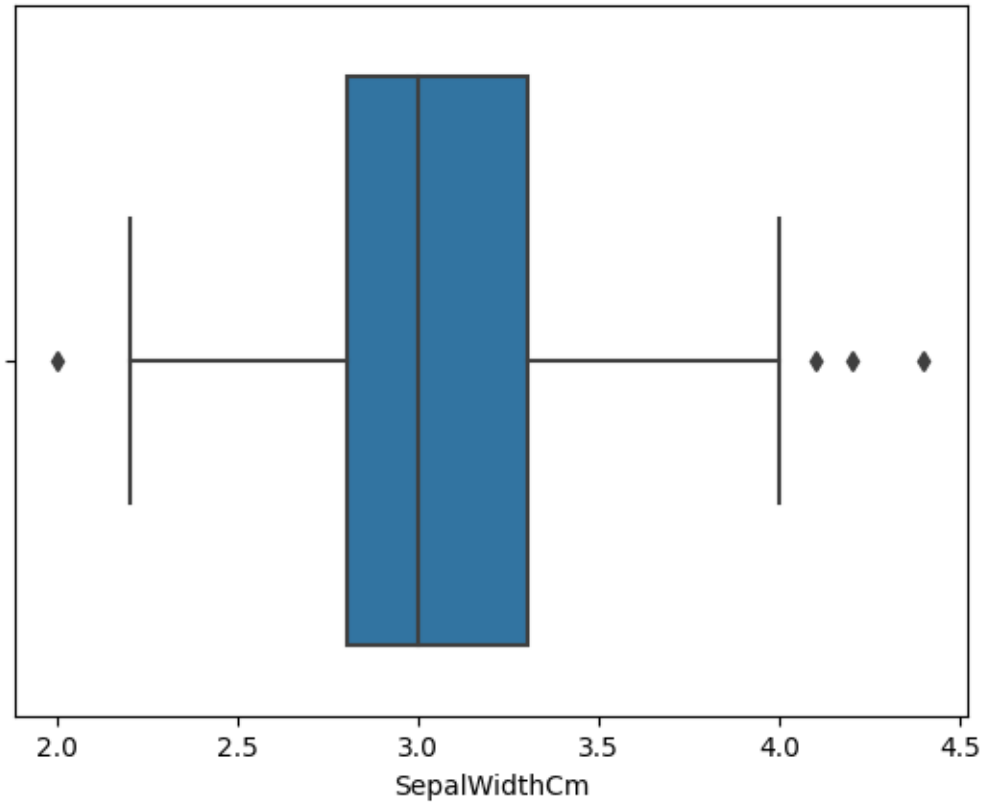
```
# Distplot plot for various features
plot = sns.FacetGrid(df, hue="Species")
plot.map(sns.distplot, "SepalLengthCm").add_legend()
# Repeat for other features
plt.show()
```



4. Outlier Handling

Outliers are detected using boxplots and handled for improved data quality.

```
# Boxplot for outlier detection
sns.boxplot(x='SepalWidthCm', data=df)
```



5. Llama-2 Model Integration

Integration of OpenAI's Llama-2 model facilitates natural language interaction, allowing users to pose questions about the dataset.

```
from transformers import AutoModelForQuestionAnswering, AutoTokenizer

# Load the Llama-2 model
tokenizer, model = load_llama_2_model()

# Ask a question
question = input('Enter your question: ')
answer = answer_question(tokenizer, model, question, df.to_string())
print(f'Answer: {answer}')
```

III. Implementation

1. Technology Stack

- **Python:** The primary programming language.
- **Prompt:** Utilized for natural language interaction.
- **LLM (Llama-2 Model):** OpenAI's language model.
- **Matplotlib:** For generating plots.
- **Pandas:** For efficient data manipulation and analysis.

2. Code Structure and Modularity

The code is organized into functions, promoting modularity and readability. Each function addresses a specific aspect, such as data reading, statistical analysis, visualization, and Llama-2 model interaction.

3. Extensibility and Scalability

The application is designed to be easily extensible for additional datasets and functionalities. Modular functions allow for scalability and adaptability to diverse datasets.

IV. User Interaction

1. Command-Line Interface (CLI)

The application features a command-line interface, providing users with an intuitive means of interacting with the system. Commands include 'help,' 'analyze,' 'plot,' and 'question.'

```
command = input('Enter a command (help, analyze, plot, question): ')

# Example usage of 'analyze'
if command == 'analyze':
    column = input('Enter the column name: ')
    print('Mean:', calculate_mean(data, column))
    print('Median:', calculate_median(data, column))
    # ... Additional statistical metrics
```

These functions define statistical analysis operations that can be performed on the data. They calculate the mean, median, mode, standard deviation, and correlation coefficient for a given column of data.

Python

```
# Define functions for generating plots
def generate_histogram(data, column):
    plt.hist(data[column])
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.title('Histogram of ' + column)
    plt.show()

def generate_scatter_plot(data, column1, column2):
    plt.scatter(data[column1], data[column2])
    plt.xlabel(column1)
    plt.ylabel(column2)
    plt.title('Scatter Plot of ' + column1 + ' vs. ' + column2)
    plt.show()

def generate_line_plot(data, column):
    plt.plot(data[column])
    plt.xlabel('Index')
    plt.ylabel(column)
    plt.title('Line Plot of ' + column)
    plt.show()
```

These functions define plot generation operations. They create histograms, scatter plots, and line plots for a given column or pair of columns of data.

```python
# Define functions for using Llama-2 model
def load_llama_2_model():
    model_name = 'facebook/bart-base'
    tokenizer = AutoTokenizer.from_pretrained(model_name)
    model = AutoModelForQuestionAnswering.from_pretrained(model_name)
    return tokenizer, model

def answer_question(tokenizer, model, question, text):
    encoded_text = tokenizer(text, return_tensors='pt')
    input_ids = encoded_text['input_ids']
    attention_mask = encoded_text['attention_mask']
    with torch.no_grad():
        outputs = model(input_ids, attention_mask=attention_mask)
    start_logits = outputs.start_logits
    end_logits = outputs.end_logits
    answer_start = torch.argmax(start_logits, dim=-1)
    answer_end = torch.argmax(end_logits, dim=-1)
    decoded_answer = tokenizer.decode(input_ids[0][answer_start:answer_end + 1])
    return decoded_answer
```

These functions define operations for using the Llama-2 model for question answering. They load the model, preprocess the input text, generate the answer logits, and decode the answer span to extract the final answer.

```python
# Main loop
while True:
    # User input
    command = input('Enter a command (help, analyze, plot, question): ')
```

This loop initiates the user interaction by prompting the user to enter a command. The available commands are 'help', 'analyze', 'plot', and 'question'.

```python
if command == 'help':
    print('Available commands:')
    print('help: Display this help message')
    print('analyze: Perform statistical analysis on a column')
    print('plot: Generate a plot of a column')
    print('question: Ask a question about the data')
```

2. User Guidance and Feedback

Clear guidance is provided to users on available commands, ensuring a positive and informed user experience. Feedback messages confirm the outcomes of user actions.

V. Future Enhancements

1. Graphical User Interface (GUI)

The addition of a Graphical User Interface (GUI) is a potential enhancement, offering a more user-friendly experience.

2. Additional Statistical Analyses

The application could be enriched with further statistical analyses, broadening its analytical capabilities.

3. Support for Different Datasets

Future iterations may extend the application to seamlessly handle diverse datasets from various domains.

4. Performance Optimization

Continued efforts will focus on optimizing the application's performance, especially with larger datasets.

VI. Conclusion

DataStatsViz stands as a powerful tool for users seeking insightful analysis and visualization of their datasets. The Statistical Analysis and Plotting Application stands as a valuable tool for data analysis and visualization. Its modular design, comprehensive functionalities, and user-friendly interface make it a powerful asset for extracting insights from CSV data. The application's ability to perform statistical analysis, generate informative plots, and answer user questions makes it a versatile tool for data exploration and understanding.