
Mini Project (1): Simple Chatbot

Name: Habibullah Farahmand

ID: DC24-M-03-102

Supervisor: Zainullah Ziarmal

Course Title: AI and Engineering

Submission Date: Oct-20-2024

Objective

The goal of this project is to create a basic chatbot capable of responding to predefined user inputs. This project demonstrates fundamental programming concepts like conditional logic, dictionaries for key-value mapping, and interaction through user inputs.

Skills and Concepts Highlighted

- **Python Basics:** Demonstrates the use of loops, conditionals, and functions.
- **Natural Language Processing (NLP):** Introduced basic conversational logic for text-based interactions.
- **User Interaction:** Enhanced understanding of handling user inputs and providing meaningful responses.

Project Workflow

1. **Predefined Questions and Responses:**
    - A dictionary is used to map questions to corresponding responses, showcasing Python's efficient data structures.
2. **Conversation Loop:**
    - A `while` loop ensures the program continuously interacts with the user until they choose to quit.
3. **Input Validation:**
    - The chatbot checks user inputs against predefined responses and handles unknown inputs gracefully by informing the user.

Real-World Application

This project serves as a foundation for building more advanced chatbots used in customer service, virtual assistants, or even mental health support tools. It represents the first step toward integrating Natural Language Processing (NLP) and Machine Learning (ML) for more complex conversational agents.

Features

1. **Interactive Interface:**
    - Users can choose from a predefined set of questions for seamless interaction.
2. **Dynamic Input Handling:**
    - The chatbot gracefully manages unrecognized inputs.
3. **Easy to Extend:**
    - Additional questions and responses can be added to expand functionality.

Results

The chatbot successfully demonstrates the following:

- Engaging in basic conversations.
- Responding accurately to predefined questions.
- Ending the interaction when the user chooses to quit.

Code Highlights

Key snippet of the implementation:

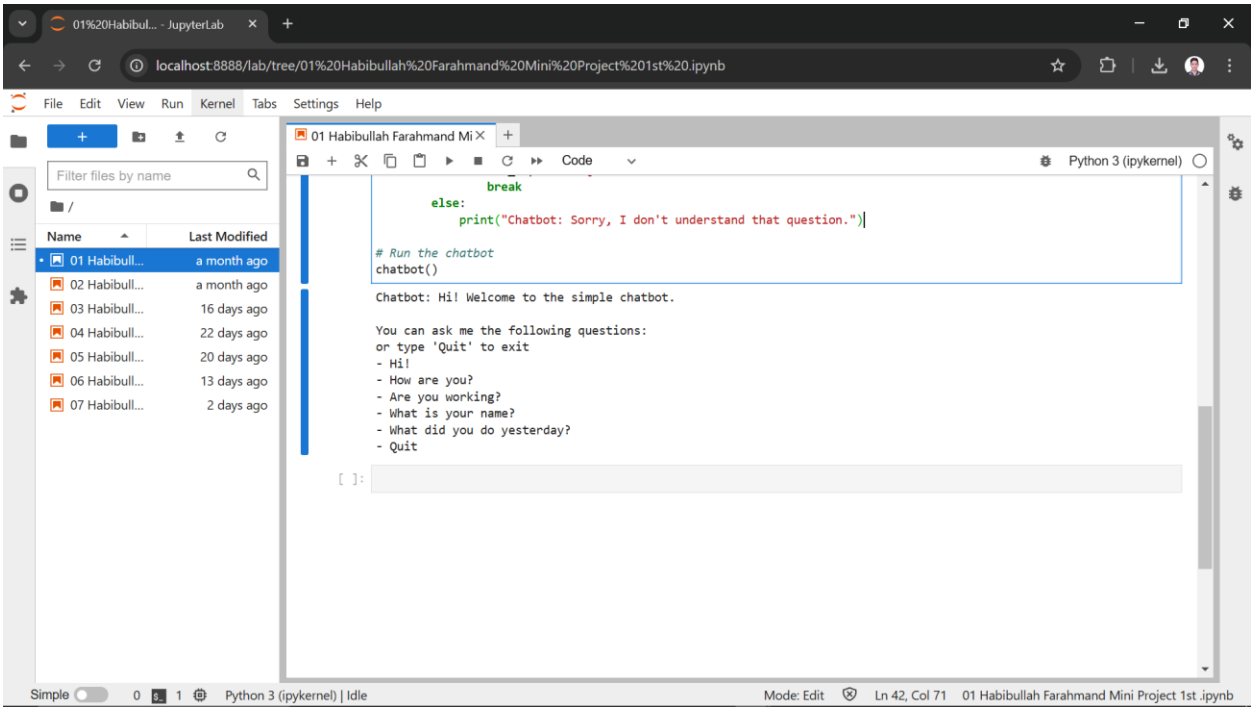
```
responses = {
    "Hi!": "Hello! How can I assist you today?",
    "How are you?": "I'm just a program, but I'm here to help!",
    "Are you working?": "Yes, I'm always ready to assist you!",
    "What is your name?": "I'm a simple chatbot created by you.",
    "What did you do yesterday?": "I was waiting for you to talk to me!",
    "Quit": "Goodbye! Have a great day!"
}
```

The logic checks if user input matches any predefined keys in the dictionary to provide an appropriate response.

Next Steps

1. **Scalability:**
   - Introduce a more flexible NLP model (e.g., `spaCy`, `NLTK`, or OpenAI APIs) to understand a variety of user inputs.
2. **Personalization:**
   - Add features like remembering user preferences or providing tailored responses.
3. **Deployment:**
   - Deploy the chatbot as a web or mobile application using frameworks like Flask or Django.

Output Screenshots



Mini Project (1.1): Descriptive Statistics and Data Visualization

Name: Habibullah Farahmand

ID: DC24-M-03-102

Supervisor: Zainullah Ziarmal

Course Title: AI and Engineering

Submission Date: Oct-20-2024

Objective:

To showcase proficiency in statistical analysis and data visualization techniques using Python. This project highlights the use of descriptive statistics and various visualization tools to analyze datasets, compute statistical measures, and create graphical representations of data.

Key Elements:

1. Skill Representation:

- **Statistical Analysis:**
  - Mean, median, mode, variance, standard deviation, skewness, percentiles, and range.
  - Central tendency and variability analysis using libraries like `numpy`, `pandas`, `scipy.stats`, and `statistics`.
- **Data Visualization:**
  - Box plots, histograms, pie charts, bar charts, scatter plots (X-Y plots), and heatmaps.

2. Real-World Application:

- Analyzed a hypothetical dataset representing COVID-19 deaths across different provinces in Afghanistan.
- Calculated statistical measures and provided visual insights into the data distribution and central tendency.

3. Creative Freedom:

- Focused on analyzing public health data to derive meaningful insights.

- Visualized data with charts to identify trends and patterns for better decision-making.

4. Variety:

- Utilized multiple libraries for redundancy and cross-validation.
- Incorporated both numerical and graphical methods to ensure robust analysis.

Project Details:

1. Dataset:

- **Descriptive Statistics Datasets:**
- `x = [5, 3, 6, 7, 2, 3, 7, 5, 9, 2, 3]`
- `x_with_nan = [8, 6, 7, 25, math.nan, math.nan, math.nan, 4, 2.0]`
- **COVID-19 Dataset:**
- `Covide_Died_case = np.array([`
- `    [4, 5, 2],`
- `    [8, 4, 2],`
- `    [3, 5, 2],`
- `    [8, 27, 4],`
- `    [32, 5, 2]`
- `])`
- `provinces = ['Kabul', 'Kundoz', 'Herat', 'Mazar', 'Bamyan']`
- `regions = ['Central', 'North', 'East']`
- `df = pd.DataFrame(Covide_Died_case, index=provinces, columns=regions)`

2. Objectives:

- Compute and compare statistical measures using different Python libraries.
- Visualize data trends and distributions using a variety of graphical methods.

3. Implementation:

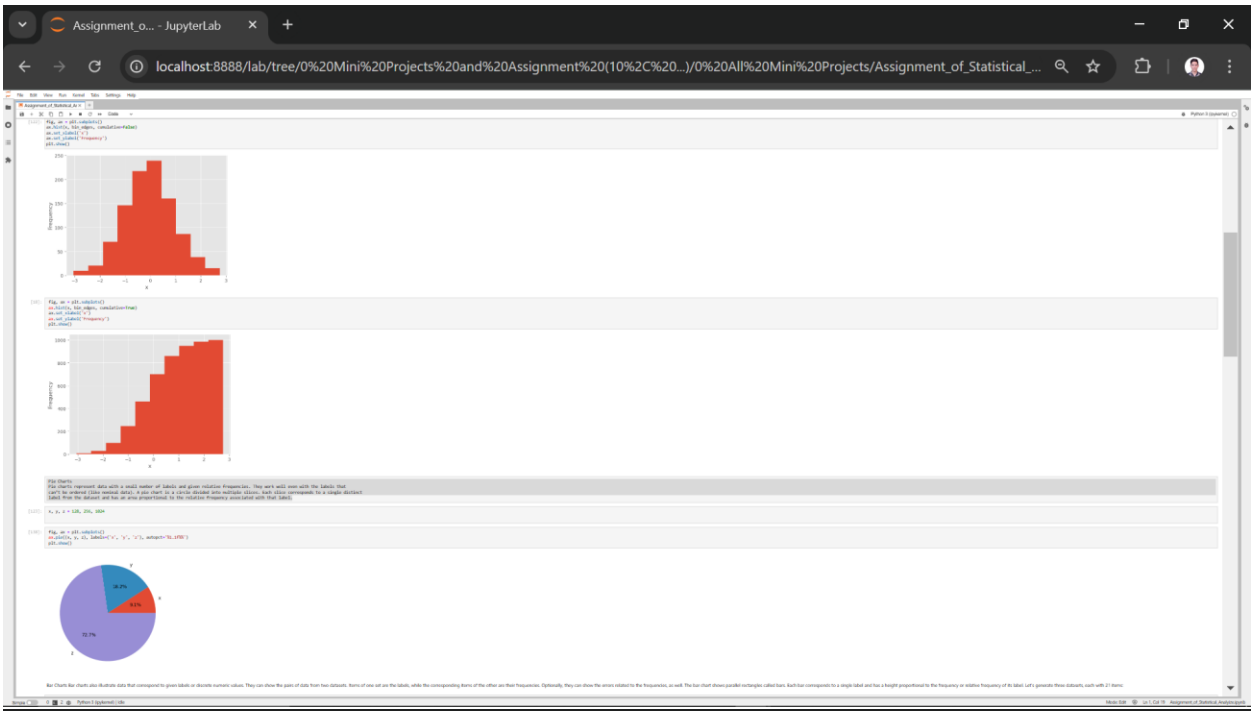
- **Statistical Analysis:**
  - Calculated measures like mean, weighted mean, geometric mean, harmonic mean, median, mode, variance, and skewness using libraries such as `numpy`, `pandas`, and `scipy.stats`.
  - Example:
  - `import numpy as np`
  - `mean_x = np.mean(x)`
  - `std_dev_x = np.std(x)`
- **Data Visualization:**
  - Created various visualizations to represent data insights:
    - **Box Plot:**
    - `import matplotlib.pyplot as plt`
    - `plt.boxplot(x)`
    - `plt.title('Box Plot of x')`

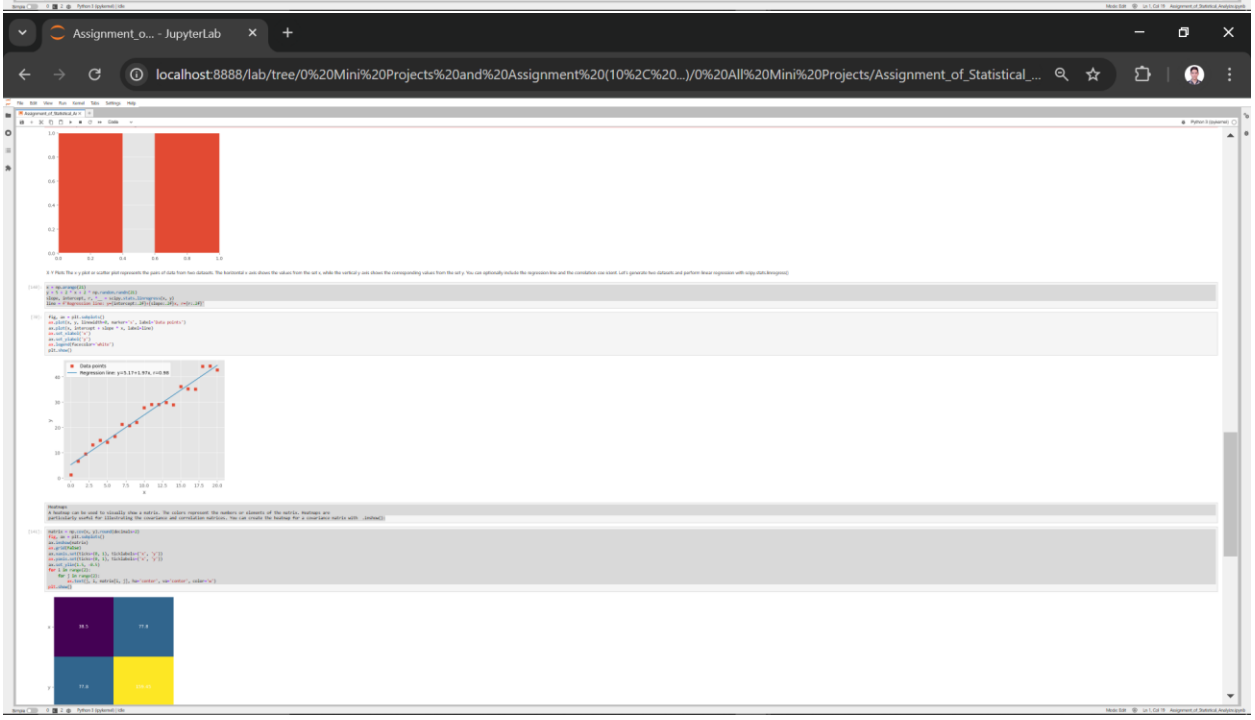
        - `plt.show()`
        - **Histogram:**
        - `plt.hist(x, bins=10)`
        - `plt.title('Histogram of x')`
        - `plt.xlabel('Value')`
        - `plt.ylabel('Frequency')`
        - `plt.show()`
        - **Pie Chart:**
        - `plt.pie([4, 5, 2], labels=regions, autopct='%1.1f%%')`
        - `plt.title('COVID-19 Deaths by Region')`
        - `plt.show()`
- **Advanced Visualizations:**
    - Generated heatmaps for the covariance matrix of the dataset.
    - `matrix = np.cov(Covide_Died_case, rowvar=False).round(decimals=2)`
    - `plt.imshow(matrix, cmap='hot', interpolation='nearest')`
    - `plt.title('Heatmap of Covariance Matrix')`
    - `plt.colorbar()`
    - `plt.show()`

Benefits and Outcomes:

- **Showcasing Talent:** Demonstrated the ability to apply statistical methods and visualization techniques to analyze real-world data.
- **Hands-On Learning:** Gained practical experience with libraries such as `numpy`, `pandas`, `matplotlib`, and `seaborn`.
- **Confidence Building:** Enhanced confidence in performing data analysis tasks and creating visual dashboards.

Output Screenshots:


Assignment_o... - JupyterLab
localhost:8888/lab/tree/0%20Mini%20Projects%20and%20Assignment%20(10%2C%20...)/0%20All%20Mini%20Projects/Assignment_of_Statistical_...


Assignment_o... - JupyterLab
localhost:8888/lab/tree/0%20Mini%20Projects%20and%20Assignment%20(10%2C%20...)/0%20All%20Mini%20Projects/Assignment_of_Statistical_...

Mini Project (2): Data Analysis on Titanic Dataset

Name: Habibullah Farahmand

ID: DC24-M-03-102

Supervisor: Zainullah Ziarmal

Course Title: AI and Engineering

Submission Date: Oct-20-2024

Objective

The goal of this project is to analyze the Titanic dataset to uncover insights about passenger survival rates based on various factors. This project demonstrates data preprocessing, feature engineering, and visualization techniques using Python.

Skills and Concepts Highlighted

- **Data Cleaning:** Handling missing values with various imputation methods.
- **Feature Engineering:** Creating meaningful new features to enhance the dataset.
- **Data Filtering:** Extracting subsets of data based on specific conditions.
- **Data Visualization:** Using libraries like `Matplotlib` and `Seaborn` for exploratory data analysis.

Project Workflow

1. **Loading and Inspecting the Dataset:**
    - Used the Titanic dataset (`train.csv`) to perform data analysis.
    - Initial inspection revealed missing values in columns like `Age`, `Cabin`, and `Embarked`.
2. **Handling Missing Values:**
    - `Age`: Filled missing values with the median age.
    - `Cabin`: Replaced missing values with "Unknown".
    - `Embarked`: Dropped rows with missing values.
3. **Feature Engineering:**
    - **Family Size:** Combined `SibSp` and `Parch` to create a new column.
    - **IsAlone:** Determined whether a passenger traveled alone.

    - **Title Extraction:** Extracted titles from the `Name` column to gain insights into passenger demographics.
4. **Data Filtering:**
    - Focused analysis on passengers aged over 18 and belonging to passenger classes 1 and 2.
5. **Visualization:**
    - Explored survival trends using various plots:
        - **Age Distribution:** Illustrated the age distribution among passengers.
        - **Survival by Gender:** Highlighted higher survival rates for females.
        - **Survival by Class:** Showed better survival rates for higher classes.
        - **Correlation Heatmap:** Identified relationships among numeric features.
        - **Family Size Impact:** Showed the relationship between family size and survival probability.

Real-World Application

This analysis provides insights that could be used in decision-making processes, such as:

- Designing safer transport systems by understanding passenger demographics.
- Enhancing machine learning models for survival prediction on Titanic-like datasets.

Results

1. **Key Findings:**
    - Women had significantly higher survival rates.
    - Passengers in 1st class had a better chance of survival compared to those in 2nd or 3rd class.
    - Family size influenced survival rates, with very large or very small family sizes showing lower survival chances.
2. **Visualizations:**
   (Include screenshots or saved images of the key plots.)

Code Highlights

Handling Missing Values:

```
df['Age'].fillna(df['Age'].median(), inplace=True)
df['Cabin'].fillna('Unknown', inplace=True)
df = df.dropna(subset=['Embarked'])
```

Feature Engineering:

```
df['FamilySize'] = df['SibSp'] + df['Parch'] + 1
df['IsAlone'] = (df['FamilySize'] == 1).astype(int)
df['Title'] = df['Name'].str.extract(' ([A-Za-z]+)\.', expand=False)
```

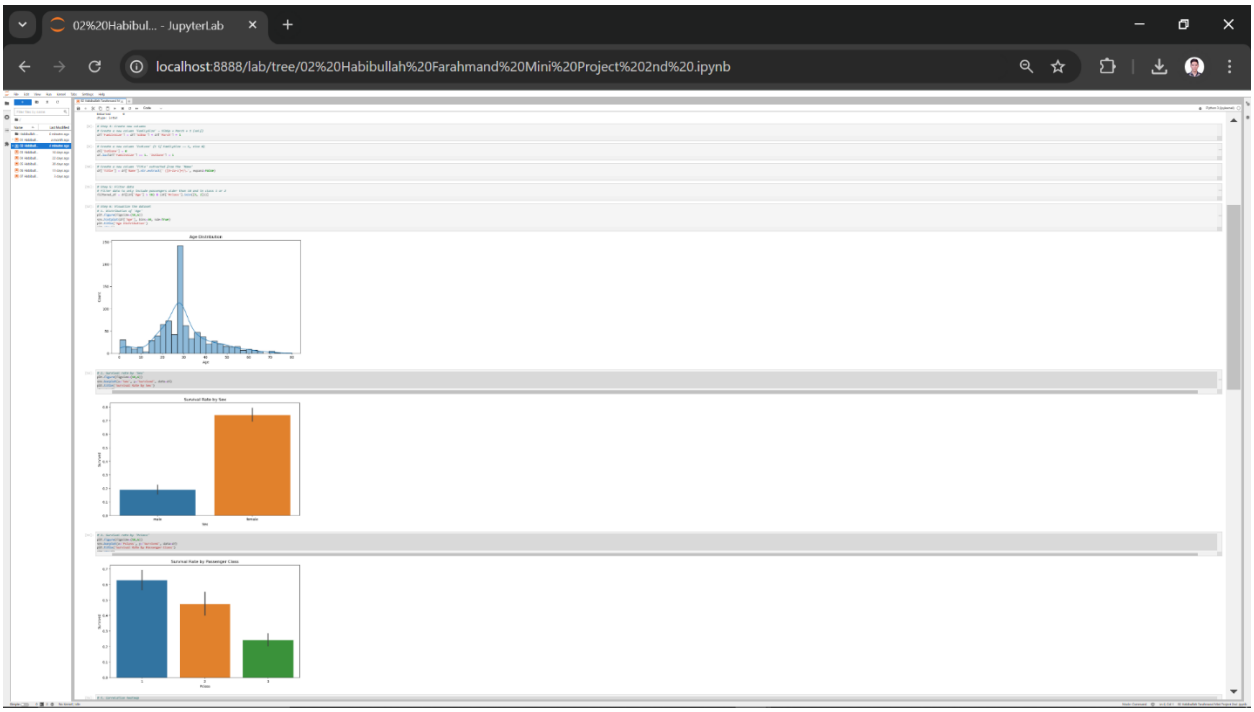
Visualizations:

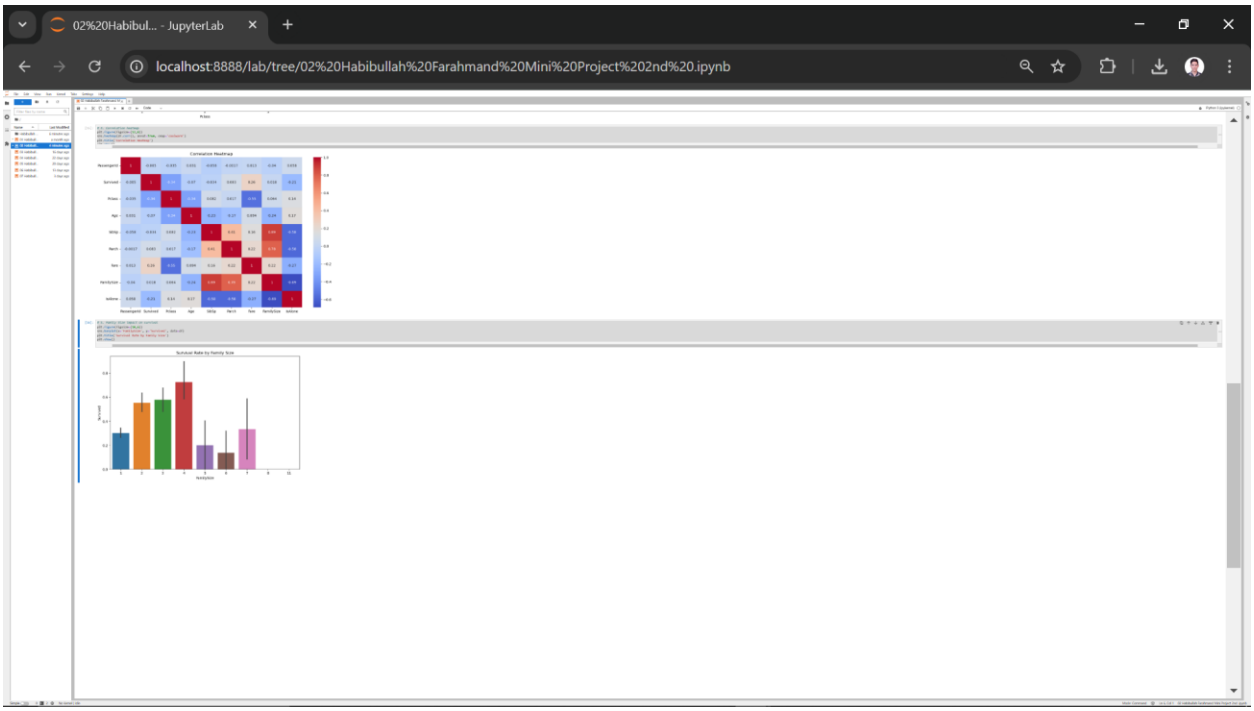
- **Survival Rate by Gender:**
- `sns.barplot(x='Sex', y='Survived', data=df)`
- **Correlation Heatmap:**
- `sns.heatmap(df.corr(), annot=True, cmap='coolwarm')`

Next Steps

1. **Predictive Modeling:**
    - Extend the analysis by applying machine learning models like logistic regression or random forests.
2. **Advanced Feature Engineering:**
    - Explore additional features like ticket price normalization or text analysis on `Name`.
3. **Web Deployment:**
    - Develop a web application to showcase the insights interactively.

Output Screenshots




02%20Habibul... - JupyterLab
localhost:8888/lab/tree/02%20Habibullah%20Farahmand%20Mini%20Project%202nd%20.ipynb
Correlation Heatmap
Survival Rate by Family Size

Mini Project (3): Cleaning and Manipulating Data

Name: Habibullah Farahmand

ID: DC24-M-03-102

Supervisor: Zainullah Ziarmal

Course Title: AI and Engineering

Submission Date: Oct-26-2024

Objective

This mini-project demonstrates the ability to clean, preprocess, and analyze real-world movie data using Python. The project focuses on handling missing values, filtering data based on specific criteria, and creating a structured workflow for reproducible data analysis. By the end, this project showcases skills in data wrangling, exploratory analysis, and presentation of results.

Key Elements of the Portfolio

1. Skill Representation

- **Data Cleaning:** Handling missing values in numerical and categorical columns.
- **Data Manipulation:** Filtering data, renaming columns, and setting indices.
- **Exploratory Data Analysis (EDA):** Analyzing IMDb scores and other key attributes of movies.
- **Pandas Proficiency:** Using advanced features of pandas for dataset exploration.

---

2. Real-World Application

- Filter and present only highly rated movies (IMDb score > 7.0) to demonstrate how data can be prepared for recommendation systems or reports for movie enthusiasts.
- Handle missing values effectively to ensure data consistency, addressing challenges faced in real-world datasets.

---

3. Creative Freedom

- Demonstrates the ability to clean and manipulate a dataset for personal or professional projects, such as creating an IMDb dashboard or a movie analysis blog.
- Enables further exploration for building recommendation systems or movie review sentiment analysis.

---

4. Variety

- **Techniques Covered:**
  - Handling missing data for both numerical and categorical columns.
  - Filtering and renaming data programmatically.
  - Preparing datasets for visualization or machine learning.
- **Future Extensions:** Integrate insights with a visualization library (e.g., Matplotlib, Seaborn) to enhance data presentation.

Benefits for Students

- **Hands-On Learning:** Involves direct engagement with real-world data, mimicking challenges faced in data science projects.
- **Showcasing Talent:** The processed dataset can be used as input for advanced models or visual dashboards.
- **Confidence Building:** Successfully completing the task boosts confidence in data preprocessing workflows.

Implementation Details

Guided Project Phases:

1. **Structured Assignment:** Clean the dataset and extract meaningful insights.
2. **Independent Work:** Enhance analysis by applying additional filters or preparing data for visualization.
3. **Milestones:** Define steps such as data loading, missing value handling, and dataset filtering.

Code Workflow

Step 1: Data Loading

```
import pandas as pd
```

```
# Load the dataset
df = pd.read_csv("movie.csv")
print(df.head())
```

Step 2: Understanding Dataset Dimensions

```
nrows, ncols = df.shape
print(f"Number of rows: {nrows}, Number of columns: {ncols}")
```

Step 3: Data Types Overview

```
print(df.dtypes)
```

Step 4: Handling Missing Values

```
for column in df.columns:
    if df[column].dtype in ['float64', 'int64']:  # Numerical columns
        df[column].fillna(df[column].median(), inplace=True)
    else:  # Categorical columns
        df[column].fillna(df[column].mode()[0], inplace=True)
```

Step 5: Filter Movies with IMDb Score > 7

```
filtered_df = df[df['imdb_score'] > 7.0]
```

Step 6: Standardizing Column Names

```
filtered_df.rename(columns=lambda col: col.upper(), inplace=True)
```

Step 7: Setting Index

```
filtered_df.set_index('MOVIE_TITLE', inplace=True)
```

Step 8: Extract Data for a Specific Movie

```
print(filtered_df.loc["Avatar"])
```

Step 9: Rename Columns

```
filtered_df.rename(columns={"MOVIE_IMDB_LINK": "IMDB_LINK"}, inplace=True)
```

Outputs

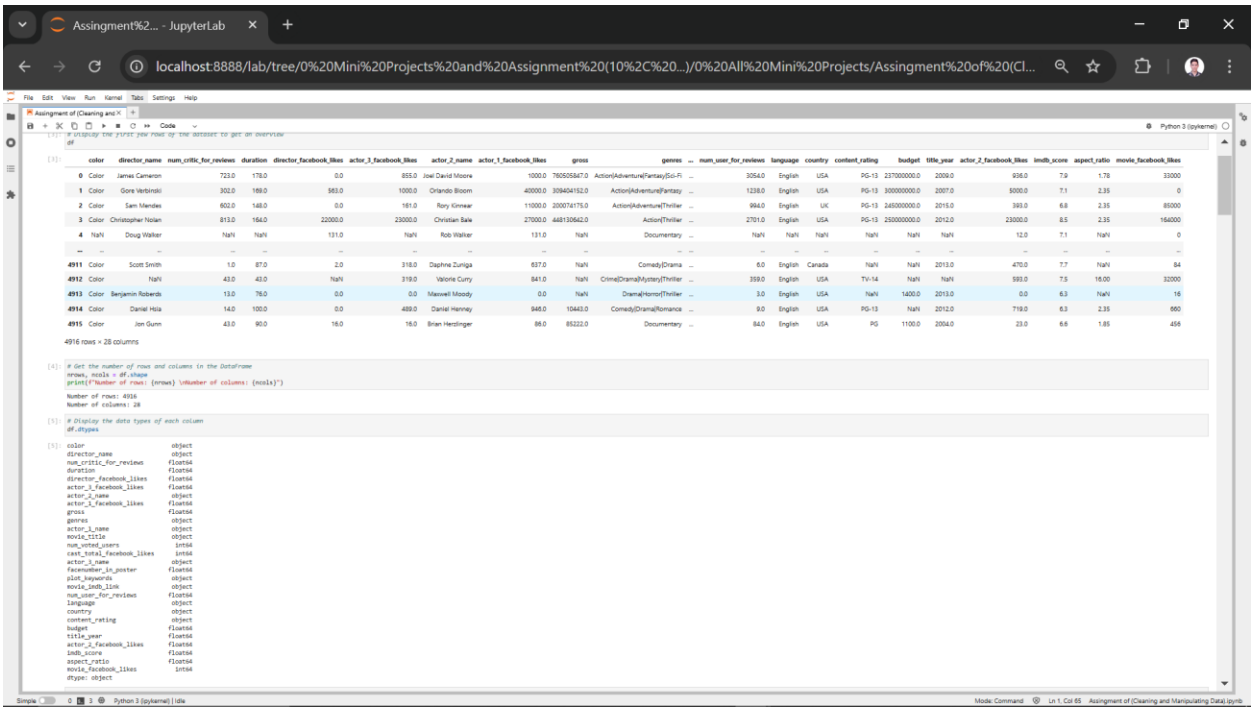
Dataset Summary

- **Initial Rows and Columns:** 4916 rows, 28 columns.
- **Processed Dataset:** Filtered movies with IMDb score > 7.0; rows reduced to 2110.

Sample Output for a Movie

```
COLOR                                Color
DIRECTOR_NAME                James Cameron
NUM_CRITIC_FOR_REVIEWS               723.0
DURATION                             178.0
DIRECTOR_FACEBOOK_LIKES                0.0
...
```

Output Screenshot:




```
# Display the first few rows of the dataset to get an overview
df
```

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross	genres	...	num_user_for_reviews	language	country	content_rating	budget	title_year	actor_2_facebook_likes	imdb_score	aspect_ratio	movie_facebook_likes
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	...	3054.0	English	USA	PG-13	237000000.0	2009.0	936.0	7.9	1.78	33000
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0	Action|Adventure|Fantasy	...	1238.0	English	USA	PG-13	300000000.0	2007.0	5000.0	7.1	2.35	0
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200074175.0	Action|Adventure|Thriller	...	994.0	English	UK	PG-13	245000000.0	2015.0	393.0	6.8	2.35	85000
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.0	Action|Thriller	...	2701.0	English	USA	PG-13	250000000.0	2012.0	23000.0	8.5	2.35	164000
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	131.0	NaN	Documentary	...	NaN	NaN	NaN	NaN	NaN	NaN	12.0	7.1	NaN	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4911	Color	Scott Smith	1.0	87.0	2.0	318.0	Daphne Zuniga	637.0	NaN	Comedy|Drama	...	6.0	English	Canada	NaN	NaN	2013.0	470.0	7.7	NaN	84
4912	Color	NaN	43.0	43.0	NaN	319.0	Valorie Curry	841.0	NaN	Crime|Drama|Mystery|Thriller	...	359.0	English	USA	TV-14	NaN	NaN	593.0	7.5	16.00	32000
4913	Color	Benjamin Roberds	13.0	76.0	0.0	0.0	Maxwell Moody	0.0	NaN	Drama|Horror|Thriller	...	3.0	English	USA	NaN	1400.0	2013.0	0.0	6.3	NaN	16
4914	Color	Daniel Hsia	14.0	100.0	0.0	489.0	Daniel Henney	946.0	10443.0	Comedy|Drama|Romance	...	9.0	English	USA	PG-13	NaN	2012.0	719.0	6.3	2.35	660
4915	Color	Jon Gunn	43.0	90.0	16.0	16.0	Brian Herzlinger	86.0	85222.0	Documentary	...	84.0	English	USA	PG	1100.0	2004.0	23.0	6.6	1.85	456

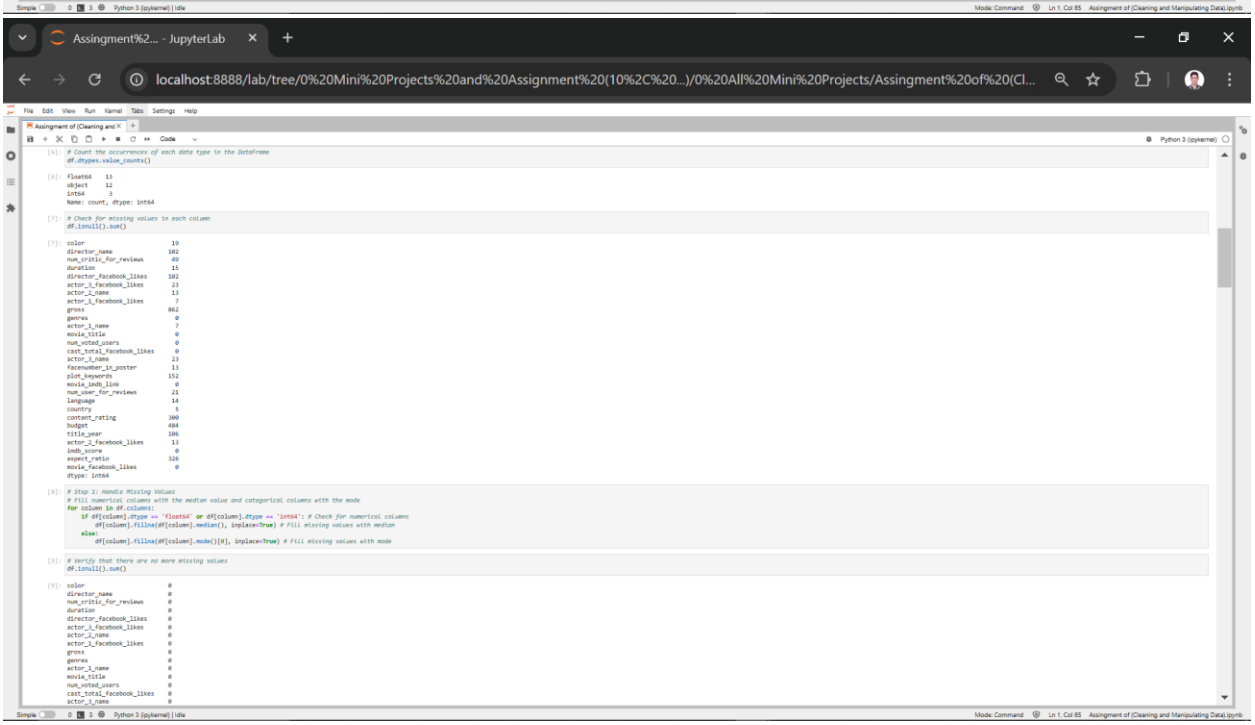
4916 rows × 28 columns

```
# Get the number of rows and columns in the DataFrame
nrows, ncols = df.shape
print(f"Number of rows: {nrows} \nNumber of columns: {ncols}")
```

```
Number of rows: 4916
Number of columns: 28
```

```
# Display the data types of each column
df.dtypes
```

```
color                      object
director_name              object
num_critic_for_reviews    float64
duration                  float64
director_facebook_likes   float64
actor_3_facebook_likes    float64
actor_2_name               object
actor_1_facebook_likes    float64
gross                     float64
genres                     object
actor_1_name               object
movie_title                object
num_voted_users             int64
cast_total_facebook_likes   int64
actor_3_name               object
facenumber_in_poster      float64
plot_keywords              object
movie_imdb_link            object
num_user_for_reviews      float64
language                   object
country                    object
content_rating             object
budget                    float64
title_year                float64
actor_2_facebook_likes    float64
imdb_score                float64
aspect_ratio              float64
movie_facebook_likes        int64
dtype: object
```




```
# Count the occurrences of each data type in the DataFrame
df.dtypes.value_counts()
```

```
float64    13
object     12
int64       3
Name: count, dtype: int64
```

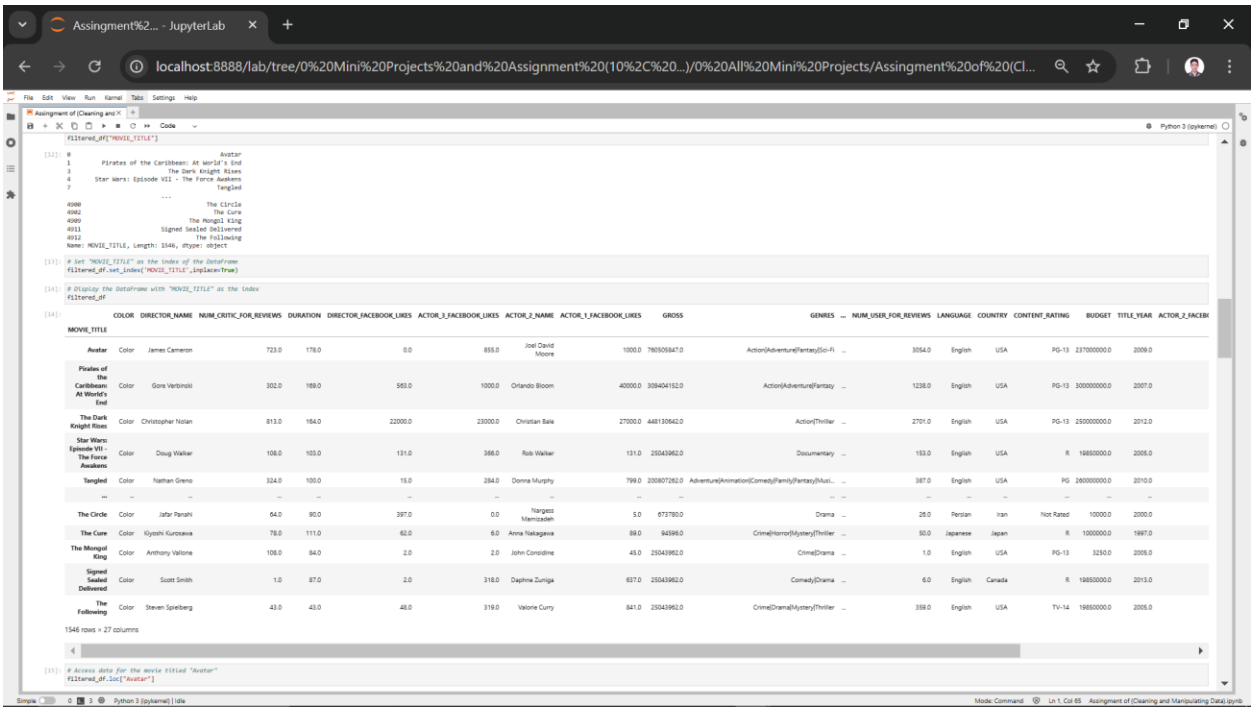
```
# Check for missing values in each column
df.isnull().sum()
```

```
color                      19
director_name             102
num_critic_for_reviews     49
duration                   15
director_facebook_likes   102
actor_3_facebook_likes     23
actor_2_name               13
actor_1_facebook_likes      7
gross                     862
genres                      0
actor_1_name                7
movie_title                 0
num_voted_users             0
cast_total_facebook_likes   0
actor_3_name               23
facenumber_in_poster       13
plot_keywords             152
movie_imdb_link             0
num_user_for_reviews       21
language                   14
country                     5
content_rating            300
budget                    484
title_year                106
actor_2_facebook_likes     13
imdb_score                  0
aspect_ratio              326
movie_facebook_likes        0
dtype: int64
```

```
# Step 1: Handle Missing Values
# Fill numerical columns with the median value and categorical columns with the mode
for column in df.columns:
    if df[column].dtype == 'float64' or df[column].dtype == 'int64': # Check for numerical columns
        df[column].fillna(df[column].median(), inplace=True) # Fill missing values with median
    else:
        df[column].fillna(df[column].mode()[0], inplace=True) # Fill missing values with mode
```

```
# Verify that there are no more missing values
df.isnull().sum()
```

```
color                     0
director_name             0
num_critic_for_reviews    0
duration                  0
director_facebook_likes   0
actor_3_facebook_likes    0
actor_2_name              0
actor_1_facebook_likes    0
gross                     0
genres                    0
actor_1_name              0
movie_title               0
num_voted_users           0
cast_total_facebook_likes 0
actor_3_name              0
```




```
filtered_df["MOVIE_TITLE"]
```

```
0                                    Avatar
1          Pirates of the Caribbean: At World's End
3                           The Dark Knight Rises
4          Star Wars: Episode VII - The Force Awakens
7                                      Tangled
                           ...
4908                            The Circle
4982                               The Cure
4989                          The Mongol King
4911                  Signed Sealed Delivered
4912                            The Following
Name: MOVIE_TITLE, Length: 1546, dtype: object
```

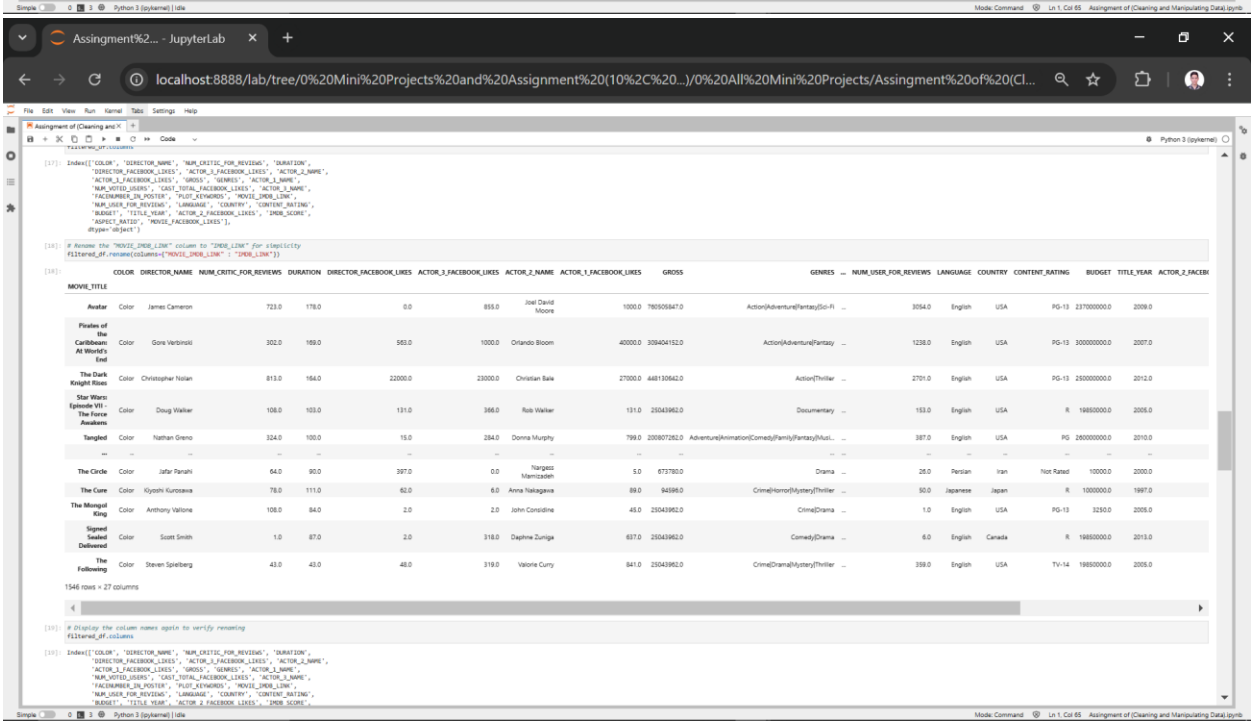
```
# Set "MOVIE_TITLE" as the index of the DataFrame
filtered_df.set_index('MOVIE_TITLE',inplace=True)
```

```
# Display the DataFrame with "MOVIE_TITLE" as the index
filtered_df
```

MOVIE_TITLE	COLOR	DIRECTOR_NAME	NUM_CRITIC_FOR_REVIEWS	DURATION	DIRECTOR_FACEBOOK_LIKES	ACTOR_3_FACEBOOK_LIKES	ACTOR_2_NAME	ACTOR_1_FACEBOOK_LIKES	GROSS	GENRES	...	NUM_USER_FOR_REVIEWS	LANGUAGE	COUNTRY	CONTENT_RATING	BUDGET	TITLE_YEAR	ACTOR_2_FACEBOOK_LIKES
Avatar	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	...	3054.0	English	USA	PG-13	237000000.0	2009.0	
Pirates of the Caribbean: At World's End	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0	Action|Adventure|Fantasy	...	1238.0	English	USA	PG-13	300000000.0	2007.0	
The Dark Knight Rises	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.0	Action|Thriller	...	2701.0	English	USA	PG-13	250000000.0	2012.0	
Star Wars: Episode VII - The Force Awakens	Color	Doug Walker	108.0	103.0	131.0	366.0	Rob Walker	131.0	25043962.0	Documentary	...	153.0	English	USA	R	198850000.0	2005.0	
Tangled	Color	Nathan Greno	324.0	100.0	15.0	284.0	Donna Murphy	799.0	200807262.0	Adventure|Animation|Comedy|Family|Fantasy|Musi...	...	387.0	English	USA	PG	260000000.0	2010.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
The Circle	Color	Jafar Panahi	64.0	90.0	397.0	0.0	Nargess Mamizadeh	5.0	673780.0	Drama	...	26.0	Persian	Iran	Not Rated	10000.0	2000.0	
The Cure	Color	Kiyoshi Kurosawa	78.0	111.0	62.0	6.0	Anna Nakagawa	89.0	94596.0	Crime|Horror|Mystery|Thriller	...	50.0	Japanese	Japan	R	1000000.0	1997.0	
The Mongol King	Color	Anthony Vallone	108.0	84.0	2.0	2.0	John Considine	45.0	25043962.0	Crime|Drama	...	1.0	English	USA	PG-13	3250.0	2005.0	
Signed Sealed Delivered	Color	Scott Smith	1.0	87.0	2.0	318.0	Daphne Zuniga	637.0	25043962.0	Comedy|Drama	...	6.0	English	Canada	R	198500000.0	2013.0	
The Following	Color	Steven Spielberg	43.0	43.0	48.0	319.0	Valorie Curry	841.0	25043962.0	Crime|Drama|Mystery|Thriller	...	359.0	English	USA	TV-14	198500000.0	2005.0	

1546 rows × 27 columns

```
# Access data for the movie titled "Avatar"
filtered_df.loc["Avatar"]
```




```
Index(['COLOR', 'DIRECTOR_NAME', 'NUM_CRITIC_FOR_REVIEWS', 'DURATION',
       'DIRECTOR_FACEBOOK_LIKES', 'ACTOR_3_FACEBOOK_LIKES', 'ACTOR_2_NAME',
       'ACTOR_1_FACEBOOK_LIKES', 'GROSS', 'GENRES', 'ACTOR_1_NAME',
       'NUM_VOTED_USERS', 'CAST_TOTAL_FACEBOOK_LIKES', 'ACTOR_3_NAME',
       'FACENUMBER_IN_POSTER', 'PLOT_KEYWORDS', 'MOVIE_IMDB_LINK',
       'NUM_USER_FOR_REVIEWS', 'LANGUAGE', 'COUNTRY', 'CONTENT_RATING',
       'BUDGET', 'TITLE_YEAR', 'ACTOR_2_FACEBOOK_LIKES', 'IMDB_SCORE',
       'ASPECT_RATIO', 'MOVIE_FACEBOOK_LIKES'],
      dtype='object')
```

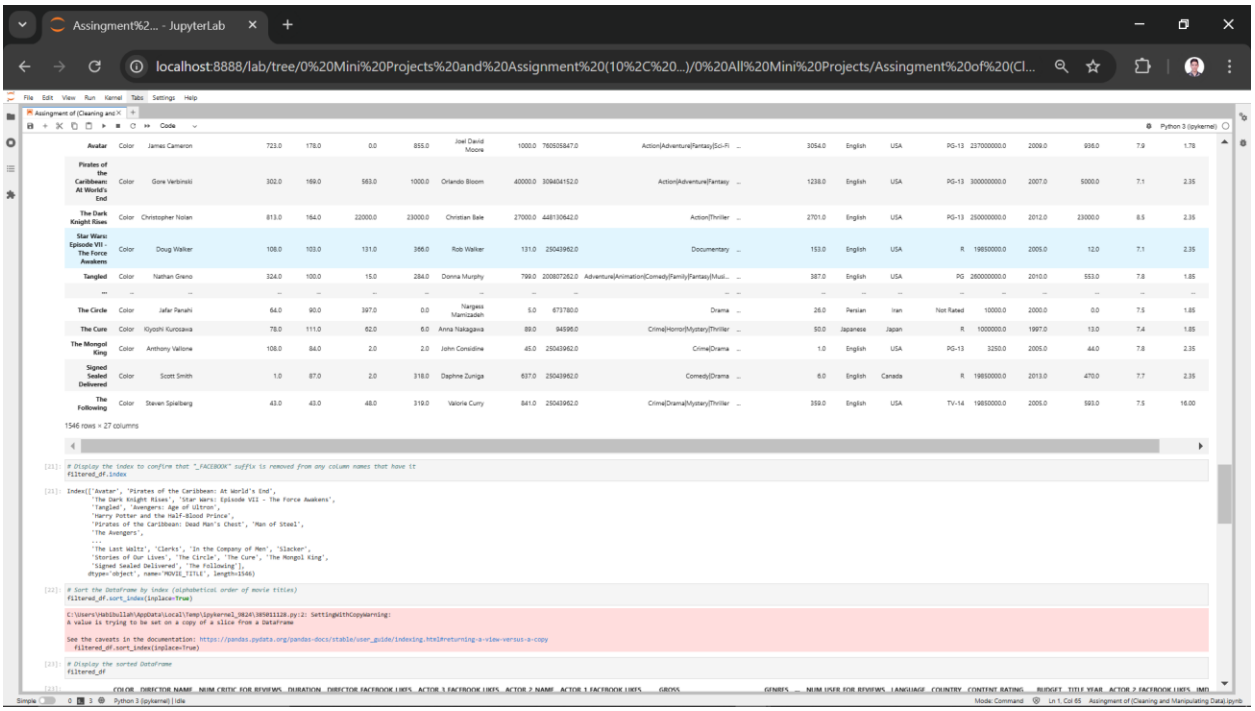
```
# Rename the "MOVIE_IMDB_LINK" column to "IMDB_LINK" for simplicity
filtered_df.rename(columns={"MOVIE_IMDB_LINK" : "IMDB_LINK"})
```

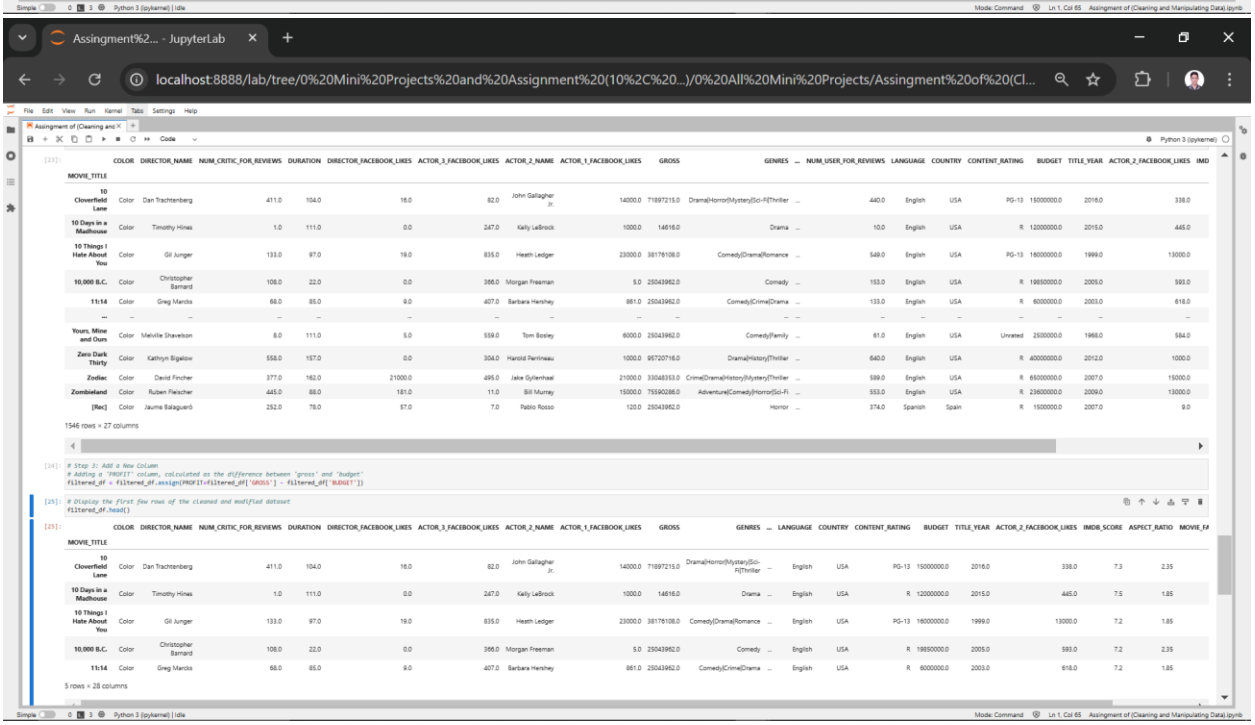
MOVIE_TITLE	COLOR	DIRECTOR_NAME	NUM_CRITIC_FOR_REVIEWS	DURATION	DIRECTOR_FACEBOOK_LIKES	ACTOR_3_FACEBOOK_LIKES	ACTOR_2_NAME	ACTOR_1_FACEBOOK_LIKES	GROSS	GENRES	...	NUM_USER_FOR_REVIEWS	LANGUAGE	COUNTRY	CONTENT_RATING	BUDGET	TITLE_YEAR	ACTOR_2_FACEBOOK_LIKES
Avatar	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	...	3054.0	English	USA	PG-13	237000000.0	2009.0	
Pirates of the Caribbean: At World's End	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0	Action|Adventure|Fantasy	...	1238.0	English	USA	PG-13	300000000.0	2007.0	
The Dark Knight Rises	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.0	Action|Thriller	...	2701.0	English	USA	PG-13	250000000.0	2012.0	
Star Wars: Episode VII - The Force Awakens	Color	Doug Walker	108.0	103.0	131.0	366.0	Rob Walker	131.0	25043962.0	Documentary	...	153.0	English	USA	R	198850000.0	2005.0	
Tangled	Color	Nathan Greno	324.0	100.0	15.0	284.0	Donna Murphy	799.0	200807262.0	Adventure|Animation|Comedy|Family|Fantasy|Musi...	...	387.0	English	USA	PG	260000000.0	2010.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
The Circle	Color	Jafar Panahi	64.0	90.0	397.0	0.0	Nargess Mamizadeh	5.0	673780.0	Drama	...	26.0	Persian	Iran	Not Rated	10000.0	2000.0	
The Cure	Color	Kiyoshi Kurosawa	78.0	111.0	62.0	6.0	Anna Nakagawa	89.0	94596.0	Crime|Horror|Mystery|Thriller	...	50.0	Japanese	Japan	R	1000000.0	1997.0	
The Mongol King	Color	Anthony Vallone	108.0	84.0	2.0	2.0	John Considine	45.0	25043962.0	Crime|Drama	...	1.0	English	USA	PG-13	3250.0	2005.0	
Signed Sealed Delivered	Color	Scott Smith	1.0	87.0	2.0	318.0	Daphne Zuniga	637.0	25043962.0	Comedy|Drama	...	6.0	English	Canada	R	198500000.0	2013.0	
The Following	Color	Steven Spielberg	43.0	43.0	48.0	319.0	Valorie Curry	841.0	25043962.0	Crime|Drama|Mystery|Thriller	...	359.0	English	USA	TV-14	198500000.0	2005.0	

1546 rows × 27 columns

```
# Display the column names again to verify renaming
filtered_df.columns
```

```
Index(['COLOR', 'DIRECTOR_NAME', 'NUM_CRITIC_FOR_REVIEWS', 'DURATION',
       'DIRECTOR_FACEBOOK_LIKES', 'ACTOR_3_FACEBOOK_LIKES', 'ACTOR_2_NAME',
       'ACTOR_1_FACEBOOK_LIKES', 'GROSS', 'GENRES', 'ACTOR_1_NAME',
       'NUM_VOTED_USERS', 'CAST_TOTAL_FACEBOOK_LIKES', 'ACTOR_3_NAME',
       'FACENUMBER_IN_POSTER', 'PLOT_KEYWORDS', 'MOVIE_IMDB_LINK',
       'NUM_USER_FOR_REVIEWS', 'LANGUAGE', 'COUNTRY', 'CONTENT_RATING',
       'BUDGET', 'TITLE_YEAR', 'ACTOR_2_FACEBOOK_LIKES', 'IMDB_SCORE',
```


Assingment%2... - JupyterLab
localhost:8888/lab/tree/0%20Mini%20Projects%20and%20Assignment%20(10%2C%20...)/0%20All%20Mini%20Projects/Assingment%20of%20(Cl...
1546 rows × 27 columns
Display the index to confirm that "_FACEBOOK" suffix is removed from any column names that have it
filtered_df.index
Index(['Avatar', 'Pirates of the Caribbean: At World's End',
'The Dark Knight Rises', 'Star Wars: Episode VII - The Force Awakens',
'Tangled', 'Avengers: Age of Ultron',
'Harry Potter and the Half-Blood Prince',
'Pirates of the Caribbean: Dead Man's Chest', 'Man of Steel',
'The Avengers',
...
'The Last Waltz', 'Clerks', 'In the Company of Men', 'Slacker',
'Stories of Our Lives', 'The Circle', 'The Cure', 'The Mongol King',
'Signed Sealed Delivered', 'The Following'],
dtype='object', name='MOVIE_TITLE', length=1546)
# Sort the DataFrame by index (alphabetical order of movie titles)
filtered_df.sort_index(inplace=True)
C:\Users\Habibullah\AppData\Local\Temp\ipykernel_9824\385811128.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
filtered_df.sort_index(inplace=True)
# Display the sorted DataFrame
filtered_df


Assingment%2... - JupyterLab
localhost:8888/lab/tree/0%20Mini%20Projects%20and%20Assignment%20(10%2C%20...)/0%20All%20Mini%20Projects/Assingment%20of%20(Cl...
1546 rows × 27 columns
# Step 3: Add a New Column
# Adding a "PROFIT" column, calculated as the difference between 'gross' and 'budget'
filtered_df = filtered_df.assign(PROFIT=filtered_df['GROSS'] - filtered_df['BUDGET'])
# Display the first few rows of the cleaned and modified dataset
filtered_df.head()
5 rows × 28 columns

Mini Project (3): Real-Time Stock Price Dashboard

Name: Habibullah Farahmand

ID: DC24-M-03-102

Supervisor: Zainullah Ziarmal

Course Title: AI and Engineering

Submission Date: Oct-26-2024

Objective

The purpose of this project is to develop a real-time stock price visualization tool using the Dash framework. The dashboard retrieves data via the Alpha Vantage API and displays dynamic candlestick charts to track stock price movements.

Skills and Concepts Highlighted

- **Data Retrieval:** Using APIs (Alpha Vantage) for real-time data.
- **Data Processing:** Cleaning and formatting JSON data into a structured `pandas` DataFrame.
- **Interactive Dashboards:** Developing dynamic, user-interactive visualizations with Dash and Plotly.
- **Web Application Development:** Creating a responsive and user-friendly interface.

Project Workflow

1. **Data Collection:**
   - Integrated the Alpha Vantage API to fetch stock market data for a specified symbol.
   - Implemented error handling to manage API limits or data unavailability.
2. **Data Transformation:**
   - Processed JSON responses into a `pandas` DataFrame.
   - Converted timestamps to datetime and formatted columns (`Open`, `High`, `Low`, `Close`, `Volume`) for visualization.
3. **Dashboard Design:**

- Built a web-based dashboard using Dash with the following features:
  - **Ticker Input:** Allows users to specify a stock symbol (e.g., `AAPL` for Apple).
  - **Real-Time Updates:** Refreshes data and visualizations every 60 seconds using `dcc.Interval`.
  - **Candlestick Chart:** Visualizes stock price trends with `Plotly`'s `Candlestick` chart.

4. **Dynamic Interactivity:**
   - Callback functions in Dash dynamically update the chart based on user input or data refreshes.

Real-World Application

This dashboard can be used for:

- Real-time tracking of stock prices by individual investors or traders.
- Educational purposes to visualize and analyze financial data trends.
- Enhancing financial services with user-specific dashboards for portfolio tracking.

Features

1. **Real-Time Data Fetching:**
   - Retrieves and updates stock data every minute.
2. **Interactive Visualizations:**
   - Candlestick chart enables easy identification of price trends.
3. **User-Friendly Interface:**
   - Minimalist design with dynamic updates for smooth user interaction.

Code Highlights

Data Retrieval via Alpha Vantage API:

```
def fetch_stock_data(symbol="AAPL", api_key="BCYSQOOG6GQLA1Y4"):
    url =
f"https://www.alphavantage.co/query?function=TIME_SERIES_INTRADAY&symbol={sym
bol}&interval=1min&apikey={api_key}"
    response = requests.get(url)
    data = response.json()
    if "Time Series (1min)" in data:
        time_series = data["Time Series (1min)"]
        df = pd.DataFrame(time_series).T
        df.index = pd.to_datetime(df.index)
        df = df.astype(float)
        df.columns = ["Open", "High", "Low", "Close", "Volume"]
```

```
        return df.sort_index()
    else:
        print("Error: Could not retrieve data from Alpha Vantage.")
        return None
```

Dynamic Graph Updates:

```
@app.callback(
    Output("stock-graph", "figure"),
    [Input("ticker-input", "value"), Input("interval", "n_intervals")]
)
def update_graph(ticker, n_intervals):
    data = fetch_stock_data(ticker)
    if data is None:
        return go.Figure()
    fig = go.Figure(data=[go.Candlestick(
        x=data.index,
        open=data["Open"],
        high=data["High"],
        low=data["Low"],
        close=data["Close"]
    )])
    fig.update_layout(title=f"{ticker} Stock Price", xaxis_title="Time",
yaxis_title="Price")
    return fig
```

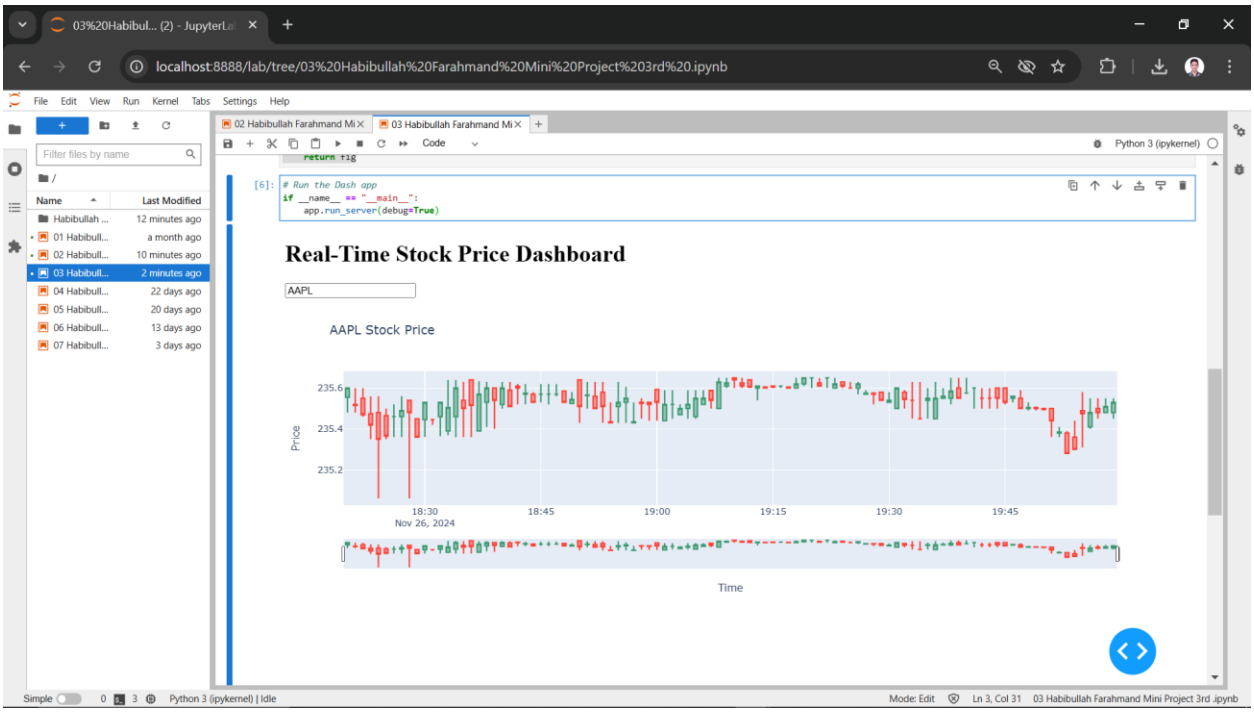
Results

- Successfully visualized real-time stock price data for user-defined stock symbols.
- Created a robust dashboard with dynamic updates, enabling continuous tracking of financial data.

Next Steps

1. **Enhance Data Sources:**
    - Integrate additional APIs (e.g., Yahoo Finance, IEX Cloud) for more comprehensive data.
2. **Add Features:**
    - Include historical data trends, key metrics (e.g., volume, market cap), and alerts for significant price changes.
3. **Deploy the Application:**
    - Host the dashboard using platforms like Heroku or AWS for broader accessibility.

Output Screenshots



Hand Written Assignment - 1:

Mini-Project: Classification of Paper Tissue Quality using KNN

Name: Habibullah Farahmand

ID: DC24-M-03-102

Supervisor: Zainullah Ziarmal

Course Title: AI and Engineering

Submission Date: Oct-27-2024

Objective

To classify whether a new paper tissue is "Good" or "Bad" based on its acid durability and strength using the K-Nearest Neighbors (KNN) algorithm.

Dataset

Acid Durability (X_1)	Strength (X_2)	Classification (Y)
7	7	Bad
7	4	Bad
3	4	Good
4	4	Good

Problem Statement

Classify the tissue with X1=3X_1 = 3 and X2=7X_2 = 7 using K=3K = 3.

Implementation

- Calculate the Euclidean distance between the new tissue and each data point in the dataset.
- Sort the distances to find the 3 nearest neighbors.
- Perform a majority vote among the neighbors' classifications.

Output

Predicted Classification: *Good*

Mini-Project: Color Classification using KNN

Name: Habibullah Farahmand

ID: DC24-M-03-102

Supervisor: Zainullah Ziarmal

Course Title: AI and Engineering

Submission Date: Oct-27-2024

Objective

To classify a color based on its brightness and saturation using K-Nearest Neighbors (KNN).

Dataset

Brightness	Saturation	Class
40	90	Red
60	25	Blue
60	10	Blue
60	60	Red
20	35	Red

Problem Statement

Classify the color with Brightness=20\text{Brightness} = 20 and Saturation=35\text{Saturation} = 35 using K=4K = 4.

Implementation

- Calculate Euclidean distances between the given point and all points in the dataset.
- Sort and select the 4 nearest neighbors.
- Apply majority voting to classify the new point.

Output

Predicted Classification: *Red*

Mini-Project: Sports Classification using Logical Rules

Name: Habibullah Farahmand

ID: DC24-M-03-102

Supervisor: Zainullah Ziarmal

Course Title: AI and Engineering

Submission Date: Oct-27-2024

Objective

To predict the sport class for a new individual, Omar, based on given rules and dataset.

Dataset

Name	Age	Gender (1 = Male, 0 = Female)	Class of Sports
Ali	35	1	Cricket
Ameena	20	0	None
Sara	40	0	Cricket, Football
Zafir	16	1	Cricket
Anjela	34	0	None

Problem Statement

Predict the sport class for Omar (Age=17,Gender=1\text{Age} = 17, \text{Gender} = 1).

Implementation

- Analyze patterns based on age and gender.
- Use logical rules and trends from the data to infer Omar's sports class.

Output

Predicted Sports Class: *Cricket*

Mini-Project: Sugar Content Prediction using Linear Regression

Name: Habibullah Farahmand

ID: DC24-M-03-102

Supervisor: Zainullah Ziarmal

Course Title: AI and Engineering

Submission Date: Oct-27-2024

Objective

To establish a linear relationship between the sugar content of fruit and the number of days since picking, and predict the sugar level at Day 10.

Dataset

Days	Sugar Content
0	7.9
1	9.5
3	11.3
4	11.8
6	12.0
8	11.3

Problem Statement

Find the sugar content at Day 10 using the formula:
$y=a+bxy = a + bx$
Where:
$b=n\sum(x\cdot y)-\sum x\sum yn\sum x2-(\sum x)2b = \frac{n \sum (x \cdot y) - \sum x \sum y}{n \sum x^2 - (\sum x)^2}$
$a=\sum y-b\sum xna = \frac{\sum y - b \sum x}{n}$

Implementation

1. Calculate aa and bb using the dataset.
2. Use the linear equation to predict the sugar content for Day 10.

Output

Predicted Sugar Level at Day 10: *Approx. 12.5*

Hand Written Assignment - 2:

Mini-Project: Fraud Detection Model Evaluation

Name: Habibullah Farahmand

ID: DC24-M-03-102

Supervisor: Zainullah Ziarmal

Course Title: AI and Engineering

Submission Date: Nov-04-2024

Objective

To evaluate the performance of a machine learning model that classifies transactions as fraudulent or non-fraudulent using the confusion matrix and key performance metrics.

Scenario

The model was tested on a dataset of 1,000 transactions. The outcomes are as follows:

- **True Positives (TP):** 300 (Fraudulent correctly predicted as fraudulent)
- **False Negatives (FN):** 50 (Fraudulent incorrectly classified as non-fraudulent)
- **True Negatives (TN):** 600 (Non-fraudulent correctly predicted as non-fraudulent)
- **False Positives (FP):** 50 (Non-fraudulent incorrectly classified as fraudulent)

---

1. Confusion Matrix

	Predicted Fraudulent	**Predicted Non-Fraudulent**
Actual Fraudulent	300 (TP)	50 (FN)
Actual Non-Fraudulent	50 (FP)	600 (TN)

---

2. Performance Metrics

1. **Accuracy**

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
$$\text{Accuracy} = \frac{300 + 600}{300 + 600 + 50 + 50} = \frac{900}{1000} = 0.90 \, (90\%)$$

2. **Precision**

$$\text{Precision} = \frac{TP}{TP + FP}$$
$$\text{Precision} = \frac{300}{300 + 50} = \frac{300}{350} = 0.857 \, (85.7\%)$$

3. **Recall (Sensitivity)**

$$\text{Recall} = \frac{TP}{TP + FN}$$
$$\text{Recall} = \frac{300}{300 + 50} = \frac{300}{350} = 0.857 \, (85.7\%)$$

4. **Specificity**

$$\text{Specificity} = \frac{TN}{TN + FP}$$
$$\text{Specificity} = \frac{600}{600 + 50} = \frac{600}{650} = 0.923 \, (92.3\%)$$

5. **F1-Score**

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$
$$\text{F1-Score} = 2 \times \frac{0.857 \times 0.857}{0.857 + 0.857} = 2 \times \frac{0.734}{1.714} = 0.857 \, (85.7\%)$$

Summary of Metrics

Metric	Value
Accuracy	90%
Precision	85.7%
Recall	85.7%
Specificity	92.3%
F1-Score	85.7%

Mini Project (4): Loan Prediction Model Comparison

Name: Habibullah Farahmand

ID: DC24-M-03-102

Supervisor: Zainullah Ziarmal

Course Title: AI and Engineering

Submission Date: Nov-05-2024

Objective

The goal of this project is to develop and compare multiple machine learning models to predict whether a loan will be approved based on various features such as the applicant's gender, marital status, income, credit history, and more.

Skills and Concepts Highlighted

- **Data Preprocessing:** Handling missing values, encoding categorical variables, and feature selection.
- **Model Training & Evaluation:** Training multiple classifiers and evaluating them using accuracy, classification report, and confusion matrices.
- **Data Visualization:** Visualizing model performance using bar charts and heatmaps.
- **Model Comparison:** Comparing the effectiveness of different models in predicting loan approval.

Project Workflow

1. **Data Preprocessing:**
    - **Handling Missing Values:** Used imputation to fill missing values based on the mode (for categorical variables) and median (for numerical variables).
    - **Feature Encoding:** Applied one-hot encoding for categorical features like `Gender`, `Married`, `Education`, etc.
    - **Target Variable:** The target variable `Loan_Status` was encoded to a binary variable (`1` for 'Y' and `0` for 'N').
    - **Feature Selection:** Dropped unnecessary columns like `Loan_ID`.
2. **Model Selection and Training:**
    - Trained five different models:

        - **Logistic Regression**: A linear model used for binary classification.
        - **Decision Tree**: A model that splits data into subsets to make predictions.
        - **Random Forest**: An ensemble of decision trees that improves prediction accuracy by averaging the results.
        - **Support Vector Machine (SVM)**: A model that finds the hyperplane that best divides the data into classes.
        - **Naive Bayes**: A probabilistic classifier based on Bayes' theorem.
    - **Data Splitting:** The dataset was split into 70% training and 30% testing.
3. **Model Evaluation:**
    - **Accuracy:** We calculated the accuracy of each model to see how often the model predicted correctly.
    - **Classification Report:** Used `classification_report` to evaluate precision, recall, and F1-score for each model.
    - **Confusion Matrix:** Generated confusion matrices to visualize the model's performance with respect to false positives and false negatives.
4. **Visualization:**
    - **Accuracy Comparison:** A bar chart was created to compare the accuracy of all five models.
    - **Confusion Matrices:** Heatmaps were used to visualize the confusion matrices for each model to see how well they performed.

Code Highlights:

Data Preprocessing:

```
df_train['Gender'].fillna(df_train['Gender'].mode()[0], inplace=True)
df_train['Married'].fillna(df_train['Married'].mode()[0], inplace=True)
df_train['Dependents'].fillna('0', inplace=True)
df_train['Self_Employed'].fillna('No', inplace=True)
df_train['LoanAmount'].fillna(df_train['LoanAmount'].median(), inplace=True)
df_train['Loan_Amount_Term'].fillna(df_train['Loan_Amount_Term'].median(),
inplace=True)
df_train['Credit_History'].fillna(0, inplace=True)
df_train.drop(columns=['Loan_ID'], inplace=True)

# Encode categorical variables
df_train = pd.get_dummies(df_train, columns=['Gender', 'Married',
'Dependents', 'Education',
                                             'Self_Employed',
'Property_Area'], drop_first=True)
df_train['Loan_Status_Y'] = df_train['Loan_Status'].map({'Y': 1, 'N': 0})
df_train.drop(columns=['Loan_Status'], inplace=True)
```

Model Training and Evaluation:

```
models = {
    "Logistic Regression": LogisticRegression(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
```

```
    "SVM": SVC(),
    "Naive Bayes": GaussianNB()
}

results = {}

for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    results[model_name] = {
        "accuracy": accuracy,
        "classification_report": classification_report(y_test, y_pred,
output_dict=True),
        "confusion_matrix": confusion_matrix(y_test, y_pred)
    }
```

Visualization of Accuracy Comparison:

```
plt.figure(figsize=(10, 5))
accuracies = [results[model]['accuracy'] for model in models]
plt.bar(models.keys(), accuracies, color=['blue', 'green', 'red', 'purple',
'orange'])
plt.title("Model Comparison - Accuracy")
plt.ylabel("Accuracy")
plt.xticks(rotation=45)
plt.show()
```

Confusion Matrix Visualization:

```
for model_name, result in results.items():
    plt.figure(figsize=(5, 4))
    sns.heatmap(result['confusion_matrix'], annot=True, fmt='d',
cmap='Blues')
    plt.title(f"{model_name} - Confusion Matrix")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()
```

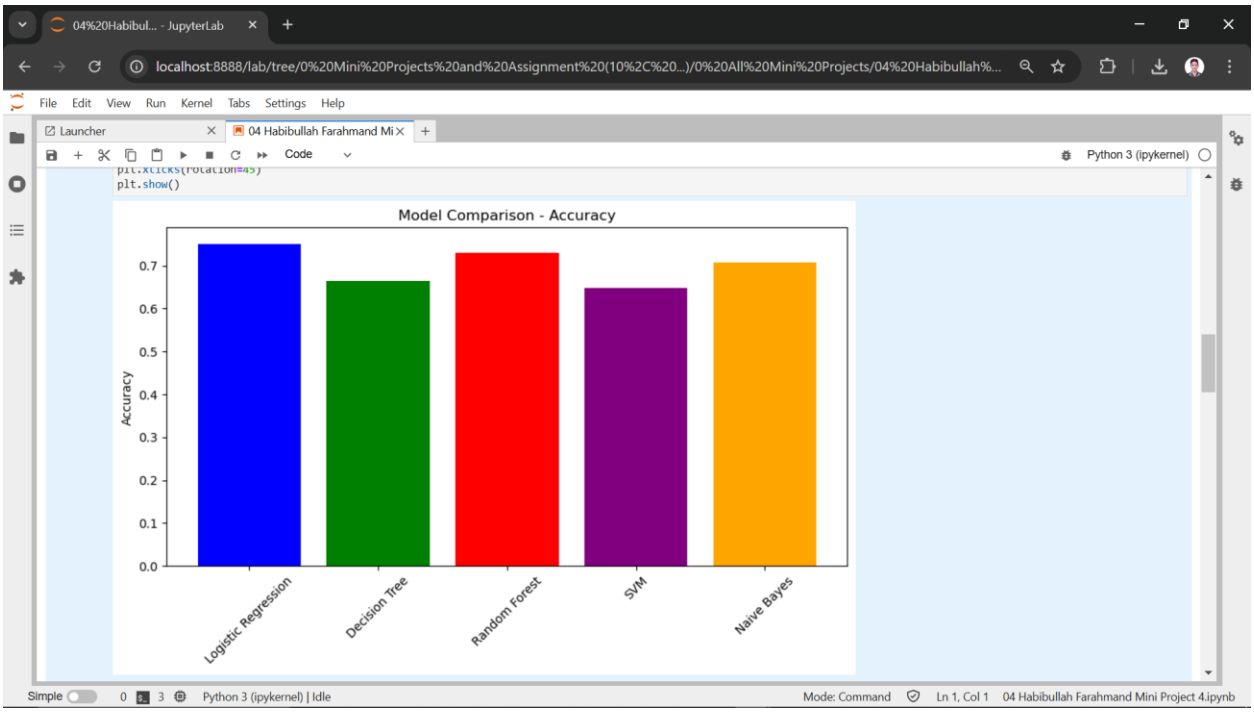
Results:

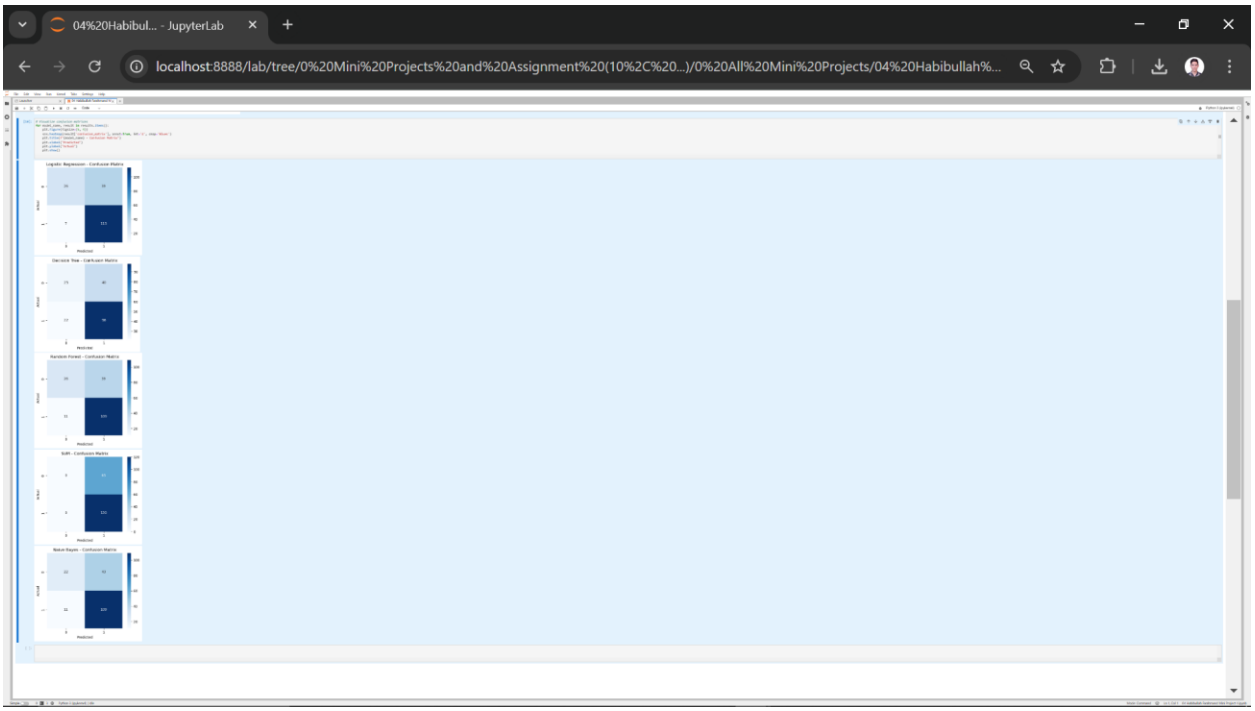
- **Accuracy Comparison:** The bar chart provides a clear visualization of the accuracy of each model.
- **Confusion Matrices:** The heatmaps help to understand the number of correct and incorrect predictions for each model, highlighting areas of improvement.

Next Steps:

1. **Hyperparameter Tuning:**
   Experiment with different hyperparameters for each model (e.g., `n_estimators` for Random Forest, `C` for SVM) to further improve performance.
2. **Cross-Validation:**
   Implement k-fold cross-validation to get a more reliable estimate of model performance.
3. **Feature Engineering:**
   Explore additional features such as applicant's income to improve predictive power.
4. **Model Deployment:**
   Deploy the best-performing model as a web service for real-time loan approval predictions.

Output Screenshots




04%20Habibul... - JupyterLab
localhost:8888/lab/tree/0%20Mini%20Projects%20and%20Assignment%20(10%2C%20...)/0%20All%20Mini%20Projects/04%20Habibullah%...

Mini Project (5): Customer Segmentation Using K-Means Clustering

Name: Habibullah Farahmand

ID: DC24-M-03-102

Supervisor: Zainullah Ziarmal

Course Title: AI and Engineering

Submission Date: Nov-07-2024

Steps Involved in the Process

1. **Data Loading and Exploration:**
   - The dataset `Online Retail.xlsx` contains transaction records, including customer ID, purchase quantity, unit price, and the country of the customer.
   - **Data Summary:**
     - The dataset has 541,909 entries initially, but after cleaning, we have 401,602 rows with 7 columns.
   - We checked for missing values and handled them, especially `CustomerID`, which had many missing values.
2. **Data Cleaning:**
   - Removed rows with missing `CustomerID` values, dropped the `Description` column (irrelevant for clustering), and removed duplicates.
   - Created a new `TotalPrice` column, representing the total amount spent by each customer (`Quantity * UnitPrice`).
3. **Customer Data Aggregation:**
   - Grouped the data by `CustomerID`, summing up the `Quantity` and `TotalPrice` to get an overview of each customer's total purchase behavior.
   - The aggregated data has 4,372 unique customers.
4. **Data Normalization:**
   - Normalized the `Quantity` and `TotalPrice` columns using `StandardScaler` to bring both features to the same scale before applying clustering.
5. **Clustering with K-Means:**
   - **Initial Clustering:**
     - Ran K-Means clustering with `n_clusters=3`, dividing customers into three groups based on normalized `Quantity` and `TotalPrice`.
   - Visualized the clusters using a scatter plot with centroids marked in red.
6. **Elbow Method:**
   - Used the **Elbow Method** to find the optimal number of clusters by plotting the inertia (within-cluster sum of squared distances) for different values of `k` (number of clusters).
   - The elbow method suggests the optimal number of clusters is around **4**.
7. **Final Clustering with k=4:**

- Ran K-Means again with 4 clusters and visualized the results using scatter plots.
- The clusters were color-coded, and the centroids were marked.

8. **Silhouette Score:**
   - Calculated the **Silhouette Score** for the clustering results, which measures the quality of clusters. The score of **0.93** indicates that the clustering is well-separated, meaning that customers are well-grouped into distinct clusters.

Key Insights and Results

1. **Clustering Outcome:**
   - The final clustering (with 4 clusters) gives a clear segmentation of customers based on their purchasing quantity and total spend.
   - Customers in different clusters represent different behaviors, which could be useful for targeted marketing strategies.
2. **Silhouette Score:**
   - The silhouette score of **0.93** suggests that the clusters are well-defined and that the K-Means algorithm has done a good job in grouping similar customers together.
3. **Visual Insights:**
   - The scatter plot with 4 clusters shows distinct customer groups, with the red `X` markers indicating the centroids of each cluster. Customers in each cluster have similar purchasing behaviors, which can help businesses identify their most valuable or least engaged customers.

Further Steps and Applications

1. **Customer Behavior Analysis:**
   - Analyze the clusters further to understand the characteristics of each group. For example, one group might represent high-spending customers, while another might represent occasional shoppers.
2. **Business Strategy:**
   - The results can be used for targeted marketing campaigns, personalized recommendations, and loyalty programs aimed at specific customer segments.
3. **Model Improvement:**
   - Experiment with different clustering algorithms such as **DBSCAN** or **Hierarchical Clustering** for comparison.
   - Explore more advanced features (e.g., frequency of purchases, recency) for customer segmentation.

Code Snippets of Key Steps

Data Aggregation:

```
customer_data = data.groupby('CustomerID').agg({
    'Quantity': 'sum',
    'TotalPrice': 'sum'
}).reset_index()
```

Normalization:

```
scaler = StandardScaler()
customer_data[['Quantity', 'TotalPrice']] =
scaler.fit_transform(customer_data[['Quantity', 'TotalPrice']])
```

K-Means Clustering:

```
kmeans = KMeans(n_clusters=4, random_state=0)
customer_data['Cluster'] = kmeans.fit_predict(customer_data[['Quantity',
'TotalPrice']])
```

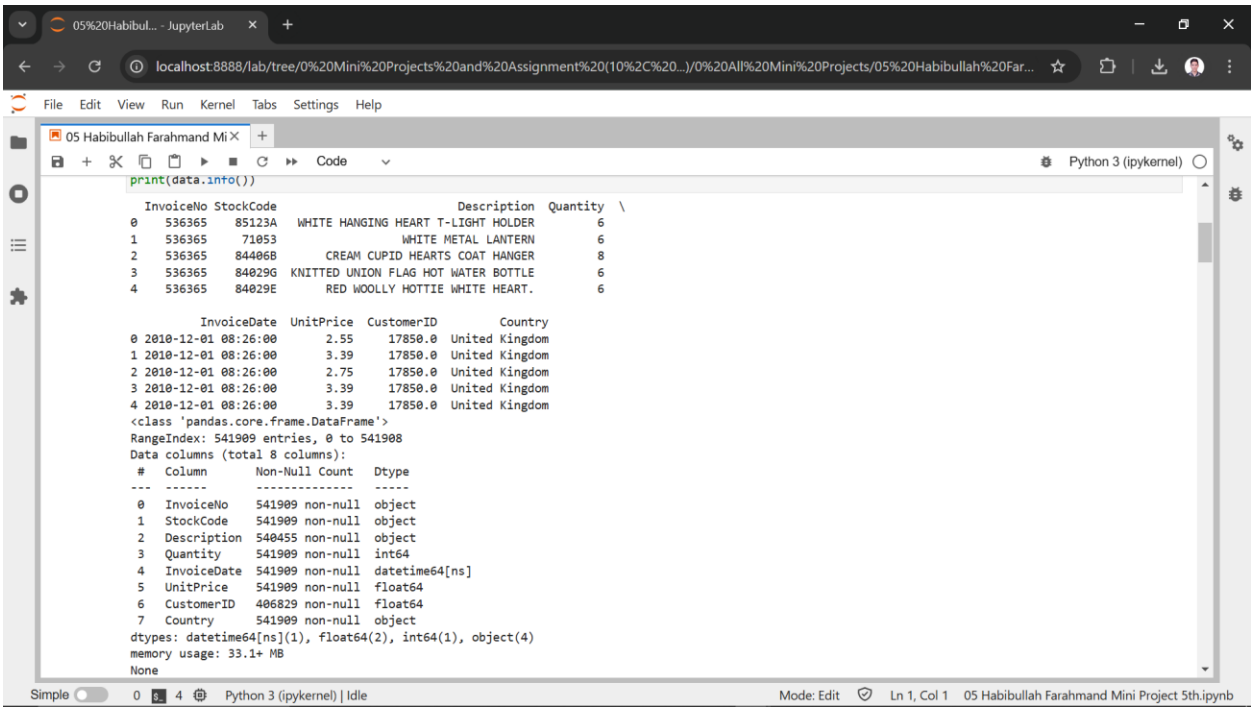
Elbow Method (for Optimal Clusters):

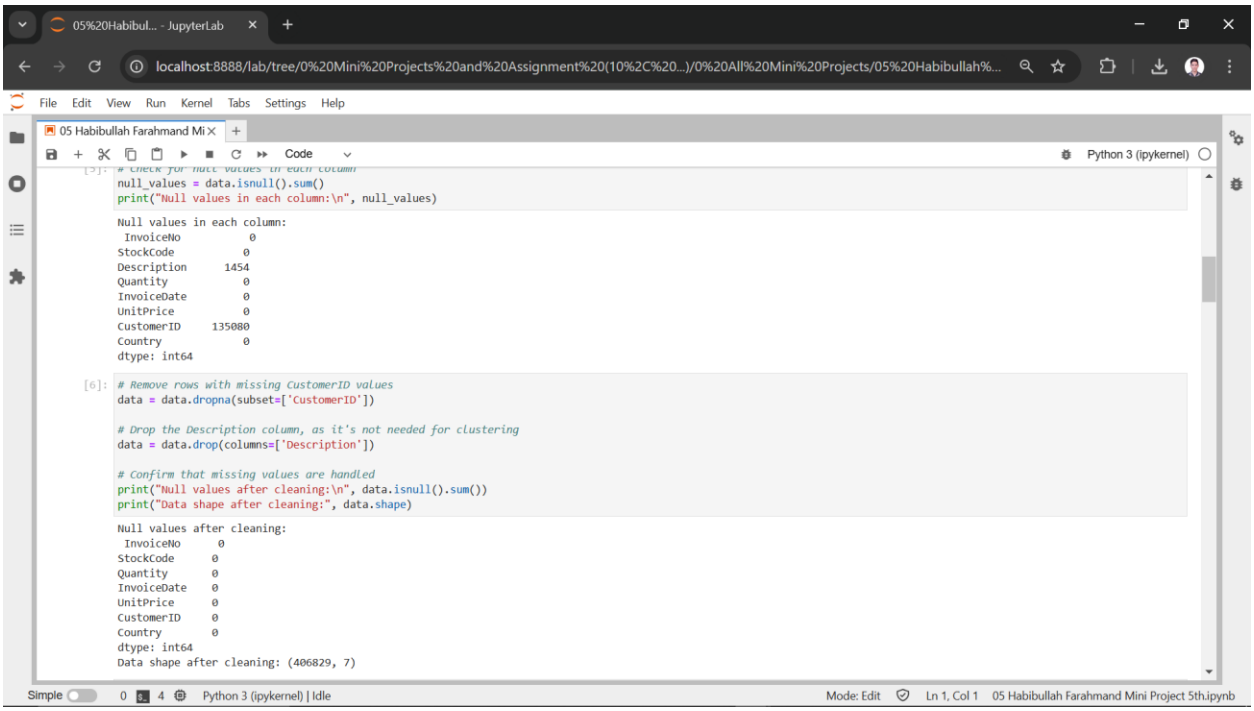
```
inertia = []
k_values = range(1, 11)

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=0)
    kmeans.fit(customer_data[['Quantity', 'TotalPrice']])
    inertia.append(kmeans.inertia_)

plt.plot(k_values, inertia, marker='o')
```

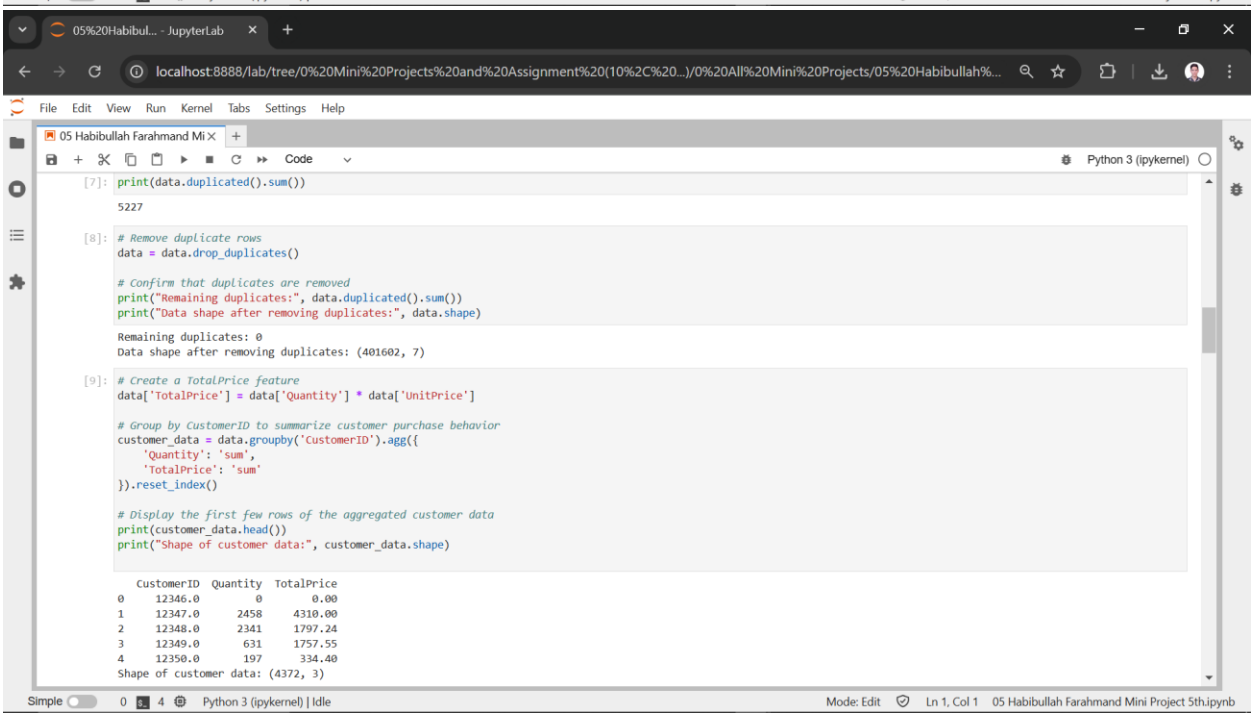
Silhouette Score:

```
from sklearn.metrics import silhouette_score
score = silhouette_score(customer_data[['Quantity', 'TotalPrice']],
kmeans.labels_)
print(f'Silhouette Score: {score}')
```

Output Screenshots


05%20Habibul... - JupyterLab
localhost:8888/lab/tree/0%20Mini%20Projects%20and%20Assignment%20(10%2C%20...)/0%20All%20Mini%20Projects/05%20Habibullah%20Far...
File Edit View Run Kernel Tabs Settings Help
05 Habibullah Farahmand Mi
Code
Python 3 (ipykernel)
print(data.info())
InvoiceNo StockCode Description Quantity \
0 536365 85123A WHITE HANGING HEART T-LIGHT HOLDER 6
1 536365 71053 WHITE METAL LANTERN 6
2 536365 84406B CREAM CUPID HEARTS COAT HANGER 8
3 536365 84029G KNITTED UNION FLAG HOT WATER BOTTLE 6
4 536365 84029E RED WOOLLY HOTTIE WHITE HEART. 6
InvoiceDate UnitPrice CustomerID Country
0 2010-12-01 08:26:00 2.55 17850.0 United Kingdom
1 2010-12-01 08:26:00 3.39 17850.0 United Kingdom
2 2010-12-01 08:26:00 2.75 17850.0 United Kingdom
3 2010-12-01 08:26:00 3.39 17850.0 United Kingdom
4 2010-12-01 08:26:00 3.39 17850.0 United Kingdom
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
Column Non-Null Count Dtype
0 InvoiceNo 541909 non-null object
1 StockCode 541909 non-null object
2 Description 540455 non-null object
3 Quantity 541909 non-null int64
4 InvoiceDate 541909 non-null datetime64[ns]
5 UnitPrice 541909 non-null float64
6 CustomerID 406829 non-null float64
7 Country 541909 non-null object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
None
Simple
Python 3 (ipykernel) | Idle
Mode: Edit
Ln 1, Col 1
05 Habibullah Farahmand Mini Project 5th.ipynb


05%20Habibul... - JupyterLab
localhost:8888/lab/tree/0%20Mini%20Projects%20and%20Assignment%20(10%2C%20...)/0%20All%20Mini%20Projects/05%20Habibullah%...
File Edit View Run Kernel Tabs Settings Help
05 Habibullah Farahmand Mi
Code
Python 3 (ipykernel)
# Check for null values in each column
null_values = data.isnull().sum()
print("Null values in each column:\n", null_values)
Null values in each column:
InvoiceNo 0
StockCode 0
Description 1454
Quantity 0
InvoiceDate 0
UnitPrice 0
CustomerID 135080
Country 0
dtype: int64
[6]:
# Remove rows with missing CustomerID values
data = data.dropna(subset=['CustomerID'])
# Drop the Description column, as it's not needed for clustering
data = data.drop(columns=['Description'])
# Confirm that missing values are handled
print("Null values after cleaning:\n", data.isnull().sum())
print("Data shape after cleaning:", data.shape)
Null values after cleaning:
InvoiceNo 0
StockCode 0
Quantity 0
InvoiceDate 0
UnitPrice 0
CustomerID 0
Country 0
dtype: int64
Data shape after cleaning: (406829, 7)
Simple
Python 3 (ipykernel) | Idle
Mode: Edit
Ln 1, Col 1
05 Habibullah Farahmand Mini Project 5th.ipynb


05%20Habibul... - JupyterLab
localhost:8888/lab/tree/0%20Mini%20Projects%20and%20Assignment%20(10%2C%20...)/0%20All%20Mini%20Projects/05%20Habibullah%...
File Edit View Run Kernel Tabs Settings Help
05 Habibullah Farahmand Mi
Code
Python 3 (ipykernel)
[7]:
print(data.duplicated().sum())
5227
[8]:
# Remove duplicate rows
data = data.drop_duplicates()
# Confirm that duplicates are removed
print("Remaining duplicates:", data.duplicated().sum())
print("Data shape after removing duplicates:", data.shape)
Remaining duplicates: 0
Data shape after removing duplicates: (401602, 7)
[9]:
Create a TotalPrice feature
data['TotalPrice'] = data['Quantity'] * data['UnitPrice']
# Group by CustomerID to summarize customer purchase behavior
customer_data = data.groupby('CustomerID').agg({
'Quantity': 'sum',
'TotalPrice': 'sum'
}).reset_index()
# Display the first few rows of the aggregated customer data
print(customer_data.head())
print("Shape of customer data:", customer_data.shape)
CustomerID Quantity TotalPrice
0 12346.0 0 0.00
1 12347.0 2458 4310.00
2 12348.0 2341 1797.24
3 12349.0 631 1757.55
4 12350.0 197 334.40
Shape of customer data: (4372, 3)
Simple
Python 3 (ipykernel) | Idle
Mode: Edit
Ln 1, Col 1
05 Habibullah Farahmand Mini Project 5th.ipynb

05%20Habibul... - JupyterLab

localhost:8888/lab/tree/0%20Mini%20Projects%20and%20Assignment%20(10%2C%20...)/0%20All%20Mini%20Projects/05%20Habibullah%...

File Edit View Run Kernel Tabs Settings Help

05 Habibullah Farahmand Mi

Code

Python 3 (ipykernel)

```
# Normalize Quantity and TotalPrice columns
customer_data[['Quantity', 'TotalPrice']] = scaler.fit_transform(customer_data[['Quantity', 'TotalPrice']])

# Display the first few rows of the normalized data
print(customer_data.head())
```

```
   CustomerID  Quantity  TotalPrice
0     12346.0 -0.239519   -0.230417
1     12347.0  0.286579    0.294087
2     12348.0  0.261537   -0.011703
3     12349.0 -0.104463   -0.016533
4     12350.0 -0.197354   -0.189723
```

[11]:

```
# Initialize K-Means with 3 clusters
kmeans = KMeans(n_clusters=3, random_state=0)

# Fit the model to the data
customer_data['Cluster'] = kmeans.fit_predict(customer_data[['Quantity', 'TotalPrice']])

# Display the first few rows with cluster assignments
print(customer_data.head())
print("Cluster centers:\n", kmeans.cluster_centers_)
```

```
   CustomerID  Quantity  TotalPrice  Cluster
0     12346.0 -0.239519   -0.230417        0
1     12347.0  0.286579    0.294087        0
2     12348.0  0.261537   -0.011703        0
3     12349.0 -0.104463   -0.016533        0
4     12350.0 -0.197354   -0.189723        0
Cluster centers:
 [[-0.06278401 -0.06014175]
 [20.48962714 23.60795438]
 [ 6.29757303  5.29538613]]
```

Simple 0 4 Python 3 (ipykernel) | Idle Mode: Edit Ln 1, Col 1 05 Habibullah Farahmand Mini Project 5th.ipynb

05%20Habibul... - JupyterLab

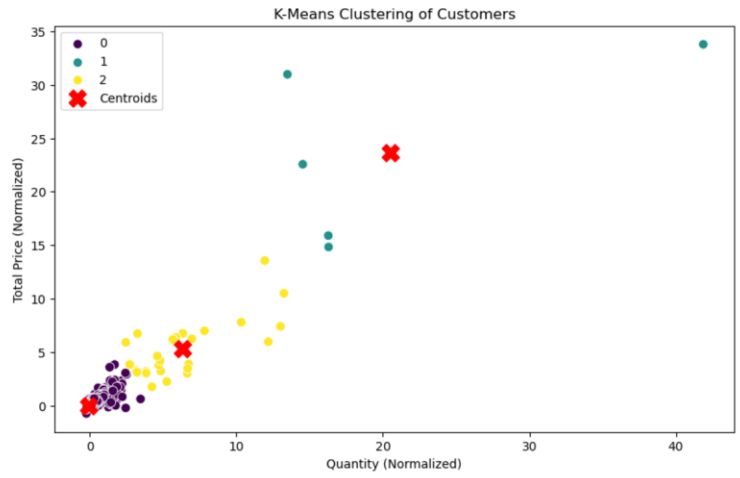
localhost:8888/lab/tree/0%20Mini%20Projects%20and%20Assignment%20(10%2C%20...)/0%20All%20Mini%20Projects/05%20Habibullah%...

File Edit View Run Kernel Tabs Settings Help

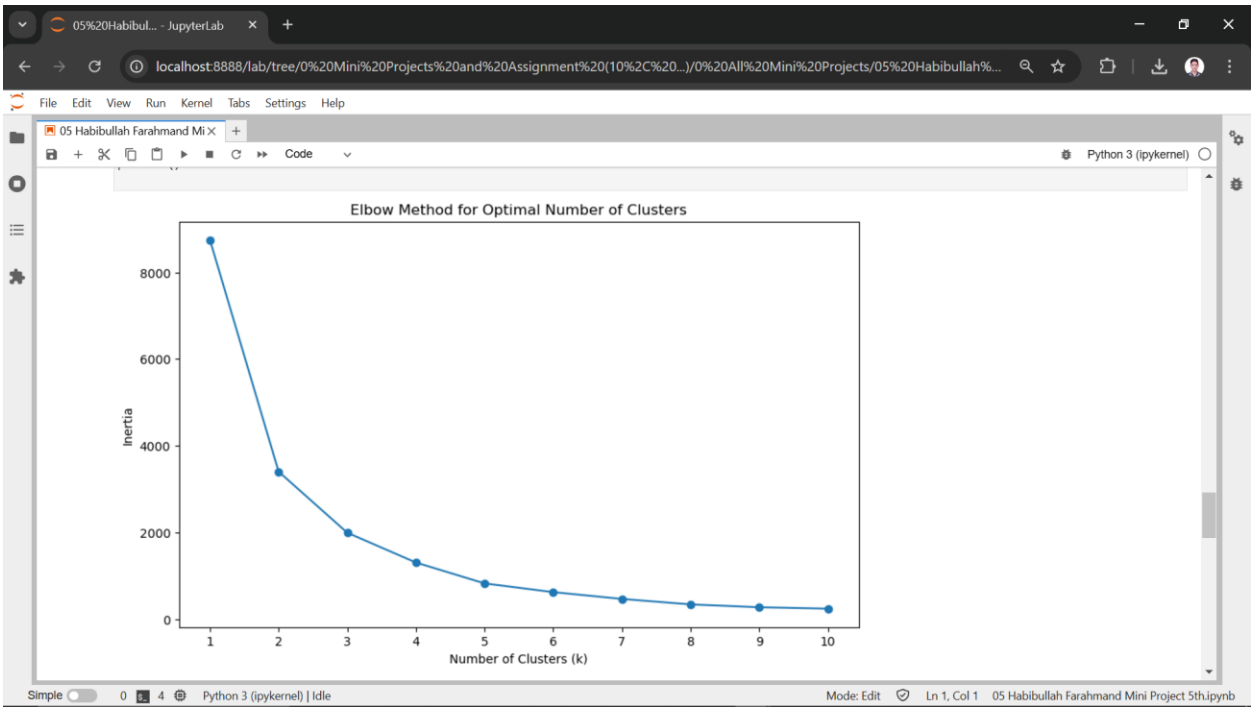
05 Habibullah Farahmand Mi

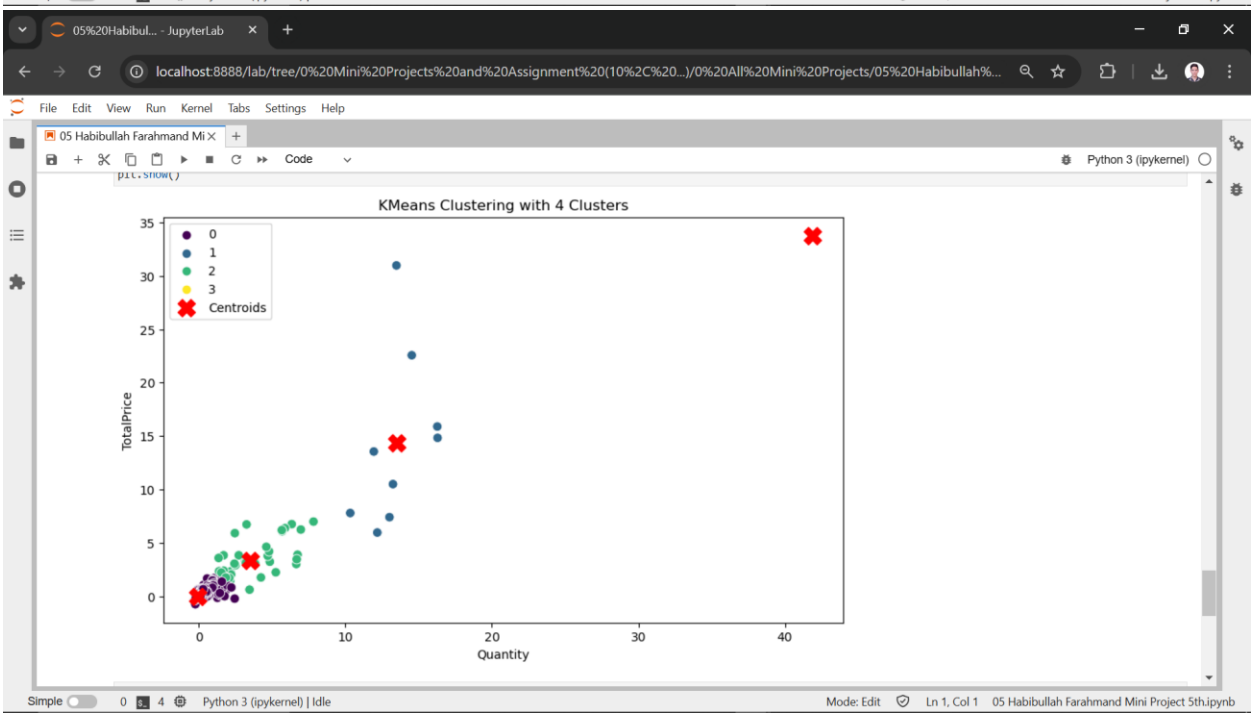
Code

Python 3 (ipykernel)



Simple 0 4 Python 3 (ipykernel) | Idle Mode: Edit Ln 1, Col 1 05 Habibullah Farahmand Mini Project 5th.ipynb


05%20Habibul... - JupyterLab
localhost:8888/lab/tree/0%20Mini%20Projects%20and%20Assignment%20(10%2C%20...)/0%20All%20Mini%20Projects/05%20Habibullah%...
File Edit View Run Kernel Tabs Settings Help
05 Habibullah Farahmand Mi
Code
Python 3 (ipykernel)
Elbow Method for Optimal Number of Clusters
Inertia
Number of Clusters (k)
Simple
Python 3 (ipykernel) | Idle
Mode: Edit
Ln 1, Col 1
05 Habibullah Farahmand Mini Project 5th.ipynb


05%20Habibul... - JupyterLab
localhost:8888/lab/tree/0%20Mini%20Projects%20and%20Assignment%20(10%2C%20...)/0%20All%20Mini%20Projects/05%20Habibullah%...
File Edit View Run Kernel Tabs Settings Help
05 Habibullah Farahmand Mi
Code
Python 3 (ipykernel)
KMeans Clustering with 4 Clusters
0
1
2
3
Centroids
TotalPrice
Quantity
Simple
Python 3 (ipykernel) | Idle
Mode: Edit
Ln 1, Col 1
05 Habibullah Farahmand Mini Project 5th.ipynb

Mini Project (6): Image Classification Using Convolutional Neural Networks (CNN) on MNIST Dataset

Name: Habibullah Farahmand

ID: DC24-M-03-102

Supervisor: Zainullah Ziarmal

Course Title: AI and Engineering

Submission Date: Nov-10-2024

---

1. Skill Representation:

- **Deep Learning Models:** The core of this project is building a Convolutional Neural Network (CNN) using TensorFlow/Keras. This demonstrates your ability to create and train deep learning models for image classification.
- **Data Engineering and Preprocessing:**
  - Data is loaded from IDX file format, normalized, and reshaped to be compatible with the neural network input.
  - Use of image preprocessing techniques to scale pixel values between 0 and 1 and reshape the data.
- **Supervised Learning:**
  - The project uses supervised learning techniques where the model is trained on labeled image data (MNIST dataset), and the model learns to classify handwritten digits.

2. Real-World Application:

- **Problem Addressed:** The project aims to classify handwritten digits, a classic problem in machine learning that is fundamental in image recognition tasks.
- **Practical Use:**
  - **Real-World Use Cases:** The ability to classify handwritten digits accurately can be applied in areas like:
    - **Postal services:** Sorting of mail based on handwritten zip codes.
    - **Banking:** Processing handwritten checks or forms.
    - **Document digitization:** Converting handwritten forms into digital formats.
- **Scalability:** This project can be extended to other image classification tasks, such as object recognition, facial recognition, or even medical image analysis.

3. Creative Freedom:

- **Model Optimization and Experimentation:** The architecture can be expanded by adding more convolutional layers, adjusting the number of filters, or incorporating techniques like dropout to prevent overfitting.
    - **Future Extensions:** Experiment with different architectures such as ResNet or VGG and evaluate performance on other datasets, such as CIFAR-10 or Fashion-MNIST.

4. Variety:

- **Supervised Learning:** This is a classic supervised learning task, where the model learns from labeled data (images with corresponding digit labels).
- **Data Engineering:** The project demonstrates preprocessing techniques such as normalization, reshaping, and loading image data from a custom file format (IDX).
- **Ethical Considerations in AI Design:** In image classification tasks, it's crucial to ensure the dataset is balanced and not biased. The MNIST dataset is quite balanced, but when applied to real-world tasks, careful consideration of data diversity (e.g., handwriting styles) is needed to avoid biased models.

---

5. Benefits for Students:

- **Showcasing Talent:** This project showcases your ability to build deep learning models, preprocess image data, and apply CNNs to solve a well-known classification problem. It's a solid example of your skills in computer vision.
- **Hands-On Learning:** Students gain experience in building and training CNNs, working with image data, and using deep learning libraries like TensorFlow/Keras.
- **Confidence Building:** Successfully building and evaluating a CNN model for image classification boosts confidence in handling more complex image recognition tasks.

---

6. Implementation in the Program:

- **Guided Project Phases:**
    - **Step 1**: Loading and preprocessing the MNIST dataset.
    - **Step 2**: Defining and compiling the CNN model.
    - **Step 3**: Training the model with training data and validating with test data.
    - **Step 4**: Evaluating the model's performance using accuracy and loss metrics.
    - **Step 5**: Visualizing the training process with accuracy and loss plots.
    - **Step 6**: Making predictions and validating them with actual labels.
- **Mentorship and Feedback:** Mentors can help students fine-tune the CNN model by adjusting hyperparameters such as the number of layers, the kernel size of convolutional layers, or the learning rate of the optimizer. They can also help with interpreting the model's performance and guiding students to avoid overfitting.

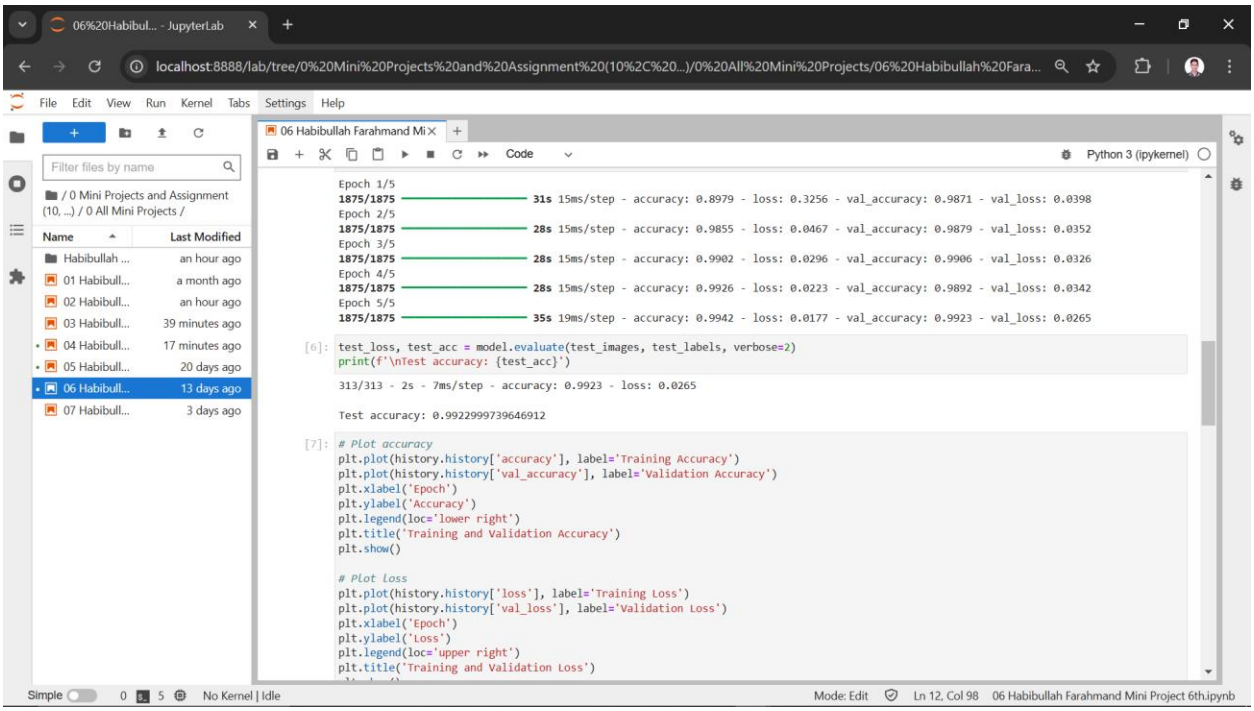
- **Collaboration Opportunities:** Students can collaborate to experiment with different models, optimize the existing CNN architecture, or extend the project to work with other datasets like CIFAR-10 for more complex image classification.
- **Portfolio Presentation:**
    - The project would make an excellent portfolio piece, with clear visualizations of training and validation accuracy/loss.
    - A demonstration of predictions on test images can be included to showcase the model's real-world applicability.

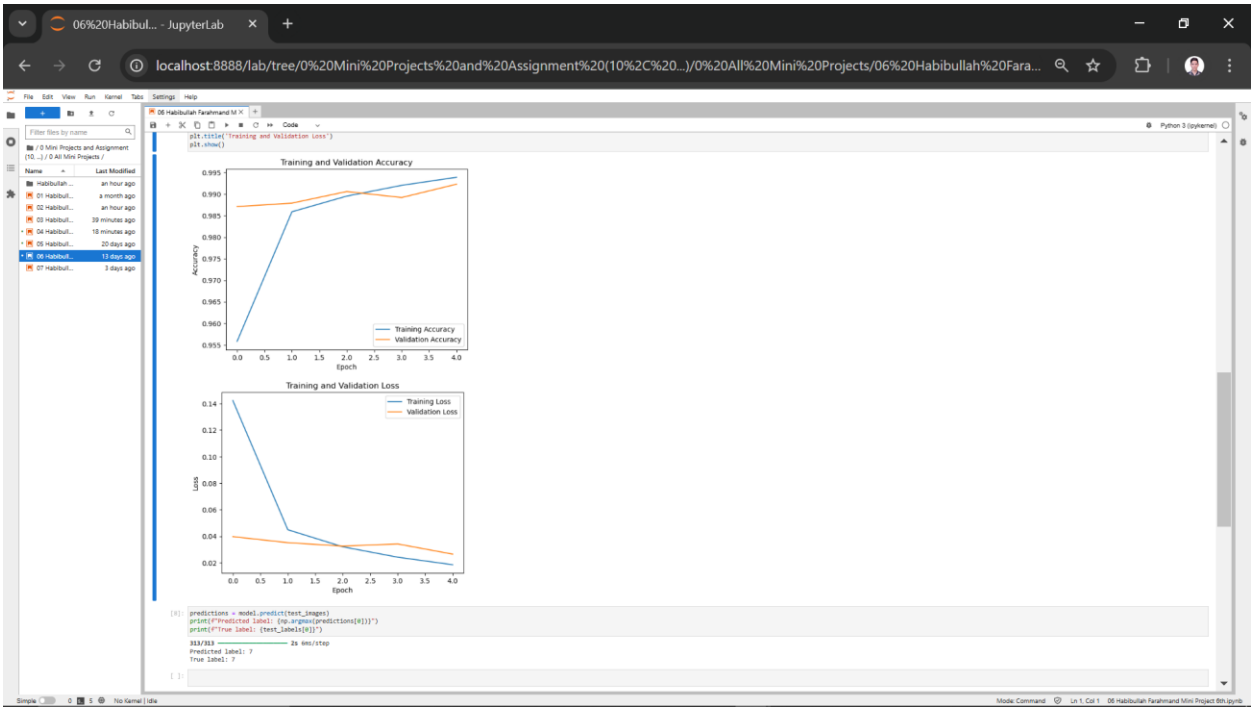
---

7. Project Documentation:

- **Introduction:** This project utilizes a Convolutional Neural Network (CNN) to classify images from the MNIST dataset of handwritten digits. The goal is to build a model that learns to identify the digits 0-9 from a set of training images and make accurate predictions on unseen test images.
- **Skills:**
    - **Python** for data manipulation and model building.
    - **TensorFlow/Keras** for building and training the deep learning model.
    - **Matplotlib** for data visualization.
- **Documentation/Case Study:**
    - **Loading Data**: Custom IDX loader is implemented to read the MNIST dataset files.
    - **Model Architecture**: A CNN model is used, with multiple convolutional layers, pooling layers, and a fully connected dense layer for classification.
    - **Model Performance**: Achieved a test accuracy of 99.23% and a loss of 0.0265, indicating excellent performance on the test set.
- **Visuals:**
    - **Accuracy and Loss Plots**: Graphs of training and validation accuracy and loss, which help in visualizing the model's learning progress.
    - **Prediction Output**: Sample predictions and true labels on test data to verify the model's accuracy.
- **Links:**
    - GitHub repository for the project code.
    - Screenshots of the model's predictions on test images.

Output Screenshots:


06%20Habibul... - JupyterLab
localhost:8888/lab/tree/0%20Mini%20Projects%20and%20Assignment%20(10%2C%20...)/0%20All%20Mini%20Projects/06%20Habibullah%20Fara...
File Edit View Run Kernel Tabs Settings Help
06 Habibullah Farahmand Mi
Code
Python 3 (ipykernel)
Filter files by name
/ 0 Mini Projects and Assignment (10, ...) / 0 All Mini Projects /
Name
Last Modified
Habibullah ... an hour ago
01 Habibull... a month ago
02 Habibull... an hour ago
03 Habibull... 39 minutes ago
04 Habibull... 17 minutes ago
05 Habibull... 20 days ago
06 Habibull... 13 days ago
07 Habibull... 3 days ago
Epoch 1/5
1875/1875 31s 15ms/step - accuracy: 0.8979 - loss: 0.3256 - val_accuracy: 0.9871 - val_loss: 0.0398
Epoch 2/5
1875/1875 28s 15ms/step - accuracy: 0.9855 - loss: 0.0467 - val_accuracy: 0.9879 - val_loss: 0.0352
Epoch 3/5
1875/1875 28s 15ms/step - accuracy: 0.9902 - loss: 0.0296 - val_accuracy: 0.9906 - val_loss: 0.0326
Epoch 4/5
1875/1875 28s 15ms/step - accuracy: 0.9926 - loss: 0.0223 - val_accuracy: 0.9892 - val_loss: 0.0342
Epoch 5/5
1875/1875 35s 19ms/step - accuracy: 0.9942 - loss: 0.0177 - val_accuracy: 0.9923 - val_loss: 0.0265
[6]:
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
print(f'\nTest accuracy: {test_acc}')
313/313 - 2s - 7ms/step - accuracy: 0.9923 - loss: 0.0265
Test accuracy: 0.9922999739646912
[7]:
# Plot accuracy
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')
plt.show()
# Plot Loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
Simple 0 5 No Kernel | Idle
Mode: Edit Ln 12, Col 98 06 Habibullah Farahmand Mini Project 6th.ipynb


06%20Habibul... - JupyterLab
localhost:8888/lab/tree/0%20Mini%20Projects%20and%20Assignment%20(10%2C%20...)/0%20All%20Mini%20Projects/06%20Habibullah%20Fara...
Training and Validation Accuracy
Accuracy
Epoch
Training Accuracy
Validation Accuracy
Training and Validation Loss
Loss
Epoch
Training Loss
Validation Loss
[8]:
predictions = model.predict(test_images)
print(f"Predicted label: {np.argmax(predictions[0])}")
print(f"True label: {test_labels[0]}")
313/313 2s 6ms/step
Predicted label: 7
True label: 7
[ ]:

Mini Project (7): Retail Sales Forecasting Using ARIMA

Name: Habibullah Farahmand

ID: DC24-M-03-102

Supervisor: Zainullah Ziarmal

Course Title: AI and Engineering

Submission Date: Nov-18-2024

---

1. Skill Representation:

- **Machine Learning Models:** This project utilizes time series forecasting with ARIMA (AutoRegressive Integrated Moving Average) to predict future retail sales based on historical data.
- **Data Visualization Dashboards:** The project incorporates visualizations using `matplotlib` to display both historical and forecasted sales data in a clear, actionable format.
- **Data Cleaning and Preprocessing:** The dataset is preprocessed to ensure that the `Date` column is in the correct format and to aggregate sales by date.
- **Supervised Learning:** Though ARIMA is more of a statistical method than traditional machine learning, the model can be viewed as supervised learning, where historical data (inputs) is used to predict future values (outputs).

2. Real-World Application:

- **Problem Addressed:** The primary objective of this project is to forecast future retail sales, which is a practical application in businesses to plan for inventory, promotions, and staffing.
- **Practical Use:**
    - **Retail Businesses**: Helps companies anticipate demand, optimize inventory levels, and enhance supply chain operations.
    - **Data-Driven Decision Making**: This project aids in making informed decisions about sales, promotions, and resource allocation.

3. Creative Freedom:

- The project was designed to leverage a real-world retail sales dataset, which can be easily applied to various business settings. While the dataset is generic, a creative spin could involve adjusting the ARIMA model to fit specific sectors like fashion, electronics, or groceries.

4. Variety:

- **Supervised Learning:** The ARIMA model employs historical sales data to predict future values, making it a classic example of supervised learning.
- **Data Engineering and Processing Pipelines:** The dataset is cleaned, processed, and split into training and testing sets to prepare it for modeling.
- **Ethical Considerations in AI Design:** This project, while more statistical in nature, highlights the importance of clean and accurate data to avoid biases in predictions, as any data issues could lead to inaccurate forecasting and poor business decisions.

---

5. Benefits for Students:

- **Showcasing Talent:** This project provides a solid demonstration of your ability to work with time series data and apply forecasting techniques—skills that are highly sought after in data science and analytics roles.
- **Hands-On Learning:** The project involves practical application of statistical models, helping to solidify the student's understanding of time series analysis and ARIMA, enhancing their readiness for the job market.
- **Confidence Building:** Completing this project gives the student confidence in their ability to build and deploy models to solve real-world problems, providing a sense of achievement in the domain of time series forecasting.

---

6. Implementation in the Program:

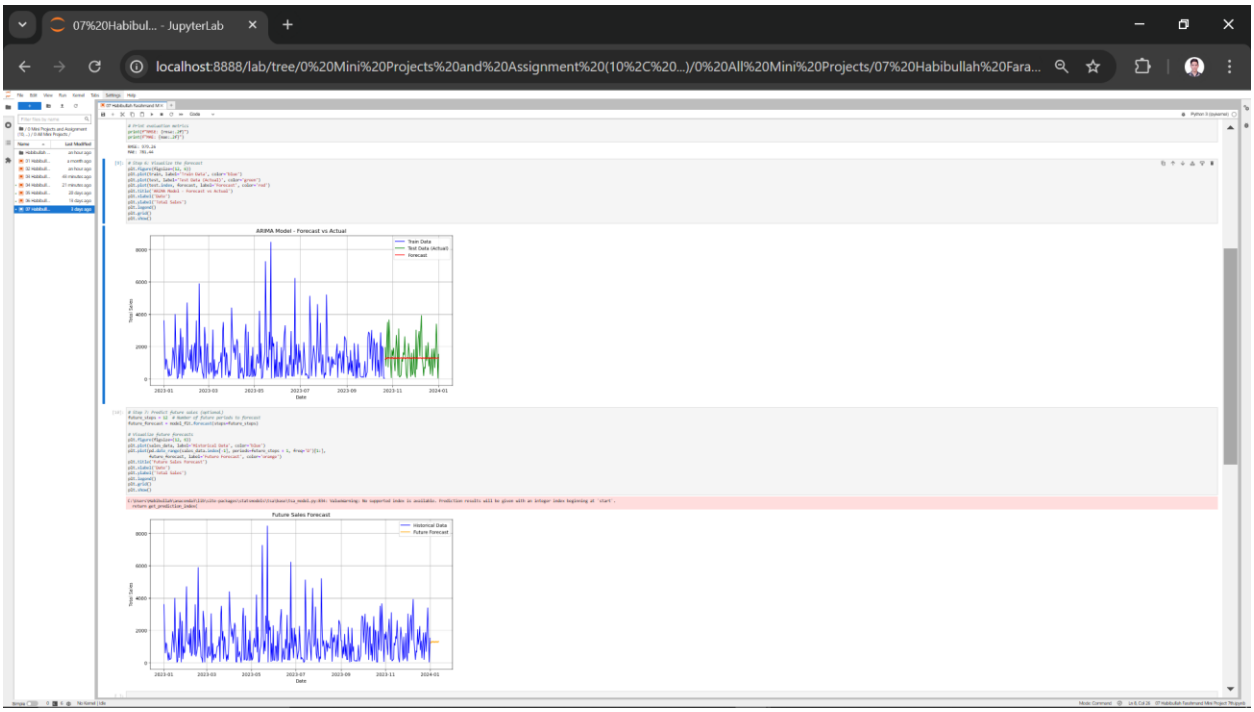
- **Guided Project Phases:**
  - **Step 1**: Loading and cleaning the dataset.
  - **Step 2**: Preprocessing the data for time series analysis.
  - **Step 3**: Splitting data into training and testing sets.
  - **Step 4**: Fitting the ARIMA model.
  - **Step 5**: Evaluating the model's performance.
  - **Step 6**: Visualizing the forecast and comparing actual vs predicted sales.
- **Mentorship and Feedback:** Mentors can provide guidance on improving model accuracy, selecting appropriate hyperparameters (ARIMA's `p`, `d`, and `q` values), and interpreting the evaluation metrics (RMSE, MAE).
- **Collaboration Opportunities:** Students can collaborate by working on different datasets, comparing their models and techniques, or adding additional features like external factors (e.g., promotions or seasonality) to the sales prediction model.
- **Portfolio Presentation:** A visual showcase of the time series predictions (using plots) and model performance metrics like RMSE and MAE can be used to present this project in a final portfolio presentation.

---

7. Project Documentation:

- **Introduction:** This project aims to forecast retail sales using ARIMA, helping businesses plan for demand fluctuations. The model predicts sales based on historical data, providing actionable insights into future sales.
- **Skills:**
  - **Python** for data manipulation and visualization.
  - **ARIMA** for time series forecasting.
  - **Matplotlib** for data visualization.
  - **Pandas** for data manipulation and processing.
- **Documentation/Case Study:**
  - The dataset is cleaned and preprocessed for time series analysis. The model's performance is evaluated using RMSE and MAE, providing insight into forecast accuracy.
  - Evaluation metrics: RMSE = 979.26, MAE = 781.44.
  - The results are presented visually using line plots, making it easy to compare actual vs predicted sales.
- **Visuals:**
  - **Forecast Visualization**: A plot showing the actual sales and the forecasted sales.
  - **Future Forecast**: A visualization showing future sales predictions.
- **Links:**
  - GitHub repository for the code.
  - Screenshots of the model's output and visualizations.

Output Screenshots:



I am pleased to confirm the submission of the mini projects documentation and its code, prepared and reviewed by me, Trainer Zainullah Ziarmal with the code 008. The zip encapsulates all required details, ensuring clarity and alignment with the outlined projects objectives.

Best regards,

Zainullah Ziarmal

Zainullahziarmal.cs786@gmail.com