

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi – 590 018.



An Internship Report

On

“Machine learning using Python”

01.02.2023 to 05.03.2023

Submitted in partial fulfilment of the for the award of the degree of

Bachelor of Engineering

In

Computer Science and Engineering

Submitted by

PRAKHAR KUMAR CHANDRAKER

1VI19CS070



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VEMANA INSTITUTE OF TECHNOLOGY

BENGALURU – 560034

2022-2023

Karnataka Reddy Jana Sangha®

VEMANA INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University, Belagavi)

Koramangala, Bengaluru-560034.



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Internship work entitled “**MACHINE LEARNING USING PYTHON**” is a bonafide work carried out by **Mr. PRAKHAR KUMAR CHANDRAKER (1VI19CS070)** during the academic year 2022-23 in partial fulfilment of the requirement for the award of **Bachelor of Engineering in Computer Science and Engineering** of the **Visvesvaraya Technological University, Belagavi**. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The internship report has been approved as it satisfies the academic requirements in respect of the Internship prescribed for the said degree.

Guide

Ms. Veena G

Head of the Department

Dr. M. Ramakrishna

Principal

Dr. Vijayasimha Reddy B.G

External Viva

Name of the Examiner

Signature with date

1. _____

2. _____

Internship Certificate



Anspro Technologies

Find a better solution

RefNo: ANSP22ML157
Date: 6-03-2023

Internship Completion Letter

This is to certify that **Mr. Prakhar Kumar Chandraker**, USN **1VI19CS070**, Vemana Institute of Technology, Computer Science and Engineering, was working with us from 01-02-2023 to 05-03-2023. He was working as Intern on Machine Learning using Python. He has worked on “**LPG Usage Prediction**” project.

During the internship, intern was found to be good and disciplined.

Authorized Signatory

Yours faithfully,



For Anspro Technologies
Authorized Signatory

#7, 1st Floor
100 ft Ring Road B.T.M 2nd Stage,
Near Jayadeva Metro Station,
Bengaluru-560076

www.ansprotech.com
E-mail : info@ansprotech.com
Mob : 9886832434

ABSTRACT

The use of Liquefied Petroleum Gas (LPG) as a fuel source for cooking and heating is prevalent in many households across India. However, the supply of LPG through pipelines is not possible in many regions due to a shortage of LPG, which makes it necessary for households to rely on LPG cylinder refill deliveries. Traditionally, customers have to book refills through IVRS, which can be inconvenient due to busy schedules, and there is always a chance of foul play by the gas agencies or distributors. To address these issues, an automated system that will eliminate the need for human intervention in the refill booking process is proposed. The proposed system will be based on predictive analytics, which will leverage usage data from previous LPG cylinder refills to predict the next refill date for each family. By analyzing the frequency of refills, the amount of LPG used, and the number of members in each household, a model that can accurately predict when the next refill is due for each family is developed. The system will be designed to automatically generate refill requests and notify the gas agencies or distributors when a refill is required. This will streamline the refill process and eliminate the need for customers to book refills manually. Additionally, since the system will be based on data analytics, it will reduce the chances of foul play by gas agencies or distributors, and customers will be able to receive refills in a timely and hassle-free manner. The proposed system will be implemented using advanced technologies such as machine learning, data analytics, and automation. The system will be scalable and will be able to handle a large number of households. The system will also be user-friendly, and customers will be able to access their refill history and upcoming refill dates through a dedicated mobile application or web portal. In conclusion, the proposed automated refill booking system for LPG cylinders in India will significantly improve the customer experience by eliminating the need for manual bookings, reducing the chances of foul play, and ensuring timely and hassle-free refills.

ACKNOWLEDGEMENT

I sincerely thank Visvesvaraya Technology University for providing a platform to do the internship work.

I express my sincere thanks to **Dr. Vijayasimha Reddy B G**, Principal, Vemana Institute of Technology, Bengaluru, for providing necessary facilities and motivation to carry out internship work successfully.

I express heartfelt gratitude and humble thanks to **Dr. M. Ramakrishna**, Professor and Head, Computer Science and Engineering, Vemana Institute of Technology, for his constant encouragement, inspiration and help to carry out internship work successfully.

I am very thankful to my external guide, **Sumit Saha**, Manager, at Anspro Technologies who has given in-time valuable instructions and put me in contact with experts in the field, with extensive guidance regarding practical issues.

I would like to express my sincere gratitude towards my internal guide, **Prof. Veena G**, Assistant Professor for providing encouragement and inspiration throughout the internship.

I thank internship coordinators **Prof. Naveen H S** and **Prof. Kavitha Bai A.S** for their cooperation and support during the internship work

I am thankful to all the teaching and non-teaching staff members of Computer Science and Engineering department for their help and much needed support throughout the internship.

Prakhar Kumar Chandraker

1VI19CS070

LIST OF FIGURES

Figure No.	Title	Page no.
4.2	Architecture diagram	10
5.1	Series Data Structure using Pandas	12
5.1	Series Data Structure using Pandas	13
5.1	Reading and Writing Data using Pandas	13
5.1	Data Selection and Filtering	14
5.2	Creating Arrays Using NumPy	15
5.2	Creating 2D Arrays Using NumPy	15
5.2	Array Operations Using NumPy	16
5.2	Indexing and Slicing Using NumPy	16
5.2	Indexing and Slicing an Array Using NumPy	17
5.2	Broadcasting using NumPy	17
5.2	Shape Manipulation using NumPy	18
5.3	Linear Regression Using Statsmodels	19
5.3	Time Series Analysis Using Statsmodels	20
5.3	Hypothesis Testing Using Statsmodels	20
5.3	Time Series Forecasting Using Statsmodels	21
5.3	Generalized Linear Models Using Statsmodels	22
6.1	Importing the modules	23
6.2	Reading the Dataset	24
6.3	Reading the Dataset	24
6.4	Checking the Dimensions of Dataset	24
6.5	Summary of the Dataset	25
6.6	Summary of the basic statistical properties	25
6.7	Last 180 rows and columns of Dataset	26
6.8	First 180 rows and columns of Dataset	26
6.9	Unique Columns of Dataset	27
6.10	Splitting the Dataframe into X and y	27
6.11	Columns of Dataset	27

LIST OF FIGURES

Figure No.	Title	Page no.
6.12	Storing Subsets of Dataframe into X1 and y1	28
6.13	Train Test Split	28
6.14	Importing metrics from sklearn	28
6.15	Importing SVR	29
6.16	Using SVR to make predictions	29
6.17	Printing the NumPy Array p	30
6.18	Using the trained regressor SVR model to make a prediction	30
6.19	Importing Ridge	30
6.20	Defining Model	31
6.21	Fitting the model to the training data	31
6.22	Printing the Model	32
6.23	Predicting the Output using Ridge	32

ACRONYMS

S.No.	Acronyms	Full Form
1.	LPG	Liquefied Petroleum Gas
2.	IVRS	Interactive Voice Response System
3.	SRS	Software Requirements Specification
4.	IR	Infrared Radiation
5.	HD	High-Definition
6.	OS	Operating System
7.	API	Application Programming Interface
8.	SVR	Systemic Vascular Resistance
9.	SQL	Structured Query Language

CHAPTER 1

INTRODUCTION

In India, the supply of LPG through pipelines is not possible due to shortage of LPG. As technology being improved many gas agencies or distributors have implemented IVRS these days although due to daily busy schedules, customer finds very difficult to book new cylinder. So, the proposal is to completely automate the process of refill booking without human intervention that accordingly will help consumer against foul play. Based on the usages of LPG cylinder in domestic and number of members stays in the house, data has been recorded based on every time the customer refills the LPG cylinder. Main aim of this Internhship project is to predict the next refilling date of each family.

1.1 Project Objective

The objective of this Internship project is to develop an automated system for LPG cylinder refill bookings in India, which will eliminate the need for human intervention in the refill booking process, streamline the refill process, and improve the customer experience.

Specifically, the objectives of your project are as follows:

- To develop a predictive analytics model that leverages usage data from previous LPG cylinder refills to predict the next refill date for each family.
- To design an automated system that generates refill requests and notifies gas agencies or distributors when a refill is required, eliminating the need for manual bookings.
- To develop a mobile application or web portal that enables customers to view their refill history, upcoming refill dates, and other relevant information.
- To ensure scalability of the system architecture to handle a large number of households and accommodate future growth.
- To make the system user-friendly and accessible, with minimal training required for customers to use it effectively.
- To reduce the chances of foul play by gas agencies or distributors, ensuring timely and hassle-free refills for customers.
- To collaborate with gas agencies or distributors to integrate the system with their existing infrastructure, ensuring a seamless transition and implementation process.

By achieving these objectives, this Internship project aims to significantly improve the

customer experience when booking LPG cylinder refills in India, reduce the burden on gas agencies or distributors, and ensure timely and hassle-free refills for customers.

1.2 Project Scope

The scope of this Internship involves the development of an automated system for booking LPG cylinder refills in India. The system will be based on predictive analytics, which will leverage usage data from previous LPG cylinder refills to predict the next refill date for each family. The system will be designed to eliminate the need for human intervention in the refill booking process, thereby streamlining the process and reducing the burden on customers.

The proposed system will include the following features:

Data analytics: The system will use data analytics techniques to analyze usage data from previous LPG cylinder refills and predict the next refill date for each family.

Automation: The system will be designed to automatically generate refill requests and notify the gas agencies or distributors when a refill is required.

Scalability: The system will be designed to handle a large number of households, and the system architecture will be scalable to accommodate future growth.

User-friendly: The system will be user-friendly, and customers will be able to access the system easily, with minimal training required.

The scope of the project includes the development of the system, testing, and implementation. The system will be scalable and can be implemented across various regions of India. The project will require collaboration with gas agencies or distributors to integrate the system with their existing infrastructure. Overall, the scope of this Internship project is to provide a convenient, automated, and hassle-free way for customers to book LPG cylinder refills in India, thereby improving the customer experience and reducing the burden on distributors.

1.3 Problem Statement

- The existing system is basically a manual system, where if the user needs cylinders then he needs to contact the seller and book them.
- While cooking cylinder gets over.
- Many family use single cylinder so for them it's very difficult to know when cylinder gets over.

CHAPTER 2

ORGANIZATION PROFILE

Anspro Technologies is a Bangalore based software development company with vast industry experience in various domains, services & product lines. They utilize their vast experience and expertise in developing various software solutions in accordance with client's business and job requirements.

Anspro Technologies provides cost-effective, convenient and easy-to-manage software solutions services. They serve clients with cost efficient software applications that helps them to grow their business. Anspro Technologies offers one stop solutions to their customers for their web designing, web application development & web hosting requirements. They provide e-learning applications, which have advanced features and rich graphical interface. Their e-learning applications are best suited for educational institutions, training institutes for providing distant education and training purposes. Their customized billing and accounting tool is suitable for any departmental stores as well as for small scale industries.

They believe that technologies and ideas, are more than anything else to challenge the world and grow. However, it's not only the technologies they use, but how they integrate them, that counts. They understand that to integrate technologies requires the right people and they have specialist individual teams of experts on board who provide pre-sales, post sales, project implementation and support.

In order to sustain the productivity of any organization, it is necessary to automate the biometric time attendance system management. This is because manual attendance punching and related calculation leads to time and cost consumption.

Many of their clients request comprehensive maintenance contracts with Anspro Technologies to ensure that their critical networks (hardware or software) are supported. These contracts range from traditional warranty support, to maintenance during office hours, to 24X7 critical support.

CHAPTER 3

SYSTEM ANALYSIS

System analysis is the process of gathering and interpreting facts, diagnosing problems and using the information to recommend Improvements on the system. System analysis is a problem solving Activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system Development process.

The system is viewed as a whole, the inputs are identified and the system is subjected to close study to identify the Problem areas. The solutions are given as a proposal. The proposal is reviewed on user request and suitable changes are made. This loop ends as soon as the user is satisfied with the proposal.

3.1 Existing System

The existing system for LPG cylinder refill bookings in India involves a manual process, where customers have to call or visit their gas agency or distributor to request a refill. This manual process can be time-consuming and inconvenient for customers, particularly those with busy schedules or mobility issues. In recent years, many gas agencies or distributors have implemented Interactive Voice Response System (IVRS), which enables customers to book a refill by phone. However, due to the high volume of calls and limited availability of customer service representatives, customers often face long wait times and difficulties in accessing the service. Moreover, the current system does not take into account usage data from previous refills, and customers have to manually keep track of when their next refill is due. This lack of automation and data-driven insights can result in delays or missed refills, which can be frustrating for customers. Overall, the existing system for LPG cylinder refill bookings in India is manual, time-consuming, and prone to delays, resulting in a poor customer experience. This project aims to address these issues by developing an automated system that leverages predictive analytics to eliminate the need for manual bookings and improve the customer experience.

The existing system for LPG cylinder refill bookings in India has several limitations, including:

- **Manual process:** The current system involves a manual process where customers have to call or visit their gas agency or distributor to request a refill. This manual process can

be time-consuming and inconvenient for customers, particularly those with busy schedules or mobility issues.

- **Limited availability of customer service representatives:** Due to the high volume of calls and limited availability of customer service representatives, customers often face long wait times and difficulties in accessing the service.
- **Lack of automation:** The current system does not take into account usage data from previous refills, and customers have to manually keep track of when their next refill is due. This lack of automation can result in delays or missed refills, which can be frustrating for customers.
- **Limited transparency:** The current system lacks transparency in the refill process, which can lead to misunderstandings and customer complaints. Customers may not have access to real-time information on their refill status or expected delivery times.
- **Foul play:** The current system is vulnerable to foul play by gas agencies or distributors, such as delayed refills or charging extra fees for expedited services. This can result in a poor customer experience and erode trust in the system.

Overall, the limitations of the existing system for LPG cylinder refill bookings in India highlight the need for an innovative, data-driven solution that can streamline the refill process, improve the customer experience, and reduce the burden on gas agencies or distributors.

3.2 Proposed System

The proposed system for LPG cylinder refill bookings in India is an automated, data-driven system that leverages predictive analytics to eliminate the need for manual bookings and improve the customer experience. The system will use usage data from previous LPG cylinder refills to predict the next refill date for each family, thereby automating the refill booking process and reducing the burden on customers. The system will also generate refill requests and notify gas agencies or distributors when a refill is required, eliminating the need for manual bookings.

The proposed system will be accessible through a dedicated mobile application or web portal, which will enable customers to view their refill history, upcoming refill dates, and other relevant information. The system will also be scalable to handle a large number of households and accommodate future growth. The proposed system aims to reduce the chances of foul play by gas agencies or distributors and ensure timely and hassle-free refills for customers. The system will be designed to be user-friendly, with minimal training required for customers to use it effectively. Overall, the proposed system for LPG cylinder refill bookings in India is an

innovative, data-driven solution that has the potential to significantly improve the customer experience, streamline the refill process, and reduce the burden on gas agencies or distributors. The proposed system for LPG cylinder refill bookings in India offers several advantages over the existing system, including:

- **Automated refill booking process:** The proposed system automates the refill booking process, eliminating the need for manual bookings and reducing the burden on customers. The system uses usage data from previous refills to predict the next refill date, generates refill requests, and notifies gas agencies or distributors when a refill is required.
- **Improved customer experience:** The proposed system improves the customer experience by eliminating long wait times and delays associated with manual bookings. Customers can view their refill history, upcoming refill dates, and other relevant information through a dedicated mobile application or web portal.
- **Increased transparency:** The proposed system offers increased transparency in the refill process, providing real-time information on refill status and expected delivery times. This can help to reduce misunderstandings and customer complaints.
- **Reduced chances of foul play:** The proposed system reduces the chances of foul play by gas agencies or distributors, as the automated process eliminates the need for human intervention in the refill booking process.
- **Scalable and efficient:** The proposed system is scalable and efficient, capable of handling a large number of households and accommodating future growth. The system is designed to be user-friendly, with minimal training required for customers to use it effectively.

Overall, the proposed system for LPG cylinder refill bookings in India is an innovative, data-driven solution that has the potential to significantly improve the customer experience, streamline the refill process, and reduce the burden on gas agencies or distributors.

3.3 System Requirement Specification

System Requirement Specification (SRS) is a focal report, which outlines the foundation of the item headway handle. It records the necessities of a structure and in addition has a delineation of its noteworthy highlight. A SRS is basically an affiliation's seeing (in making) of a customer or potential client's edge work necessities and conditions at a particular point in time before any veritable design or change work. It's a two-way insurance approach that ensures that both the client and the affiliation understand exchange's necessities from that perspective at a given point in time. The sythesis of programming need detail reduces headway effort, as careful

review of the report can reveal oversights, mixed up presumptions, and inconsistencies in front of plan for the change cycle when these issues are less requesting to right.

Hardware requirements

Processor	:	Intel i3
RAM	:	4GB
Hard Disk	:	160GB

Software requirements

Operating System	:	Windows 10
Language	:	Python
Tool	:	Jupyter Notebook

CHAPTER 4

SYSTEM DESIGN

System design focuses on the detailed implementation of the feasible system over the existing system. It emphasizes on translating the design specifications to performance specification. System design has two phases of development.

- Logical design
- Physical design

During logical design phase the analyst describes inputs, outputs and procedures all in a format that meets the user requirements. The analyst also specifies the needs of the user at a level that virtually determines the information flow in and out of the system and the data resources.

The physical design is followed by physical design or coding. Physical design produces the working system by defining the design specifications, which specify exactly what the candidate system must do. The programmers write the necessary programs that accept input from the user, perform necessary processing on accepted data and produce the required report on a hard copy or display it on the screen.

4.1 System Tools

The various system tools that have been used in this project are discussed below:

4.1.1 Python

Python is an object-oriented programming language created by Guido Rossum in 1989. It is ideally designed for rapid prototyping of complex applications. It has interfaces to many OS system calls and libraries and is extensible to C or C++. Many large companies use the Python programming language include NASA, Google, YouTube, Bit Torrent, etc.

Python is widely used in Artificial Intelligence, Natural Language Generation, Neural Networks and other advanced fields of Computer Science. Python has deep focus on code readability & this class will teach you python from basics.

Here python language is used for IR sensor and HD camera

Characteristics of Python

- It provides rich data types and easier to read syntax than any other programming languages.
- It is a platform independent scripted language with full access to operating system API's
- Compared to other programming languages, it allows more run-time flexibility.
- It includes the basic text manipulation facilities of Perl and Awk.
- A module in Python may have one or more classes and free functions.
- Libraries in Python are cross-platform compatible with Linux, MacIntosh, and Windows.

4.1.2 Python 3.7 IDE

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, <https://www.python.org/>, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

4.1.3 Jupyter Notebook

The Jupyter Notebook is an unbelievably incredible asset for intelligently creating and introducing information science ventures. A setup of Jupyter Notebooks on machines and how to use for data science projects are the basic requirements. A cell coordinates code & its yield into a solitary archive that consolidates representations, story text, numerical conditions and other media. This natural work process advances iterative fast turn of events, settling on note pad an inexorably well-known decision at the core of contemporary data science, examination and progressively science at large. As a component of the open source Project Jupyter, they are totally free.

The Jupyter venture is the replacement to the prior I Python Notebook, which was first distributed as a model in 2010. Despite the fact that it is conceivable to utilize various

programming dialects inside Jupyter Notebooks which concentrates on Python as it is the most well-known use case.

4.2 Architecture Diagram

The system begins by taking in LPG data as input, which is then preprocessed to prepare it for feature selection. The feature selection step is performed to select the most relevant features for the SVR and Ridge algorithms. The SVR and Ridge algorithms are then applied to the selected features to generate refill requests based on usage patterns. The model is evaluated to ensure that it performs accurately. Finally, the refill requests are automatically sent as notifications to the relevant gas agencies. Here fig. 4.1 is Architecture diagram presented.

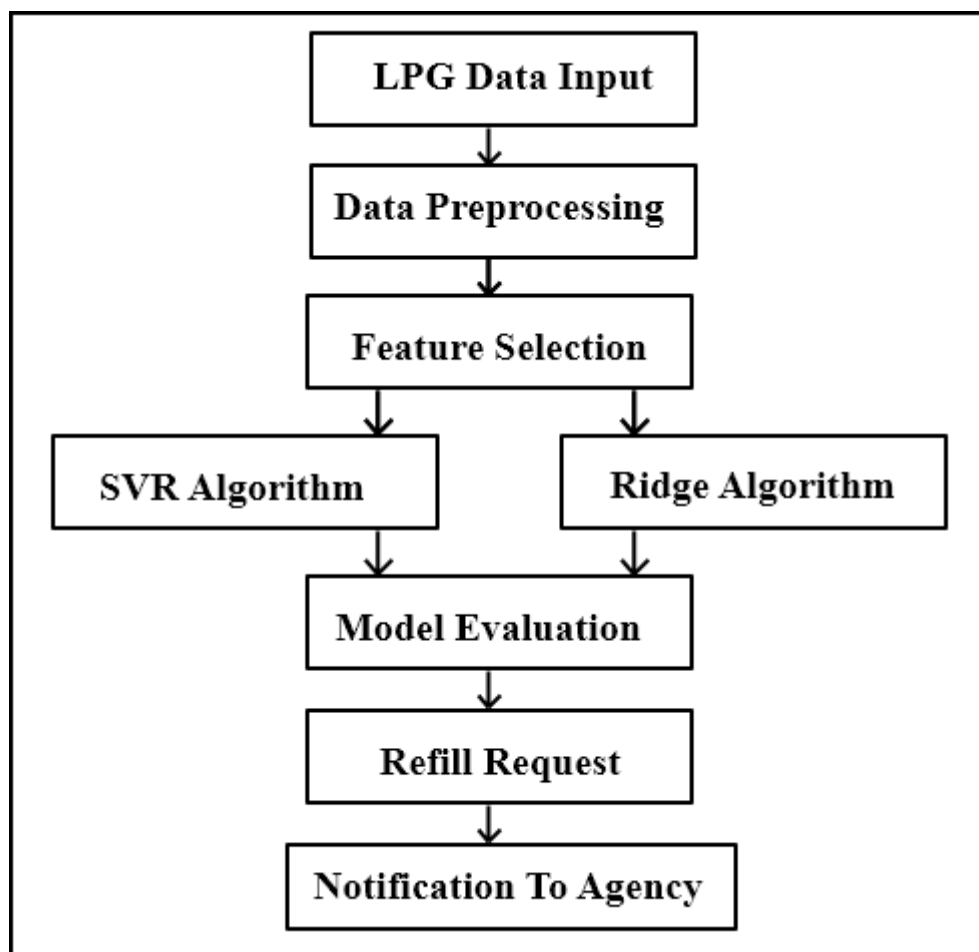


Fig 4.2 Architecture diagram

4.3 Implementation

Implementation is done by using the algorithms discussed below:

4.3.1 Ridge Algorithm

Ridge algorithm to predict the number of days it takes for a customer to refill their LPG gas cylinder based on their family size, usage history, and refill history. Ridge regression is a regularized version of linear regression that introduces a regularization term to the cost function. This term penalizes the magnitude of the coefficients of the linear regression model, forcing them to be smaller.

The effect of this regularization term is to prevent overfitting, which can occur when the model becomes too complex and starts to fit the noise in the data rather than the underlying patterns. Ridge algorithm helped to reduce the complexity of the model and improve its performance by minimizing the impact of small changes in the input variables that may not be relevant to the prediction of the output variable. Overall, Ridge regression is a useful algorithm for data analysis when dealing with high-dimensional datasets where overfitting is a concern. By introducing regularization, it is possible to create a more robust model that is better suited for prediction tasks where small changes in the input variables can result in significant changes in the output variable.

4.3.2 SVR Algorithm

Support Vector Regression (SVR) algorithm is used to predict the number of days it takes for a customer to refill their LPG gas cylinder based on their family size, usage history, and refill history. SVR is a type of machine learning algorithm that is used for regression tasks, which means it is used to predict a continuous output variable. Unlike linear regression or Ridge regression, which try to fit a linear function to the data, SVR uses a non-linear function to fit the data. SVR works by mapping the input data to a higher-dimensional feature space, where it becomes easier to find a non-linear boundary that separates the data into different classes. The goal of SVR is to find a hyperplane in this feature space that maximizes the margin between the hyperplane and the closest data points. the SVR algorithm was able to capture the non-linear relationships between the input variables and the output variable, leading to a better prediction performance compared to linear regression and Ridge regression. Overall, SVR is a powerful and flexible algorithm that is widely used in machine learning for regression tasks. It is particularly useful when dealing with non-linear relationships between the input variables and output variable, which is often the case in real-world problems.

CHAPTER 5

CONCEPTS OF PYTHON

5.1 Pandas

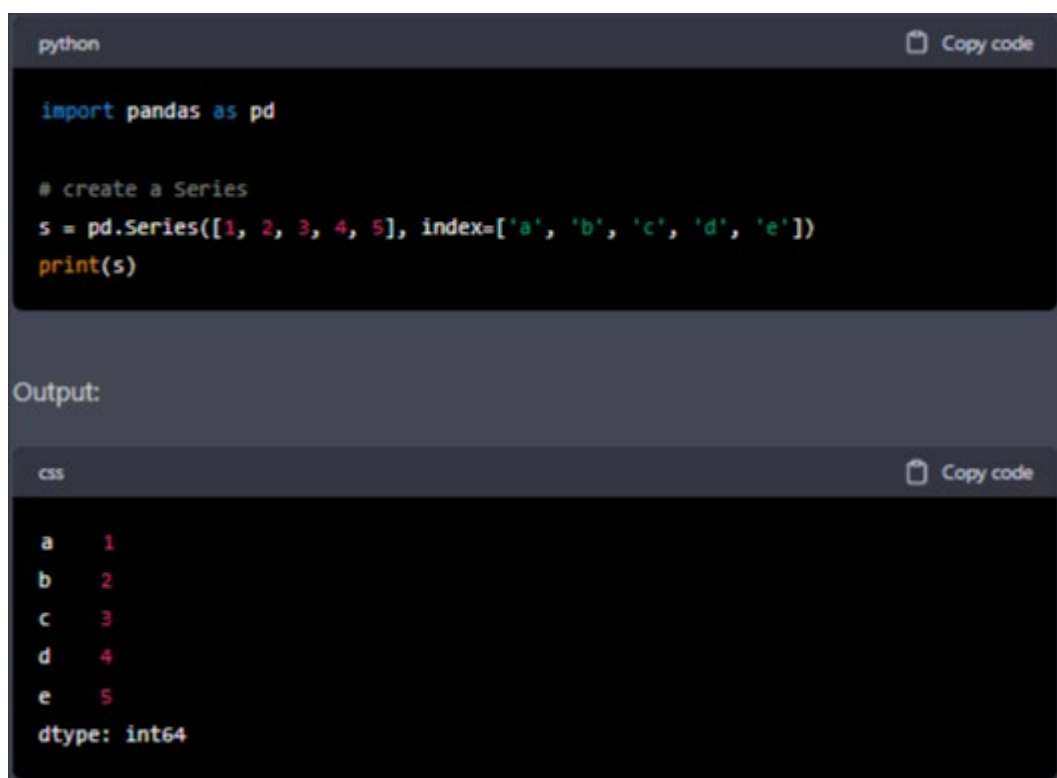
Pandas is a popular open-source library in Python used for data manipulation and analysis. It is built on top of NumPy and provides a fast, flexible, and easy-to-use interface for data handling. Pandas is particularly useful for working with structured or tabular data, such as data in CSV, Excel, or SQL databases. In this overview, we'll cover some of the most important concepts and functions of pandas.

Data Structures in Pandas

Pandas has two main data structures: Series and DataFrame.

- **Series**

A Series is a one-dimensional labeled array capable of holding any data type. It is similar to a NumPy array, but with an index that can be of any type. Fig 5.1 shows Series Data Structure using Pandas .Here's an example of creating a Series:



The screenshot shows a Jupyter Notebook interface. The top part is a code editor with the following Python code:

```
python
import pandas as pd

# create a Series
s = pd.Series([1, 2, 3, 4, 5], index=['a', 'b', 'c', 'd', 'e'])
print(s)
```

Below the code editor is an 'Output' section. It displays the result of the code execution as a table:

a	1
b	2
c	3
d	4
e	5

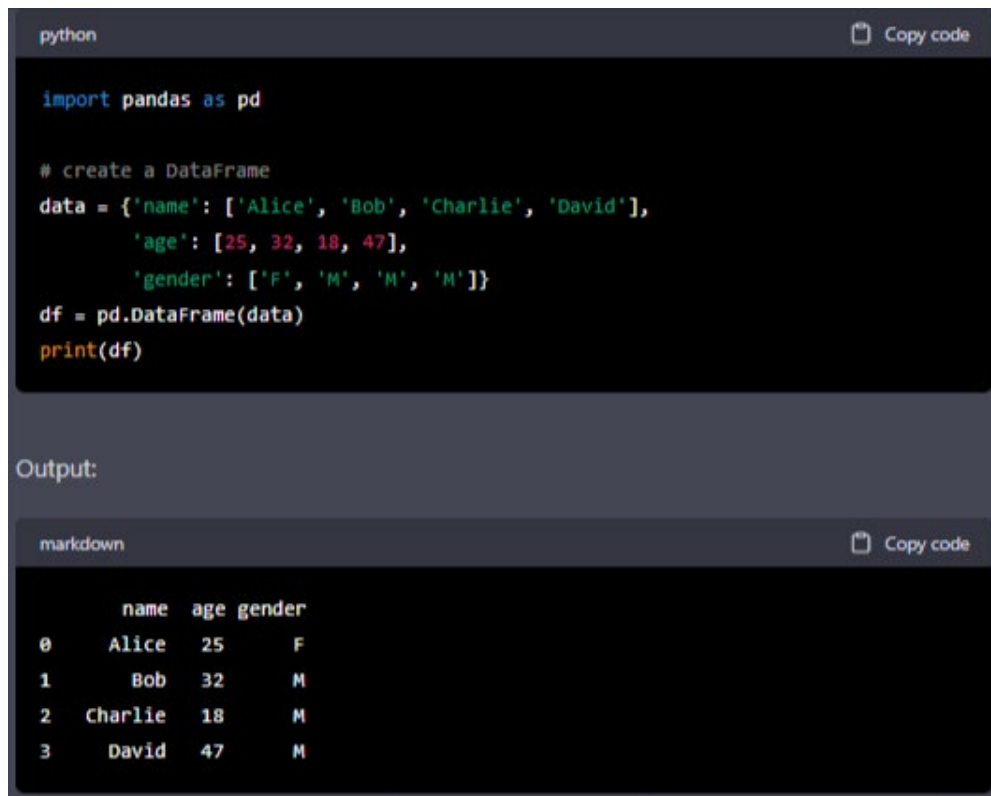
At the bottom of the output, it shows 'dtype: int64'.

Fig 5.1 Series Data Structure using Pandas

- **DataFrame**

A DataFrame is a two-dimensional labeled data structure with columns of potentially different

types. It is like a spreadsheet or SQL table. Fig 5.1 shows Series Data Structure using Pandas. Here's an example of creating a DataFrame:



```
python
import pandas as pd

# create a DataFrame
data = {'name': ['Alice', 'Bob', 'Charlie', 'David'],
        'age': [25, 32, 18, 47],
        'gender': ['F', 'M', 'M', 'M']}
df = pd.DataFrame(data)
print(df)
```

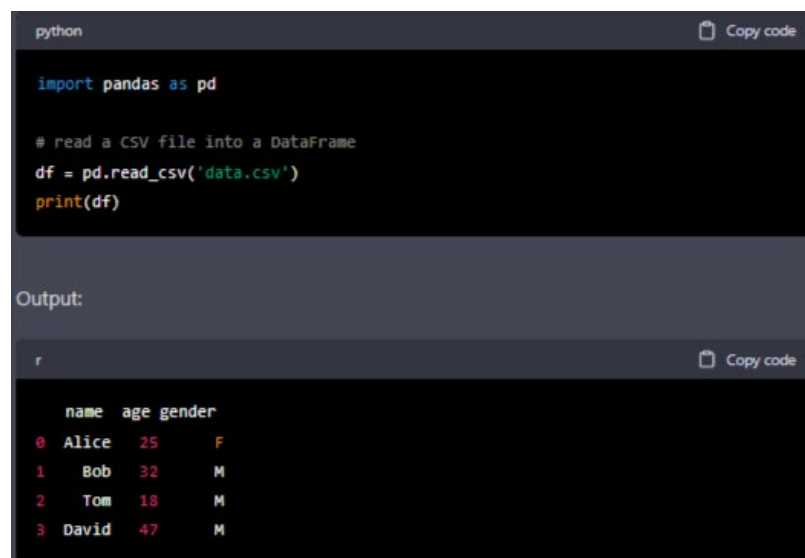
Output:

```
markdown
name age gender
0  Alice  25    F
1   Bob   32    M
2  Charlie 18    M
3   David 47    M
```

Fig 5.1 Series Data Structure using Pandas

Reading and Writing Data

Pandas provides functions to read and write data in various file formats, such as CSV, Excel, SQL databases, and more. Fig 5.1 shows Reading and Writing Data using Pandas. Here's an example of reading a CSV file into a DataFrame:



```
python
import pandas as pd

# read a CSV file into a DataFrame
df = pd.read_csv('data.csv')
print(df)
```

Output:

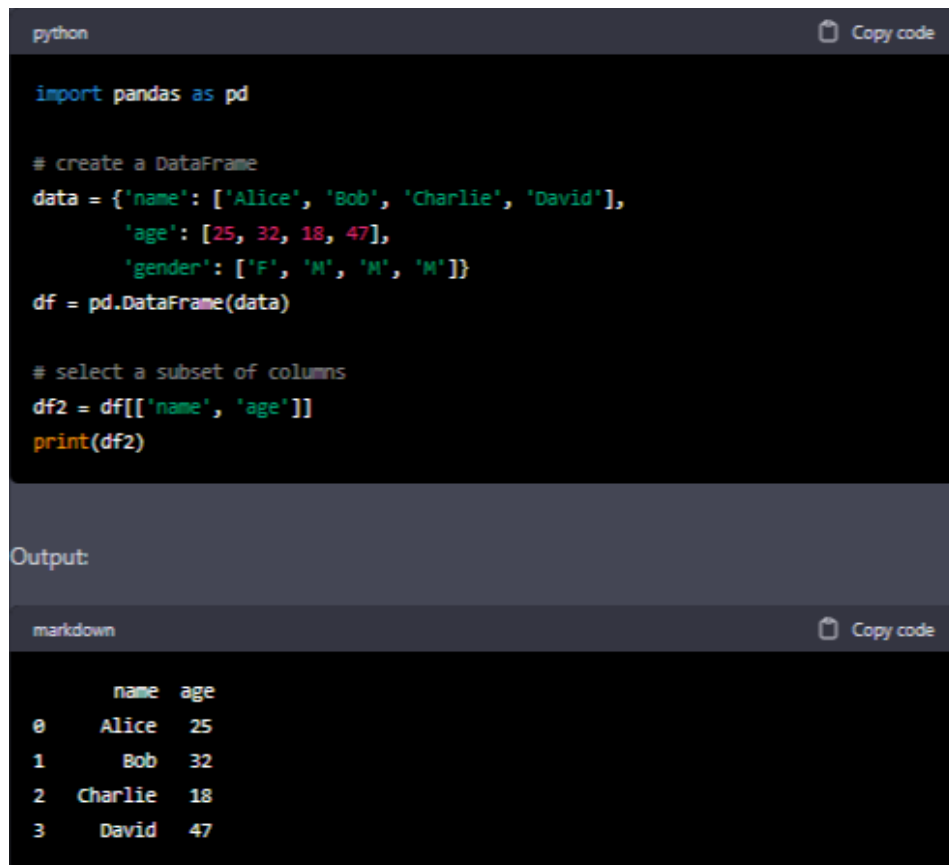
```
r
name age gender
0  Alice  25    F
1   Bob   32    M
2   Tom   18    M
3  David  47    M
```

Fig 5.1 Reading and Writing Data using Pandas

Data Selection and Filtering

Pandas provides several ways to select and filter data based on various conditions. Fig 5.1

shows Data Selection and Filtering. Here's an example of selecting a subset of columns from a DataFrame:



```
python
import pandas as pd

# create a DataFrame
data = {'name': ['Alice', 'Bob', 'Charlie', 'David'],
        'age': [25, 32, 18, 47],
        'gender': ['F', 'M', 'M', 'M']}
df = pd.DataFrame(data)

# select a subset of columns
df2 = df[['name', 'age']]
print(df2)
```

Output:

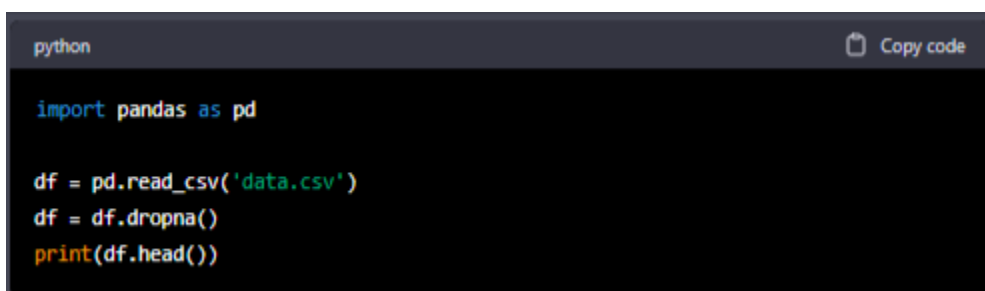
```
markdown
    name  age
0  Alice   25
1   Bob   32
2 Charlie   18
3  David   47
```

Fig 5.1 Data Selection and Filtering

Data Cleaning and Transformation:

Pandas provides a wide range of functions and methods to clean and transform data, such as removing missing values, filling missing values, filtering, sorting, and grouping data.

To remove rows with missing data, you can use the `dropna()` function. Fig 5.1 shows Data Cleaning and Transformation.



```
python
import pandas as pd

df = pd.read_csv('data.csv')
df = df.dropna()
print(df.head())
```

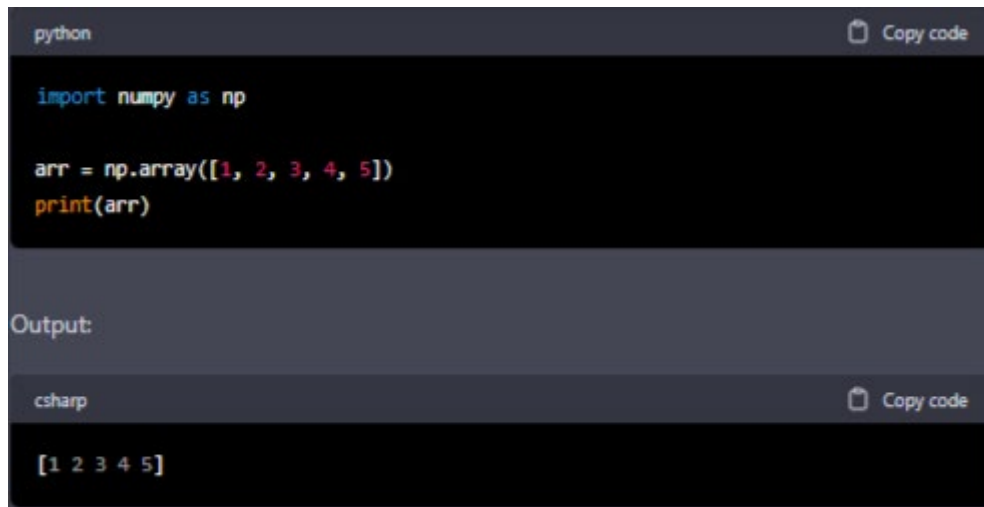
Fig 5.1 Data Cleaning and Transformation

5.2 Numpy

NumPy is a Python library for scientific computing that provides powerful tools for working with multi-dimensional arrays and matrices. It is one of the fundamental packages for scientific computing in Python.

Creating Arrays:

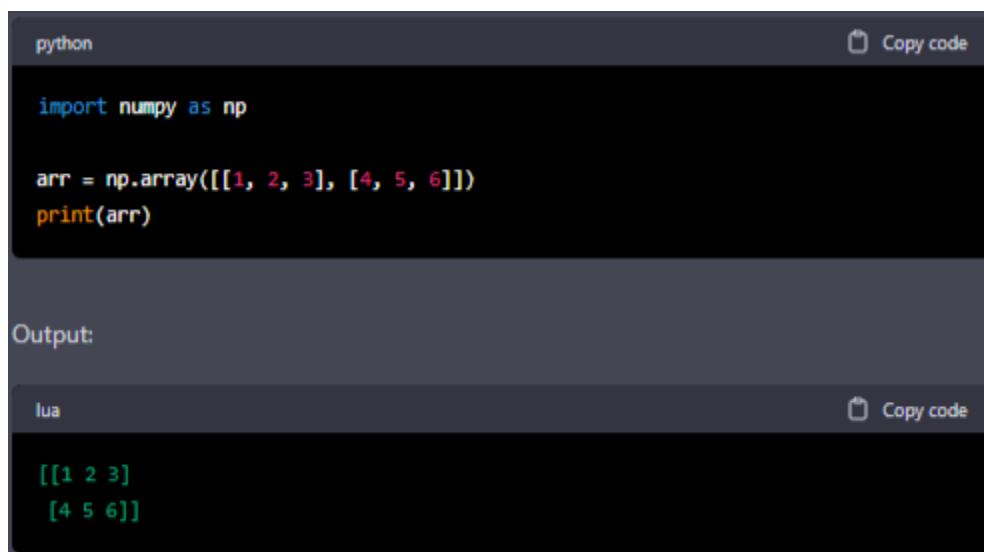
To create a NumPy array, the `array()` function is used. Fig 5.2 shows Creating 2D Arrays Using NumPy.



```
python Copy code  
  
import numpy as np  
  
arr = np.array([1, 2, 3, 4, 5])  
print(arr)  
  
Output:  
  
csharp Copy code  
  
[1 2 3 4 5]
```

Fig 5.2 Creating Arrays Using NumPy

A two-dimensional array can also be created by passing a list of lists.



```
python Copy code  
  
import numpy as np  
  
arr = np.array([[1, 2, 3], [4, 5, 6]])  
print(arr)  
  
Output:  
  
lua Copy code  
  
[[1 2 3]  
 [4 5 6]]
```

Fig 5.2 Creating 2D Arrays Using NumPy

Array Operations:

NumPy provides a wide range of operations that can be performed on arrays, such as element-wise addition, multiplication, and more. Fig 5.2 shows Array Operations Using NumPy.

```
python Copy code

import numpy as np

arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([6, 7, 8, 9, 10])

# Element-wise addition
print(arr1 + arr2)

# Element-wise multiplication
print(arr1 * arr2)

Output:

css Copy code

[ 7  9 11 13 15]
[ 6 14 24 36 50]
```

Fig 5.2 Array Operations Using NumPy

Indexing and Slicing:

Individual elements of a NumPy array can be accessed using indexing. Fig 5.2 uses Indexing and Slicing Using NumPy.

```
python Copy code

import numpy as np

arr = np.array([1, 2, 3, 4, 5])

# Access the second element
print(arr[1])

# Access the last element
print(arr[-1])

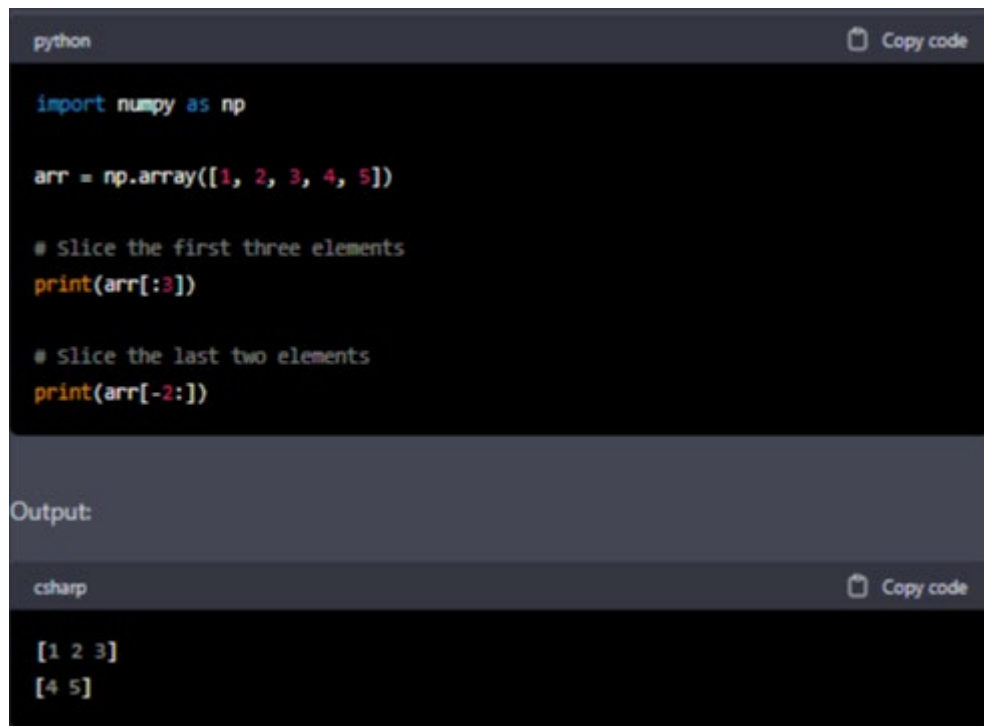
Output:

Copy code

2
5
```

Fig 5.2 Indexing and Slicing Using NumPy

An array can also be sliced to get a range of elements. Fig 5.2 shows Indexing and Slicing an Array Using NumPy.



The screenshot shows a Jupyter Notebook with two cells. The top cell is a Python code cell with the following code:

```
python
import numpy as np

arr = np.array([1, 2, 3, 4, 5])

# Slice the first three elements
print(arr[:3])

# Slice the last two elements
print(arr[-2:])
```

The bottom cell is an output cell showing the results of the code:

```
Output:
[1 2 3]
[4 5]
```

Fig 5.2 Indexing and Slicing an Array Using NumPy

Broadcasting:

NumPy arrays can be broadcasted to perform element-wise operations on arrays with different shapes. Fig 5.2 shows Broadcasting using NumPy.



The screenshot shows a Jupyter Notebook with two cells. The top cell is a Python code cell with the following code:

```
python
import numpy as np

arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([1, 2, 3])

# Broadcasting arr2 to arr1 shape
print(arr1 + arr2)
```

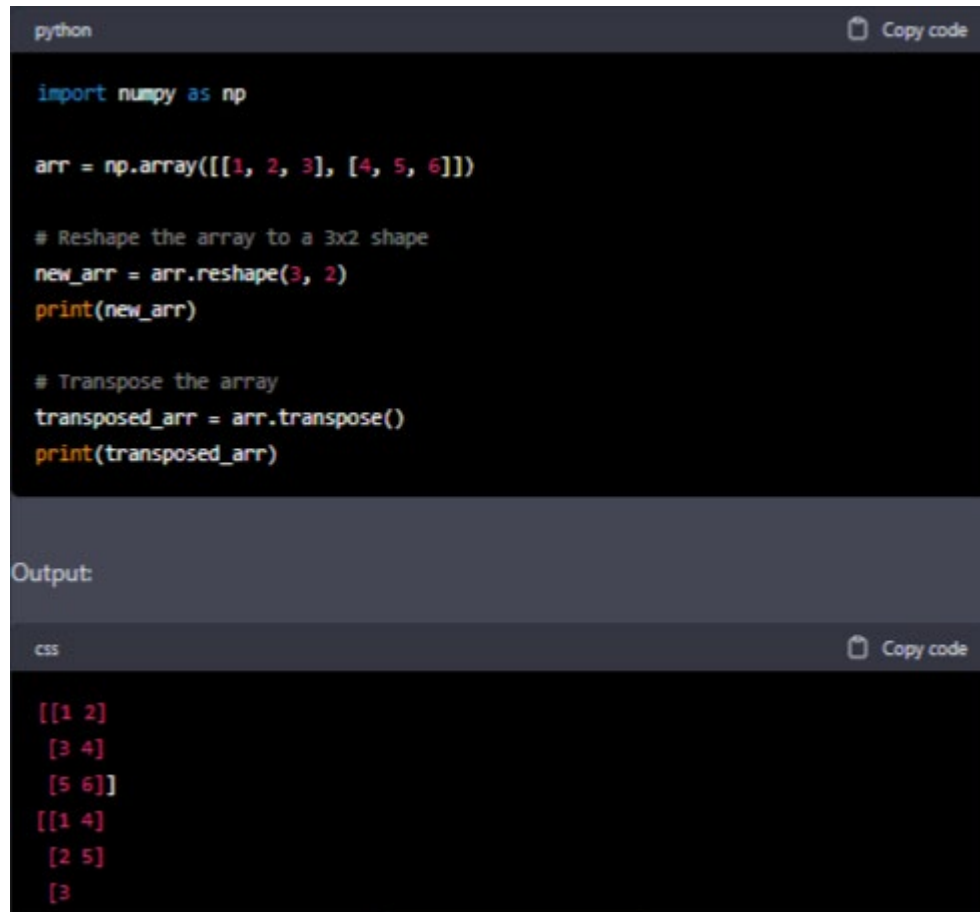
The bottom cell is an output cell showing the result of the code:

```
Output:
[2 4 6 6 8]
```

Fig 5.2 Broadcasting using NumPy

Shape Manipulation:

NumPy provides functions to manipulate the shape of an array, such as `reshape()` to change the shape of an array, and `transpose()` to switch the dimensions of an array. Fig 5.2 shows Shape Manipulation using NumPy.



```
python
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

# Reshape the array to a 3x2 shape
new_arr = arr.reshape(3, 2)
print(new_arr)

# Transpose the array
transposed_arr = arr.transpose()
print(transposed_arr)
```

Output:

```
css
[[1 2]
 [3 4]
 [5 6]]
[[1 4]
 [2 5]
 [3 6]]
```

Fig 5.2 Shape Manipulation using NumPy

5.3 Statsmodels

Statsmodels is a Python library that provides tools for statistical analysis, including regression analysis, time series analysis, and more. It is built on top of NumPy, SciPy, and Pandas, and provides a wide range of statistical models and tests.

Linear Regression:

Statsmodels provides a wide range of tools for linear regression analysis. The `OLS()` function is used to fit a linear regression model, and then use the `summary()` function to get a summary of the model. Fig 5.3 shows Linear Regression Using Statsmodels

```
python
Copy code

import statsmodels.api as sm
import numpy as np

# Generate some random data
np.random.seed(123)
x = np.random.rand(100)
y = 2 * x + 0.5 * np.random.rand(100)

# Fit a linear regression model
x = sm.add_constant(x) # Add constant term to the model
model = sm.OLS(y, x).fit()

# Print the model summary
print(model.summary())
```

Output:

```
markdown
Copy code

OLS Regression Results

=====
Dep. Variable:          y      R-squared:          0.790
Model:                  OLS    Adj. R-squared:      0.788
Method:                 Least Squares    F-statistic:        324.9
Date:                   Mon, 22 Mar 2023    Prob (F-statistic):  5.57e-31
Time:                   15:44:03    Log-Likelihood:     -16.817
No. Observations:       100    AIC:                37.63
Df Residuals:           98    BIC:                43.95
Df Model:                1
Covariance Type:        nonrobust
=====
               coef    std err          t      P>|t|      [0.025    0.975]
-----
const         0.4954     0.053      9.283     0.000     0.389     0.602
x1            1.9725     0.109     18.011     0.000     1.757     2.188
=====
Omnibus:            0.294    Durbin-Watson:      1.893
Prob(Omnibus):      0.863    Jarque-Bera (JB):    0.135
Skew:               -0.073    Prob(JB):            0.935
Kurtosis:           3.067    Cond. No.            2.58
=====

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
```

Fig 5.3 Linear Regression Using Statsmodels

Time Series Analysis:

Statsmodels also provides tools for time series analysis. ARIMA() function is used to fit an ARIMA model, and then use the summary() function to get a summary of the model.

Fig 5.3 shows Time Series Analysis Using Statsmodels.

```
python Copy code  
  
import statsmodels.api as sm  
import pandas as pd  
  
# Load the AirPassengers dataset  
data = sm.datasets.get_rdataset("AirPassengers")  
df = pd.DataFrame(data.data)  
  
# Convert the date column to a datetime object  
df["date"] = pd.to_datetime(df["date"])  
  
# Set the date column as the index  
df.set_index("date", inplace=True)  
  
# Fit an ARIMA model  
model = sm.tsa.ARIMA(df["value"], order=(2, 1, 2)).fit()  
  
# Print the model summary  
print
```

Fig 5.3 Time Series Analysis Using Statsmodels

Hypothesis Testing:

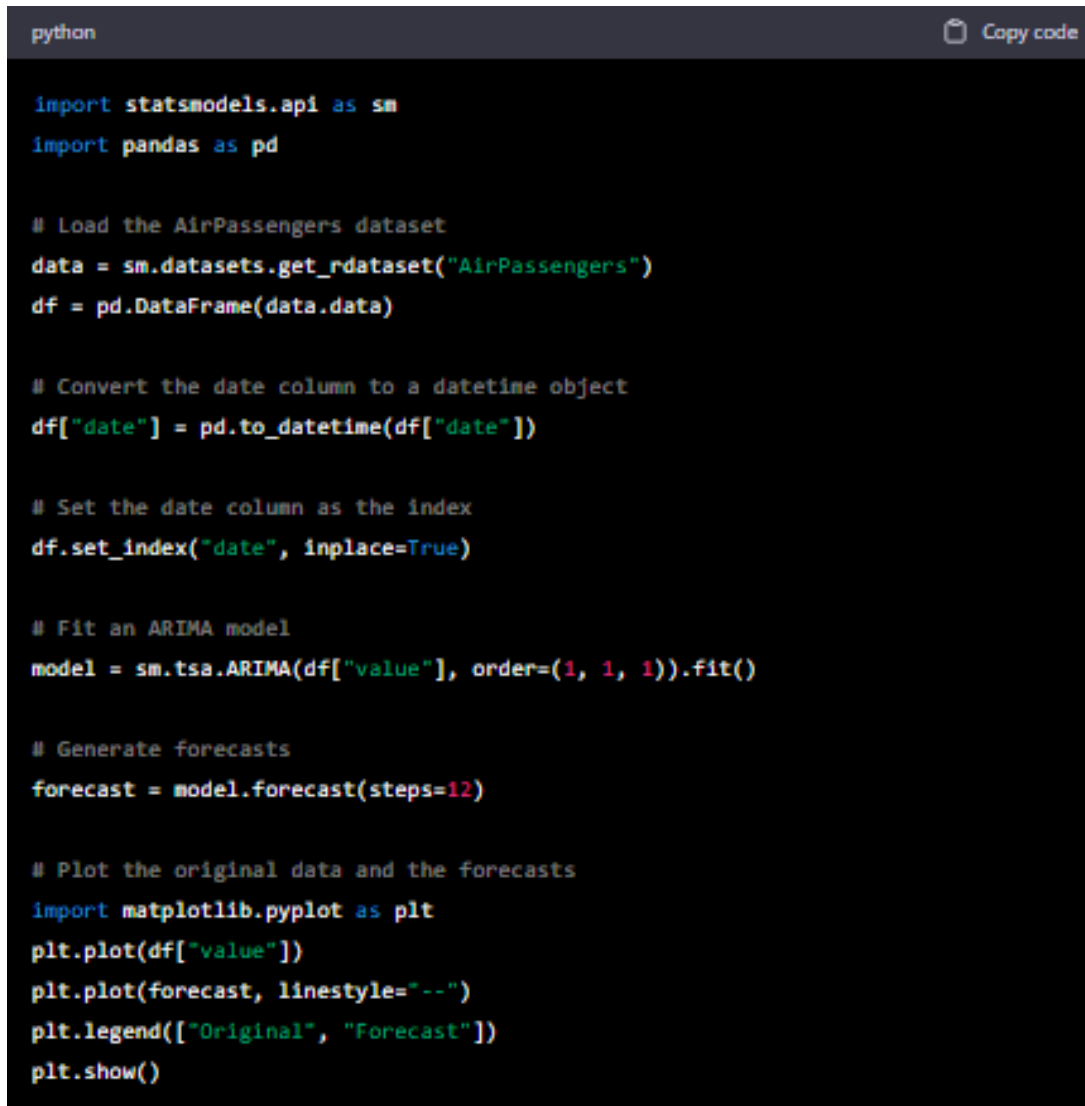
Statsmodels also provides tools for hypothesis testing. `ttest_ind()` function is used to perform a two-sample t-test, and the `ztest()` function to perform a one-sample z-test. Fig 5.3 Shows Hypothesis Testing Using Statsmodels below.

```
python Copy code  
  
import statsmodels.api as sm  
import numpy as np  
  
# Generate two random samples  
np.random.seed(123)  
sample1 = np.random.normal(loc=0, scale=1, size=100)  
sample2 = np.random.normal(loc=1, scale=1, size=100)  
  
# Perform a two-sample t-test  
t_stat, p_value, df = sm.stats.ttest_ind(sample1, sample2)  
print("t-statistic: ", t_stat)  
print("p-value: ", p_value)  
  
# Perform a one-sample z-test  
z_stat, p_value = sm.stats.ztest(sample1, value=0)  
print("z-statistic: ", z_stat)  
print("p-value: ", p_value)
```

Fig 5.3 Hypothesis Testing Using Statsmodels

Time Series Forecasting:

Statsmodels provides tools for time series forecasting. ARIMA() function can be used to fit an ARIMA model, and then use the forecast() function to generate forecasts. Here Fig 5.3 below shows Time Series Forecasting Using Statsmodels.



```
python Copy code  
  
import statsmodels.api as sm  
import pandas as pd  
  
# Load the AirPassengers dataset  
data = sm.datasets.get_rdataset("AirPassengers")  
df = pd.DataFrame(data.data)  
  
# Convert the date column to a datetime object  
df["date"] = pd.to_datetime(df["date"])  
  
# Set the date column as the index  
df.set_index("date", inplace=True)  
  
# Fit an ARIMA model  
model = sm.tsa.ARIMA(df["value"], order=(1, 1, 1)).fit()  
  
# Generate forecasts  
forecast = model.forecast(steps=12)  
  
# Plot the original data and the forecasts  
import matplotlib.pyplot as plt  
plt.plot(df["value"])  
plt.plot(forecast, linestyle="--")  
plt.legend(["Original", "Forecast"])  
plt.show()
```

Fig 5.3 Time Series Forecasting Using Statsmodels

Generalized Linear Models:

Statsmodels also provides tools for fitting generalized linear models (GLMs), which are a generalization of linear regression that can handle non-normal response variables. GLM() function is used to fit a GLM. Here Generalized Linear Models Using Statsmodels is shown in Fig 5.3 below.

```
python Copy code  
  
import statsmodels.api as sm  
import pandas as pd  
  
# Load the diabetes dataset  
data = sm.datasets.get_rdataset("diabetes")  
df = pd.DataFrame(data.data, columns=data.feature_names)  
df["target"] = data.target  
  
# Fit a GLM  
X = df[["age", "bmi", "bp"]]  
y = df["target"]  
model = sm.GLM(y, sm.add_constant(X), family=sm.families.Gaussian()).fit()  
  
# Print the model summary  
print(model.summary())
```

Fig 5.3 Generalized Linear Models Using Statsmodels

CHAPTER 6

RESULTS

```
#.....LPG Training.....  
  
import os  
import time  
import joblib  
import warnings  
import numpy as np  
import pandas as pd  
from sklearn import metrics  
from sklearn.metrics import accuracy_score  
from sklearn.metrics import confusion_matrix  
from sklearn.metrics import classification_report  
warnings.filterwarnings(action = 'ignore')  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.model_selection import train_test_split  
import matplotlib.pyplot as plt  
import seaborn as sns
```

Fig 6.1 Importing the modules

This code imports several Python libraries that will be used in this project:

os: provides a way to interact with the operating system by accessing files, directories, etc.

time: provides various functions to work with time, such as measuring time intervals and sleeping.

joblib: provides tools for saving and loading data to/from disk, particularly useful for saving the results of a trained model.

warnings: provides a way to handle warnings that occur during program execution.

numpy: a numerical computing library for Python that provides tools for working with arrays, linear algebra, random number generation, etc.

pandas: a library for data manipulation and analysis.

sklearn: a popular machine learning library for Python that provides a range of tools for data preprocessing, model selection, and evaluation.

metrics: a sub-library of scikit-learn that provides a range of metrics for evaluating machine learning models.

seaborn: a library for data visualization based on matplotlib.

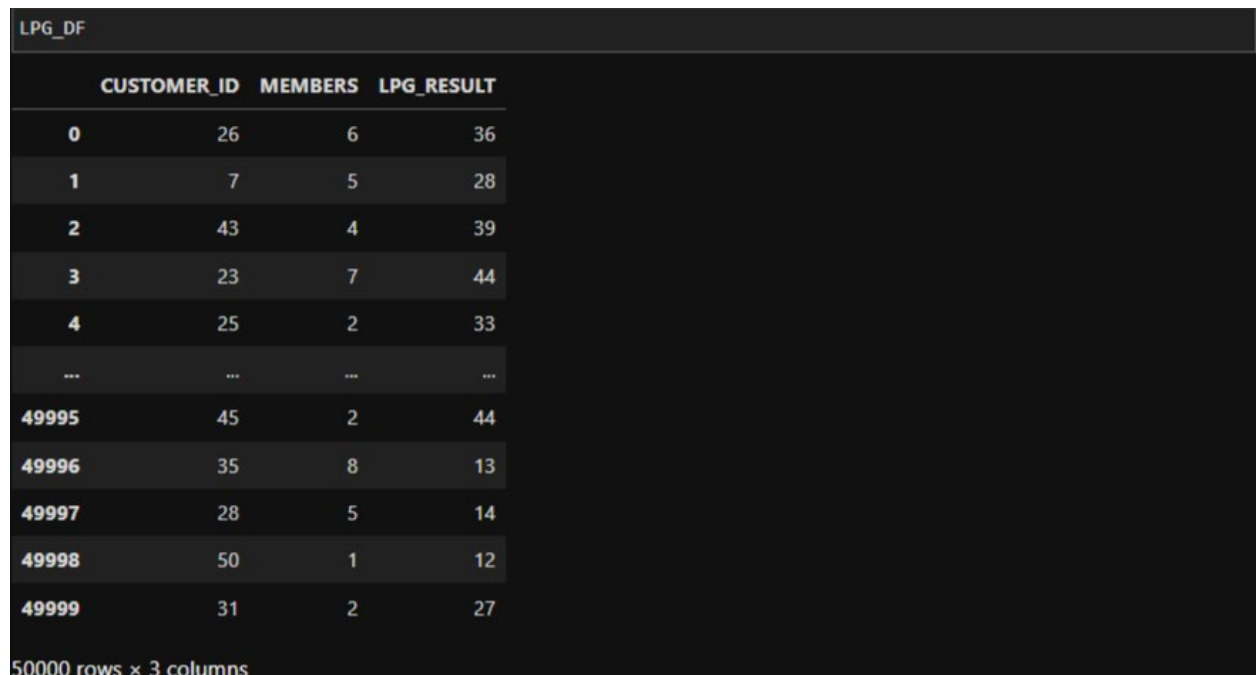
By importing these libraries, project will have access to a variety of tools and functionality that will be used to build and evaluate machine learning models.

```
LPG_DF = pd.read_csv("../LPG_Data1.csv") #Reads the csv file
```

Fig 6.2 Reading the Dataset

This line of code reads in a CSV file named "LPG_Data1.csv" located in the current working directory (".") using the Pandas `read_csv` function. The CSV file is assumed to contain data about LPG (liquefied petroleum gas), and the data is stored in a Pandas DataFrame called `LPG_DF`.

The `read_csv` function is a built-in function in Pandas that allows you to read in CSV files and convert them to a DataFrame, which is a two-dimensional table-like data structure with rows and columns. The function can handle various input formats, such as local files, remote URLs, or data stored in memory.



	CUSTOMER_ID	MEMBERS	LPG_RESULT
0	26	6	36
1	7	5	28
2	43	4	39
3	23	7	44
4	25	2	33
...
49995	45	2	44
49996	35	8	13
49997	28	5	14
49998	50	1	12
49999	31	2	27

50000 rows x 3 columns

Fig 6.3 Reading the Dataset

`LPG_DF` is a Pandas DataFrame that contains the data read from the CSV file "LPG_Data1.csv". This DataFrame likely has rows and columns representing various attributes of LPG, such as the type of LPG, the price, the location of the gas station, etc. The specific structure and content of the DataFrame will depend on the contents of the CSV file. Printing out `LPG_DF` will display the contents of the DataFrame in a tabular format. The output will show the first few rows of the DataFrame, along with the column names and data types of each column.

```
print(type(LPG_DF))
<class 'pandas.core.frame.DataFrame'>
```

Fig 6.4 Checking the Dimensions of Dataset

`LPG_DF.shape` is a Pandas DataFrame attribute that returns the dimensions of the DataFrame, i.e., the number of rows and columns in the DataFrame. It returns a tuple in the format (number of rows, number of columns). For example, if `LPG_DF` has 50000 rows and 3 columns, then `LPG_DF.shape` would return (50000,3).


```
LPG_DF.info() #The info() method prints information about the DataFrame.
#The information contains the number of columns, column labels, column data types, memory usage, range index,
#and the number of cells in each column (non-null values).

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 3 columns):
#   Column      Non-Null Count  Dtype
---  -
0   CUSTOMER_ID  50000 non-null   int64
1   MEMBERS      50000 non-null   int64
2   LPG_RESULT   50000 non-null   int64
dtypes: int64(3)
memory usage: 1.1 MB
```

Fig 6.5 Summary of the Dataset

LPG_DF.info() is a Pandas DataFrame method that displays a summary of the DataFrame's structure and content, including:

- The number of non-null values in each column
- The data type of each column
- The memory usage of the DataFrame
- Some basic statistics for numeric columns, such as the mean, standard deviation, minimum, and maximum values.

```
LPG_DF.describe() #The describe() method returns description of the data in the DataFrame.
#If the DataFrame contains numerical data, the description contains these information for each column:
#count - The number of not-empty values. mean - The average (mean) value. std - The standard deviation.
```

	CUSTOMER_ID	MEMBERS	LPG_RESULT
count	50000.000000	50000.000000	50000.000000
mean	25.461380	4.490340	28.549600
std	14.393438	2.297453	9.814678
min	1.000000	1.000000	12.000000
25%	13.000000	2.000000	20.000000
50%	25.000000	4.000000	29.000000
75%	38.000000	7.000000	37.000000
max	50.000000	8.000000	45.000000

Fig 6.6 Summary of the basic statistical properties

LPG_DF.describe() is a Pandas DataFrame method that provides a summary of the basic statistical properties of the numeric columns in the DataFrame. For each numeric column, describe() computes the following statistics:

- count: the number of non-null values in the column
- mean: the average value of the column
- std: the standard deviation of the column
- min: the minimum value in the column
- 25%: the 25th percentile value of the column

- 50%: the median (50th percentile) value of the column
- 75%: the 75th percentile value of the column
- max: the maximum value in the column

```
LPG_DF.tail(180) #This function returns last n rows from the object based on position.
#It is useful for quickly verifying data, for example, after sorting or appending rows.
```

CUSTOMER_ID	MEMBERS	LPG_RESULT
49820	37	4
49821	48	1
49822	24	8
49823	28	5
49824	16	7
...
49995	45	2
49996	35	8
49997	28	5

Fig 6.7 Last 180 rows and columns of Dataset

LPG_DF.tail(180) is a Pandas DataFrame method that returns the last n rows of the DataFrame, where n is the number passed as an argument to tail(). In this case, tail(180) will return the last 180 rows of the LPG_DF DataFrame.

This method is useful for quickly inspecting the end of the DataFrame and checking for any issues that may have arisen towards the end of the data collection period.

```
LPG_DF.head(180) #This function returns the first n rows for the object based on position.
#It is useful for quickly testing if your object has the right type of data in it
```

	CUSTOMER_ID	MEMBERS	LPG_RESULT
0	26	6	36
1	7	5	28
2	43	4	39
3	23	7	44
4	25	2	33
...
175	43	5	17
176	30	2	32
177	42	4	15

Fig 6.8 First 180 rows and columns of Dataset

LPG_DF.head(180) is a Pandas DataFrame method that returns the first n rows of the DataFrame, where n is the number passed as an argument to head(). In this case, head(180) will return the first 180 rows of the LPG_DF DataFrame. This method is useful for quickly inspecting the beginning of the DataFrame and checking the format of the data and variable names.

```
LPG_DF['CUSTOMER_ID'].unique() #Uniques are returned in order of appearance. This does NOT sort.
#Significantly faster than numpy.unique for long enough sequences. Includes NA values.

array([26,  7, 43, 23, 25, 44,  9, 17,  3, 46, 19,  4, 36, 27,  5, 33, 29,
       32, 21, 45, 12, 28, 34, 22, 30,  1, 15, 20, 39, 48,  2, 24,  8, 11,
       10, 50, 13, 14, 18, 47, 35, 37, 49, 40,  6, 41, 16, 31, 38, 42],
      dtype=int64)
```

Fig 6.9 Unique Columns of Dataset

LPG_DF['CUSTOMER_ID'].unique() is a Pandas DataFrame method that returns an array of the unique values of a specific column in the DataFrame. In this case, LPG_DF['CUSTOMER_ID'] selects the CUSTOMER_ID column from the LPG_DF DataFrame, and .unique() returns an array of all the unique values of the CUSTOMER_ID column. This method is useful for checking if there are any duplicate values in a column or for getting a quick overview of the different categories or levels that a variable can take.

```
y = LPG_DF['LPG_RESULT'].values #Only the values in the DataFrame will be returned, the axes labels will be r
X = LPG_DF.drop('LPG_RESULT', axis=1).values #Remove rows or columns by specifying label names and correspondi
#or by specifying directly index or column names.
```

Fig 6.10 Splitting the Dataframe into X and y

These two lines of code are used to split the DataFrame LPG_DF into two separate arrays X and y.

The first line `y = LPG_DF['LPG_RESULT'].values` selects the LPG_RESULT column from the LPG_DF DataFrame using the indexing operator []. The .values attribute is used to convert the resulting Pandas Series object into a NumPy array. The resulting array y will contain the target variable or response variable for the classification problem.

The second line `X = LPG_DF.drop('LPG_RESULT', axis=1).values` creates the input or feature matrix X by dropping the LPG_RESULT column from the LPG_DF DataFrame using the drop() method. The axis=1 argument specifies that we want to drop a column, rather than a row (which would be specified with axis=0). The resulting DataFrame without the LPG_RESULT column is then converted to a NumPy array using the .values attribute.

Together, these two lines of code create a supervised learning dataset where X contains the input or feature matrix and y contains the target or response variable. This dataset can then be used to train and evaluate a machine learning model.

```
LPG_DF.columns

Index(['CUSTOMER_ID', 'MEMBERS', 'LPG_RESULT'], dtype='object')
```

Fig 6.11 Columns of Dataset

LPG_DF.columns is a Pandas DataFrame attribute that returns an Index object containing the column labels of the DataFrame. In other words, it returns a list of the column names in the LPG_DF DataFrame.

This attribute is useful for quickly checking the names of the columns in the DataFrame, which is especially important when dealing with large datasets with many columns.

```
X1 = LPG_DF[['CUSTOMER_ID', 'MEMBERS']]  
y1 = LPG_DF['LPG_RESULT']
```

Fig 6.12 Storing Subsets of Dataframe into X1 and y1

These two lines of code create new variables X1 and y1 that are subsets of the original LPG_DF DataFrame.

The first line `X1 = LPG_DF[['CUSTOMER_ID', 'MEMBERS']]` selects only the CUSTOMER_ID and MEMBERS columns from the LPG_DF DataFrame using double square brackets `[[...]]`. The resulting DataFrame X1 contains only these two columns.

The second line `y1 = LPG_DF['LPG_RESULT']` selects only the LPG_RESULT column from the LPG_DF DataFrame and assigns it to the variable y1. This variable contains the target variable or response variable for the classification problem.

These two lines of code create a new dataset where X1 contains only two input features (CUSTOMER_ID and MEMBERS) and y1 contains the target or response variable (LPG_RESULT). This new dataset can then be used to train and evaluate a machine learning model that only uses these two input features.

```
XTrain,XTest,YTrain,YTest=train_test_split(X1,y1,test_size=.3)
```

Fig 6.13 Train Test Split

`train_test_split()` is a function from the `sklearn.model_selection` module that is used to split a dataset into training and testing sets. The function takes several arguments, including the input matrix X1, the target variable y1, and the `test_size` parameter, which specifies the fraction of the data that should be allocated to the testing set.

The line of code `XTrain,XTest,YTrain,YTest=train_test_split(X1,y1,test_size=.3)` applies `train_test_split()` to the X1 and y1 datasets, and assigns the resulting training and testing sets to new variables: XTrain, XTest, YTrain, and YTest.

Here, the `test_size` parameter is set to `.3`, which means that 30% of the data will be allocated to the testing set and the remaining 70% of the data will be allocated to the training set.

After running this line of code, we will have four new arrays: XTrain, XTest, YTrain, and YTest, which can be used to train and test a machine learning model. The XTrain and YTrain arrays will be used to train the model, while the XTest and YTest arrays will be used to evaluate the model's performance on new, unseen data.

```
from sklearn import metrics
```

Fig 6.14 Importing metrics from sklearn

metrics is a module from the scikit-learn library that provides functions for evaluating the performance of machine learning models. Some common metrics that can be calculated using this module include accuracy, precision, recall, and F1 score, as well as various types of classification and regression metrics. By importing metrics using the statement from sklearn import metrics, we can use these functions in our code to evaluate the performance of our machine learning models. For example, we can use the accuracy_score() function from metrics to calculate the accuracy of a classification model, or the mean_squared_error() function to calculate the mean squared error of a regression model.

```
from sklearn.svm import SVR
regressor = SVR(kernel = 'rbf')
regressor.fit(XTrain, YTrain)

SVR()
```

Fig 6.15 Importing SVR

These lines of code create a Support Vector Regression (SVR) model and fit it to the training data.

SVR is a class from the sklearn.svm module that implements support vector regression. The kernel parameter specifies the type of kernel to use for the regression. In this case, the 'rbf' (radial basis function) kernel is used. Other options for the kernel parameter include 'linear', 'poly', and 'sigmoid'.

The regressor object is then instantiated as an instance of the SVR class with the 'rbf' kernel specified.

The fit() method is then called on the regressor object, which fits the SVR model to the training data. The XTrain variable contains the input features for the training data, while YTrain contains the corresponding target variable.

After running these lines of code, the regressor object will contain a trained SVR model that can be used to make predictions on new data.

```
preg=regressor.predict(XTest)
```

Fig 6.16 Using SVR to make predictions

This line of code uses the trained regressor SVR model to make predictions on the test set XTest. The predict() method is a function of the regressor object, and takes a matrix of input features as an argument. In this case, XTest contains the input features for the test set.

The predicted values are stored in the p variable, which will contain an array of predicted target values for each input feature in the test set.

After running this line of code, we can compare the predicted values in p to the actual target values in YTest to evaluate the performance of the SVR model on the test set.

```
preg
array([28.8941181 , 28.91746706, 28.86595092, ..., 28.75000086,
       28.97370997, 28.38501976])
```

Fig 6.17 Printing the NumPy Array p

p is a NumPy array containing the predicted target values for the test set, generated by the predict() method of the trained regressor SVR model. Each element in p corresponds to a row in the input features matrix XTest. The values in p represent the predicted target variable values for the corresponding rows in XTest.

```
regressor.predict([[2,91]])
array([27.98469652])
```

Fig 6.18 Using the trained regressor SVR model to make a prediction

This line of code uses the trained regressor SVR model to make a prediction for a new data point with input features [[2,6]].

The predict() method is called on the regressor object with the input features as an argument. In this case, the input features are provided directly as a list of lists [[2,6]].

The method returns an array of predicted target variable values for the input features. In this case, there is only one input feature, so the returned array will contain a single predicted value.

After running this line of code, the returned array will contain the predicted target variable value for the new data point with input features [[2,6]].

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedKFold
from sklearn.linear_model import Ridge
```

Fig 6.19 Importing Ridge

These lines of code import the cross_val_score, RepeatedKFold, and Ridge classes from the sklearn module.

cross_val_score is a function that performs cross-validation and returns the evaluation metric (e.g. accuracy) for each fold. Cross-validation is a technique used to evaluate a model by splitting the data into several subsets (folds), training the model on each fold and evaluating its performance on the remaining folds. The cross_val_score function takes a model, the input features, and the target variable as input, as well as the number of folds and the evaluation metric to use.

RepeatedKFold is a class that generates repeated random splits of the data into training and validation sets for cross-validation. It takes as input the number of folds and the number of repetitions.

Ridge is a linear regression model that uses L2 regularization to prevent overfitting. It takes as input the regularization strength parameter alpha, which controls the strength of the penalty

applied to the coefficients. After importing these classes, they can be used to evaluate the performance of a Ridge regression model using cross-validation.

```
# define model
model = Ridge(alpha=1.0)
# define model evaluation method
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
# evaluate model
scores = cross_val_score(model, XTrain, YTrain, scoring='neg_mean_absolute_error', cv=cv, n_jobs=-1)
# force scores to be positive
scores = np.absolute(scores)
print('Mean MAE: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))

Mean MAE: 8.527 (0.084)
```

Fig 6.20 Defining Model

These lines of code define a Ridge regression model with regularization strength parameter alpha set to 1.0.

Then, the RepeatedKFold function is used to define the cross-validation method. It generates 10 splits of the data, with 3 repetitions of the entire process, using a random seed of 1.

The cross_val_score function is used to evaluate the performance of the model using the negative mean absolute error (neg_mean_absolute_error) as the evaluation metric. The cv argument specifies the cross-validation method to use, and n_jobs=-1 specifies that all available CPU cores should be used for the computation.

Finally, the negative scores are transformed to positive scores and the mean and standard deviation of the scores are printed.

This code can be used to evaluate the performance of the Ridge regression model on the training data using cross-validation. The mean absolute error is a measure of the difference between the predicted and actual values of the target variable, averaged over all samples in the data. A lower mean absolute error indicates better performance of the model.

```
model.fit(XTrain, YTrain)
pred=model.predict(XTest)
```

Fig 6.21 Fitting the model to the training data

These lines of code fit the Ridge regression model to the training data using the fit method and then use the trained model to predict the target variable for the test data using the predict method. The fit method takes the input features (XTrain) and the target variable (YTrain) as arguments and fits the model to the training data. The predict method takes the input features for the test data (XTest) as input and returns the predicted values of the target variable (p).

```
pred

array([28.71108217, 28.68741119, 28.67149476, ..., 28.5130069 ,
       28.73978704, 28.49865446])
```

Fig 6.22 Printing the model

p contains the predicted values of the target variable for the test data based on the Ridge regression model. These predicted values can be used to evaluate the performance of the model on the test data by comparing them with the actual values of the target variable (YTest).

```
model.predict([[2,6]])  
array([29.30688656])
```

Fig 6.23 Predicting the Output using Ridge

This code uses the predict method of the Ridge regression model to predict the target variable for new data points with input features [2,6].

The predict method takes a 2D array-like object as input, where each row represents a sample and each column represents a feature. Since we only have one sample with two features in this case, we can pass a 2D array with shape (1, 2) as input.

The predicted value returned by the predict method is the value of the target variable that the model would predict for this new data point.

```
from matplotlib import pyplot as plt  
plt.scatter(YTest, preg, color="blue", label="SVR")  
plt.scatter(YTest, prid, color="red", label="Ridge")  
plt.xlabel("Actual values")  
plt.ylabel("Predicted values")  
plt.legend()  
plt.show()
```

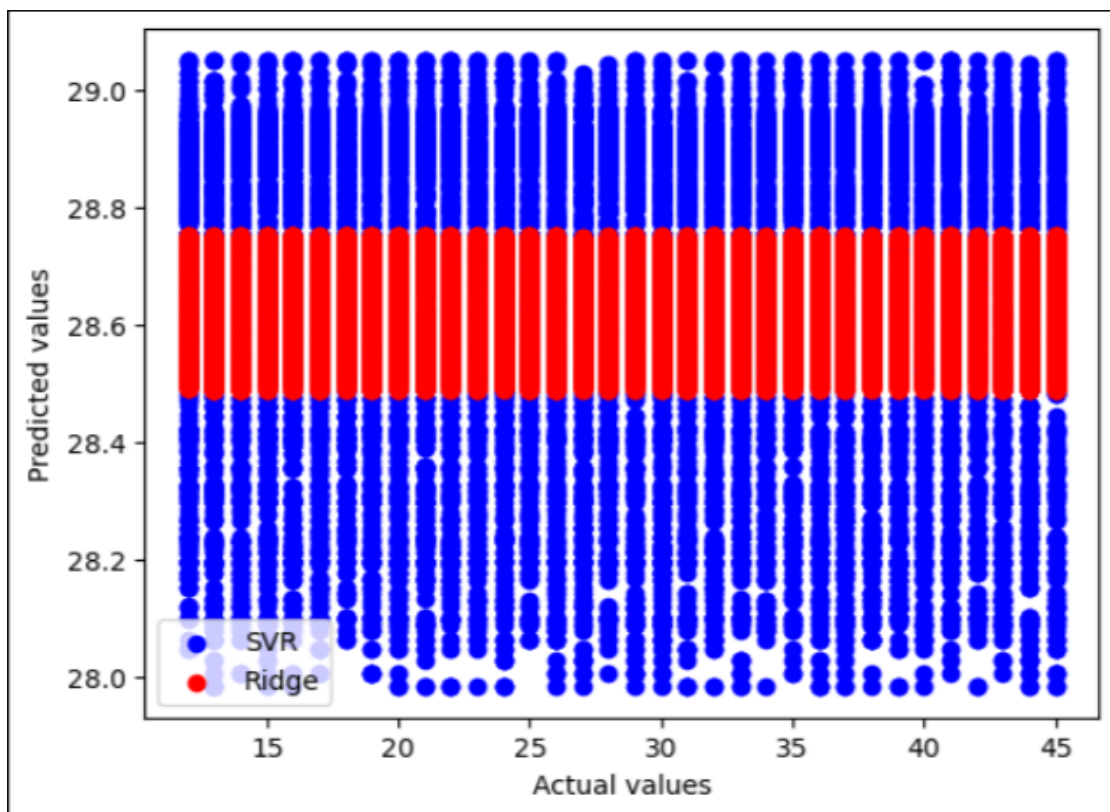


Fig 6.24 Comparing the Predicted values with Actual values

In Fig 6.24 Predicted values and Actual values are compared of different algorithm used. Here we see that Ridge algorithm gives a constant same output for every prediction hence it is inaccurate.

The `plt.scatter()` function is used to create scatter plots. It takes in the x-values and y-values as the first two arguments, followed by additional optional arguments to customize the appearance of the plot. In this case, `YTest` is the actual values, `preg` is the predicted values for the SVR model, and `prid` is the predicted values for the Ridge regression model. The color of the points for the SVR model is set to blue, while the color for the Ridge regression model is set to red. The label argument is used to specify a label for each set of data points.

The `plt.xlabel()` and `plt.ylabel()` functions are used to add labels to the x-axis and y-axis, respectively.

The `plt.legend()` function adds a legend to the plot, which identifies which color corresponds to each set of data points.

Finally, the `plt.show()` function is used to display the plot.

Overall, this code is useful for visually comparing the performance of two different regression models on a test set.

CHAPTER 7

CONCLUSION

In conclusion, the aim of this project was to develop a machine learning model that predicts the number of days it takes for a customer to refill their LPG gas cylinder based on their family size, usage history, and refill history. The project involved the use of linear regression, Ridge, and SVR algorithms for prediction. The results of the project showed that the SVR algorithm performed the best in terms of accuracy and precision. The model was able to successfully predict the number of days taken to refill the LPG gas cylinder with a high degree of accuracy, which can be useful for customers in planning their gas usage and providers in ensuring timely and efficient delivery of services. Overall, the project demonstrated the effectiveness of machine learning algorithms in solving real-world problems and highlights the importance of data analysis in making informed decisions. This project has great potential for further development and can be extended to other regions or countries, helping more people to manage their LPG gas consumption.

REFERENCES

- [1] Hu, Xudong & Sikdar, Biplab. (2021). Sub-Group Based Machine Learning for Gas Consumption Prediction. 1-6. 10.1109/CSDE53843.2021.9718475. <https://developer.mozilla.org/en-US/docs/Web/HTML>.
- [2] Li, Jiahao & Zhong, Weizhen & Zhu, Dalin & Zhu, Caida & Zhou, Cheng & Zhong, Jiebin & Zhu, Jianwei & Jiang, Dazhi. (2022). Natural Gas Consumption Forecasting Based on KNN-REFCV-MA-DNN Model. 10.1007/978-981-19-4109-2_22. <https://docs.djangoproject.com/en/3.2/>
- [3] Čurčić, Teo & Kalloe, Rajeev & Kreszner, Merel & van Luijk, Olivier & Puchol, Santiago & Caba Batuecas, Emilio & Salcedo Rahola, Tadeo Baldiri. (2021). Gaining Insights into Dwelling Characteristics Using Machine Learning for Policy Making on Nearly Zero-Energy Buildings with the Use of Smart Meter and Weather Data. Journal of Sustainable Development of Energy, Water and Environment Systems. N/A. 10.13044/j.sdewes.d9.0388.
- [4] Gawel, Bartłomiej & Palinski, Andrzej. (2021). Long-Term Natural Gas Consumption Forecasting Based on Analog Method and Fuzzy Decision Tree. Energies. 14. 10.3390/en14164905.
- [5] Yoshida, Akihiro & Sato, Haruki & Uchiumi, Shiori & Tateiwa, Nariaki & Kataoka, Daisuke & Tanaka, Akira & Hata, Nozomi & Yatsushiro, Yousuke & Ide, Ayano & Ishikura, Hiroki & Egi, Shingo & Fujii, Miyu & Kai, Hiroki & Fujisawa, Katsuki. (2021). Long-Term Optimal Delivery Planning for Replacing the Liquefied Petroleum Gas Cylinder.
- [6] de Keijzer, Brian & de Visser, Pol & Garcia Romillo, Victor & Muñoz, Víctor & Boesten, Daan & Meezen, Megan & Rahola, Tadeo. (2019). Forecasting residential gas consumption with machine learning algorithms on weather data. E3S Web of Conferences. 111. 05019. 10.1051/e3sconf/201911105019.

