Department of Computer Engineering and Applications

GLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O. - Chaumuha,
Mathura - 281406

Declaration

We hereby declare that the work which is being presented in the Mini Project “**Data Prediction and Plotting using Python”,** in partial fulfilment of the requirements for Mini-Project LAB, is an authentic record of our own work carried under the supervision of **Mr. Piyush Vashistha, Assistant Professor, GLA University, Mathura**.

Adarsh Kushwaha
(161500030)

Abhishek Garg
(161500015)

Mohit Sharma
(161500324)

Ritika
(161500461)

Department of Computer Engineering and Applications
GLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O. - Chaumuha,
Mathura - 281406

Certificate

This is to certify that the project entitled **“*Data Prediction and Plotting using Python*”** carried out in Mini Project – II Lab is a bonafide work done by ***Adarsh Kushwaha (161500030), Abhishek Garg (160500015), Mohit Sharma (161500324) and Ritika (161500461)*** and is submitted in partial fulfilment of the requirements for the award of the degree Bachelor of Technology (Computer Science & Engineering).

Signature with date

Name of Supervisor: Mr. Piyush Vashistha Sir

Signature with date

Mr. Vaibhav Diwan Sir

(Mini Project Coordinator)

Date:

Acknowledgement

It gives us a great sense of pleasure to present the report of the B. Tech Mini Project undertaken during B. Tech. Third Year. This project in itself is an acknowledgement to the inspiration, drive and technical assistance contributed to it by many individuals. This project would never have seen the light of the day without the help and guidance that we have received.

Our heartiest thanks to ***Prof. (Dr.) Anand Singh Jalal,*** *Head of Dept., Department of CEA for providing us with an encouraging platform to develop this project, which thus helped us in shaping our abilities towards a constructive goal.*

We owe special debt of gratitude to ***Mr. Piyush Vashistha,*** *Assistant Professor, Department of CEA, for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. He has showered us with all his extensively experienced ideas and insightful comments at virtually all stages of the project & has also taught us about the latest industry-oriented technologies.*

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind guidance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Adarsh Kushwaha

Abhishek Garg

Mohit Sharma

Ritika

Abstract

At times, many start-ups fail as the entrepreneurs lack in the knowledge of the computers and how they can use them to expand or improve their business. Our idea was to create a simple application that can be use by every person having basic knowledge of computers. The person just has to arrange his data into a particular format or structure as defined by us and them he can both see the predictions for future and the plots of his data as well.

In our mini-project, we created an application using Python and some external libraries of Python like numpy and matplotlib that can take data from the users in the form of CSV files in the predefined format and the can manipulate that data to predict future values and plot the data imported in the forms of pie-charts and bar-graphs. The prediction for different entities can be done on an yearly as well as monthly basis for the future, whereas the plotting is done to clearly visualise and monitor the data.

Table of Contents

Chapter 1

Introduction

1.1 Motivation and Overview

The motivation for our project came from the feeling of increasing the use of computers across the entire country. Our main objective was to help the who are trying to establish a new business but are not well acquainted with the world of computing and how computers can be used to increase their business many folds. We developed an application that can predict and plot data imported by it as per the user requirements.

1.2 Objective

The final application developed is capable of reading CSV files given by the user and reading data from it, and then the user can carry out the operation he wishes to, either predicting future values or making plots in the form of pie-charts and bar-graphs as per his needs. One of the constraints of the application is that the user needs to enter the data in the CSV file as per the given frame only or else the application would not be able to responds perfectly.

1.3 Definitions, Acronyms, and Abbreviations

1.3.1 Definitions

- Datasets : The data provided to plot and analyse by the user which will be imported by the application.
- matplotlib : Python library for plotting of data.
- numpy : Python library for storing data in different forms and performing calculations on it using pre-defined functions.
- scipy : Python library for manipulation of data using scientific methods.

- tkinter : Python library for developing GUI-based applications.

1.4 Purpose

The purpose of this project developed by us in Python is that is can be used for predicting future values and plotting of datasets. It also has a GUI-based interface.

1.5 Scope

1. The application developed as per this project can be used for plotting and predicting of data by importing datasets for the same in a particular frame of data.

2. Predicted data values and plotted graphs like bar-graph and pie-charts would be generated as the output of this application.

3. A limitation that this application would have is that it would not be able to predict or plot data if entered in a form which is different from the given format.

1.6 Future Plans

In future, we are planning to expand our project by applying the concepts of machine learning to the application can extract useful data from data entered by the user and perform operations on it. Thus, the user would not be required to enter the data in the required format.

Chapter 2

Software Requirement Analysis

2.1 Problem Statement

The problem we are trying to solve here is to help people with limited knowledge of computers by enabling them to predict and plot the data they as per their requirements. Our main focus was to target businessmen who have not established their business. Our aim was to help the monitor and analyse their sales on different scales and see predictions about their sales in future using past data.

2.2 Main Functionalities of the modules

2.2.1 Plotting of Data

- Introduction : This function is to plot the imported data from datasets on various types of graphs as per the compatibility of the data.
- Inputs : Datasets(as per the prescribed format by the developers)
- Processing : Application and processing using matplotlib and tkinter library of Python.
- Outputs : Graphs are generated as per the user's requirement and data's compatibility.

2.2.2 Prediction of Data

- Introduction : This function is to analyse and predict future data that would be generated as per the given datasets.
- Inputs : Datasets(as per the prescribed format by the developers) and prediction description.

- Processing : using various external libraries of python such as numpy and complex calculations would be made to generate the results.
- Outputs : Values for future are generated as the output of this function.

2.3 Non-Functional Requirements

2.3.1 Performance

The system or the application should deliver its best performance at all times if the datasets imported for plotting or predicting are in the prescribed format as per the developers. It should not slow down or show any fatal errors at these instances, given that the hardware and OS are performing at their best.

2.3.2 Reliability

The application must be reliable as the graphs generated should be accurate and easily comprehensible even to the lay-men. They must be neatly presented by GUI for the user. The predicted data should be at least have an accuracy of 70% with precise outputs.

2.3.3 Availability

The application must have an all time availability as it does not require anything specific like Internet connectivity for its performance.

2.3.4 Security

Due to no need of Internet connectivity, the application must be really secure and there should not be any sort of leakage of data during any point of operation.

2.3.5 Maintainability

The application is easily maintainable and its requires only updating the python version and the user needs to make sure that all the files of the entire software are kept in the same directories as they are kept initially by the developers.

2.3.6 Portability

The application hai highly portable as the source can be easily copied from one place to another place and it only needs python 3.x and the external libraries to be installed on the host machine.

2.4 Hardware Requirements

- Laptop with minimum 4 GB RAM

2.5 Software Requirements

- Python 3.x
- External Libraries : numpy, matplotlib and tkinter(Tcl/Tk for Mac OS X).
- Any Spreadsheet Software for CSV files.

Chapter 3

Software Design

3.1 Data Flow Diagrams

3.1.1 DFD Level 0

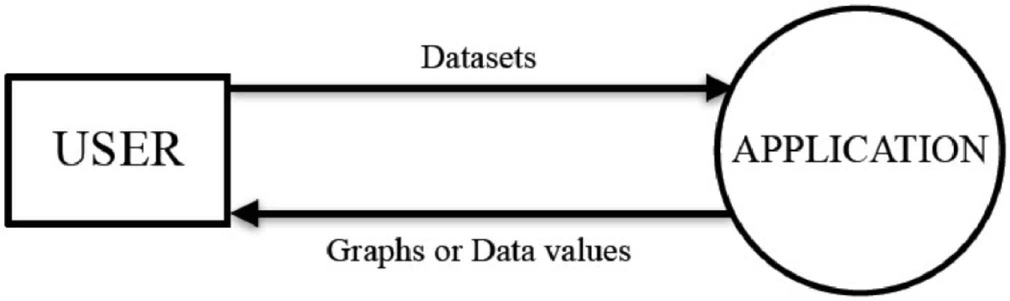


Fig. 3.1.1 : DFD Level 0

3.1.2 DFD Level 1

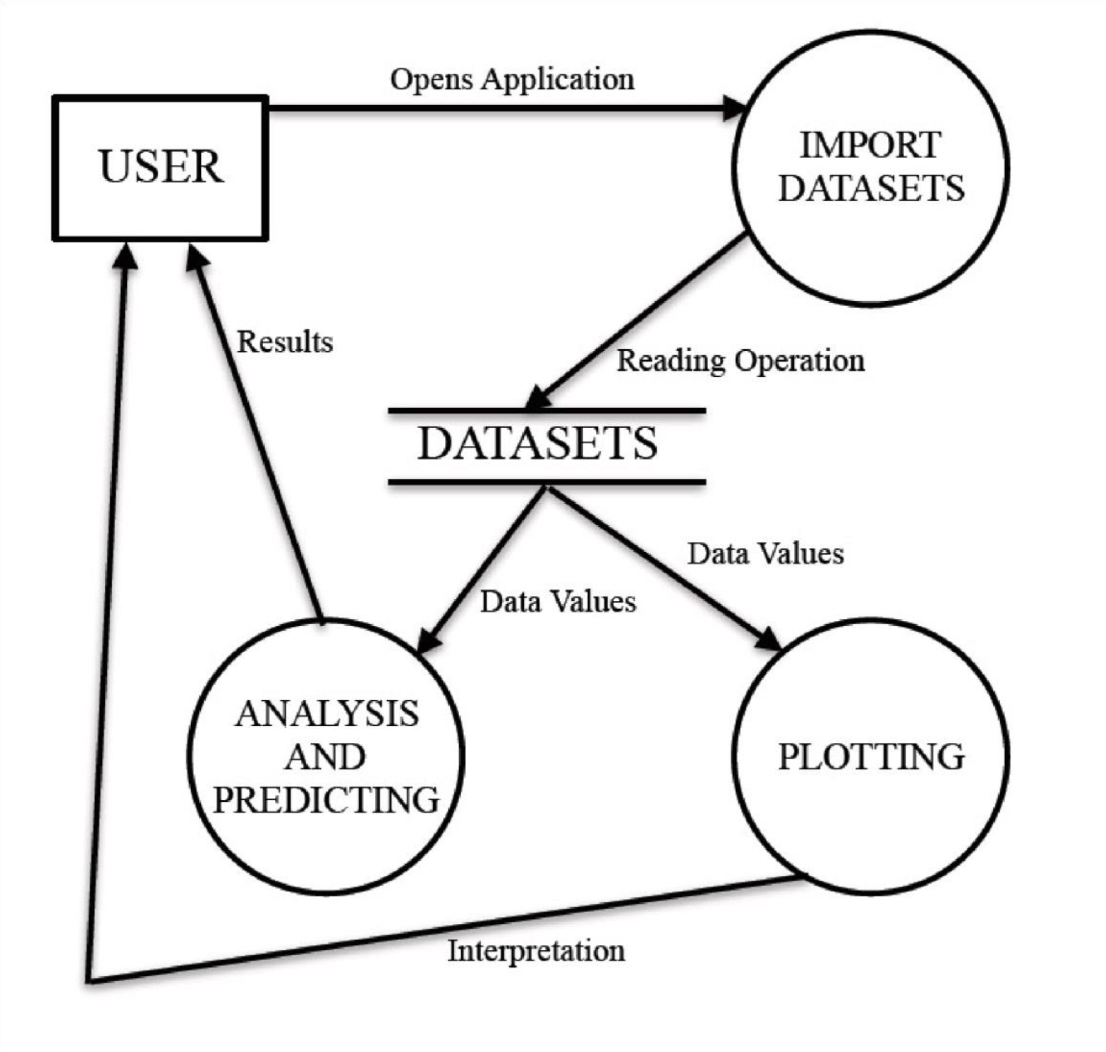


Fig 3.1.2 : DFD Level 1

3.2 Sequence Diagrams

3.2.1 Plotting Of Graph

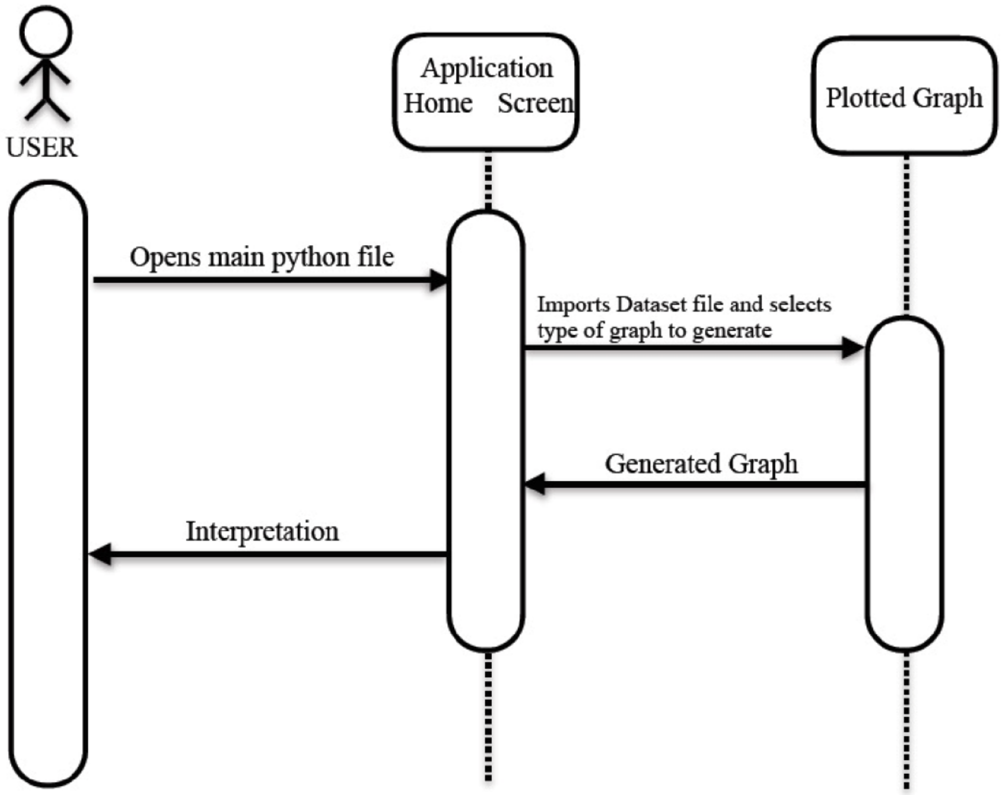


Fig. 3.2.1 : Sequence Diagram for Plotting Model.

3.2.2 Analysis and Predicting of Data

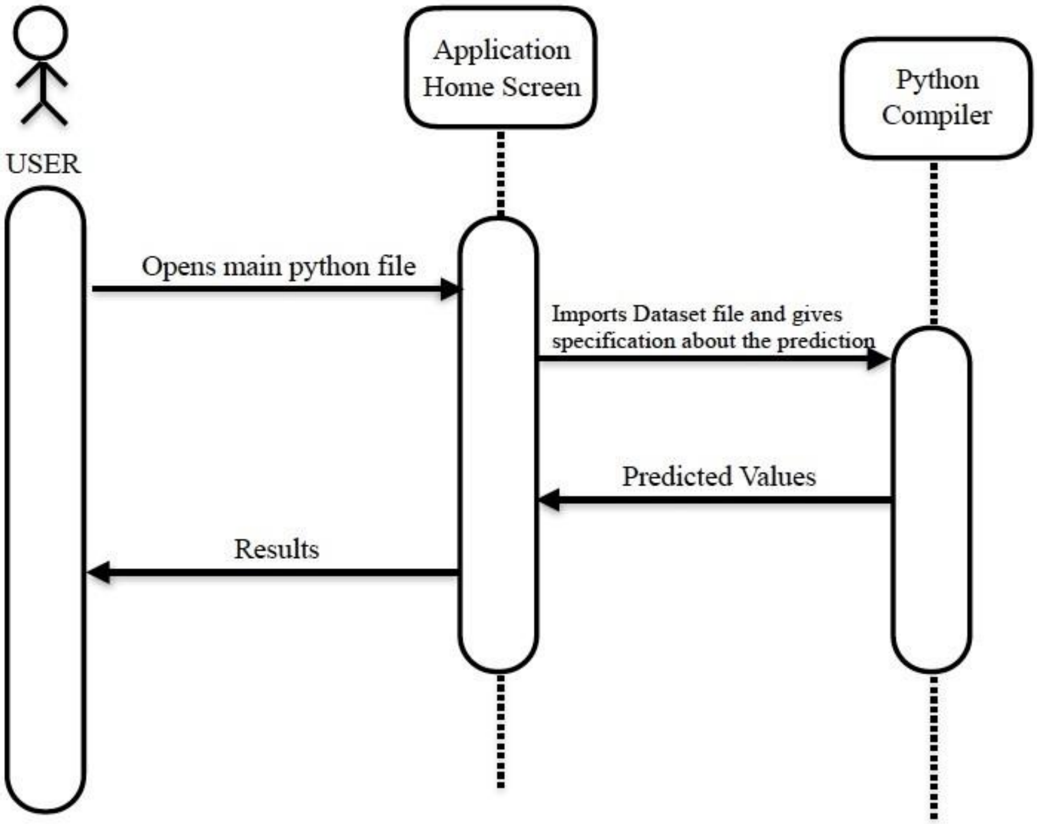


Fig. 3.2.2 : Sequence Diagram for Prediction Mode

3.3 Usecase Diagram

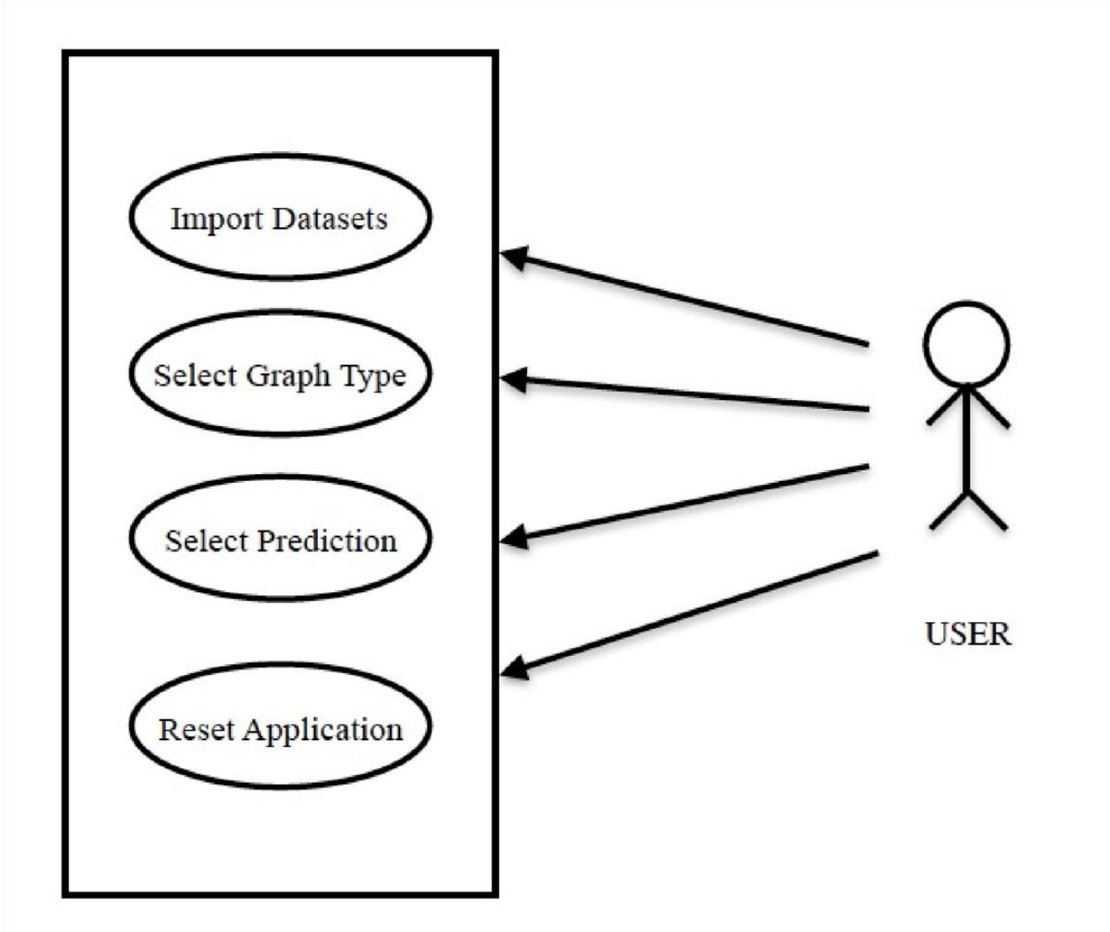


Fig. 3.3.1 : Usecase Diagram

Chapter 4

Testing

4.1 Black Box Testing

The Testing method used for Black Box Testing is Equivalence Class Testing. The reason for using this approach is that our application has so may combinations of values for its variable that the main logic function can accept.

Variables For The Main Function (possible number of values it can take) :

- md (2)
- yr (11)
- mn (13)
- cr (9)
- gr (3)
- py (14)

Therefore, the total number of distinct test could have been :
2*11*13*9*3*14=1,08,108(impossible to test and show results for)

So, in Equivalence Class Testing, 0 denotes ALL(all values taken into consideration) and 1 denotes any specific value and N/A denotes that the value of that variable is not applicable for that scenario.

Table for Black Box Testing :

Sno.	Mode (md)	Year (yr)	Month (mn)	Car (cr)	Graph (gr)	Prediction Year (py)	Output Generated
1	1	N/A	0	1	N/A	1	Predicted value for a car for a year.
2	1	N/A	1	1	N/A	1	Predicted value for a car in a particular month in a year.
3	1	0	1	1	1	N/A	Plot for any particular car sold in the years in a given month.
4	1	1	0	1	1	N/A	Plot for a particular car sold in all months of a year.

Table 4.1 : Equivalence Class Testing Table

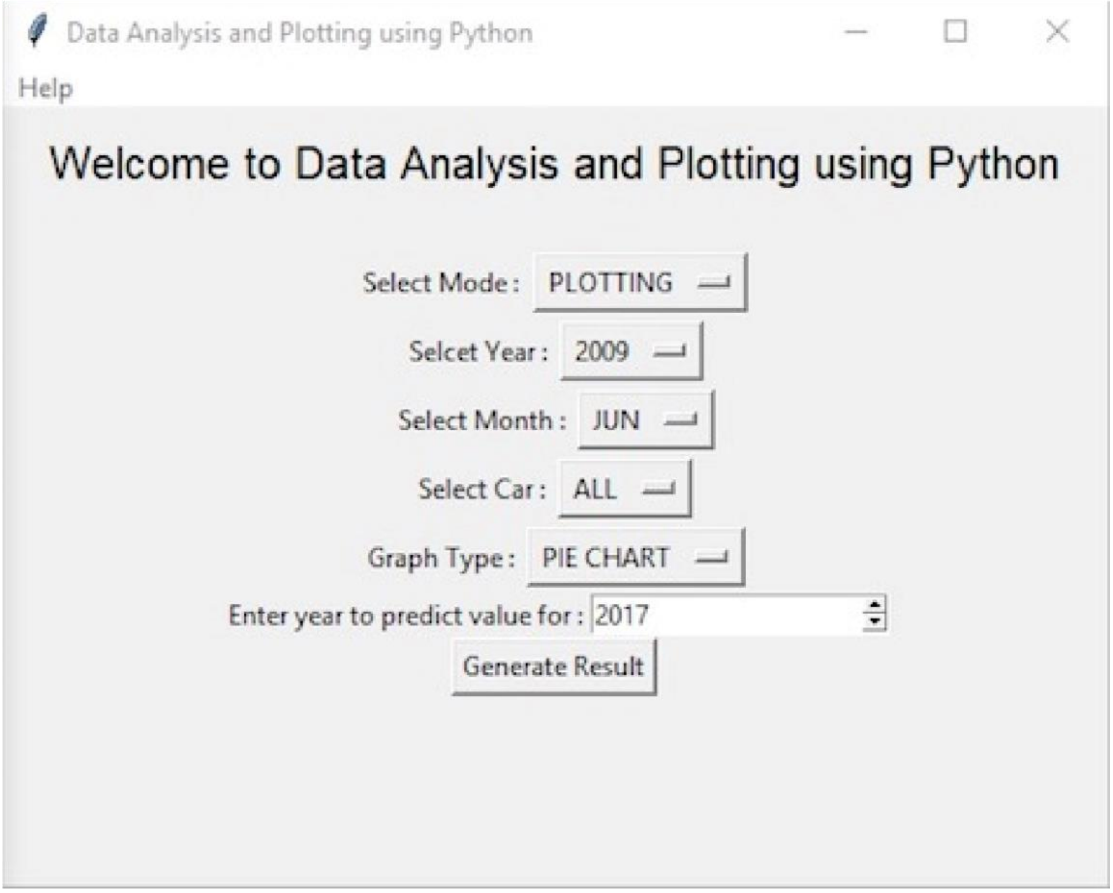
Chapter 5

Implementation and User Interface

5.1 User Interfaces

The application has been given a GUI-based interface for the user to interact with it, select the working mode, enter the details required and generate outputs as per their needs.

Following are the image of the interfaces : **5.1.1 Home Screen**



Data Analysis and Plotting using Python

Welcome to Data Analysis and Plotting using Python

Select Mode : PLOTTING

Selcet Year : 2009

Select Month : JUN

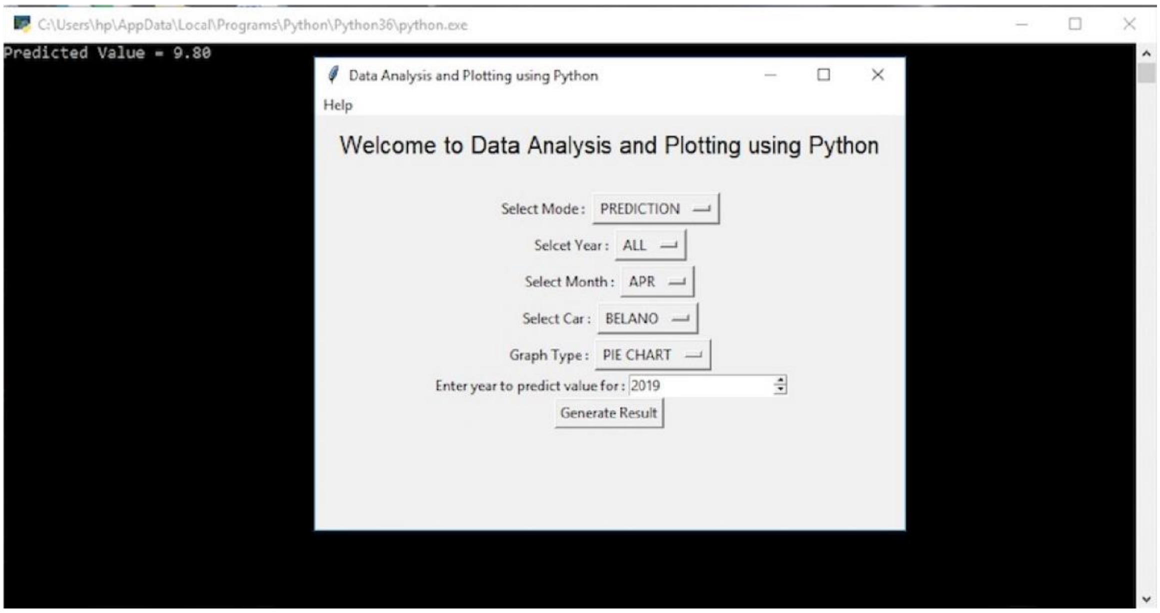
Select Car : ALL

Graph Type : PIE CHART

Enter year to predict value for : 2017

Fig 5.1 : Home Screen

This is the home or main screen of our application. It shows all the distinct options as dropdown menus and the year for prediction is a spinbox widget of Tkinter. The menu


C:\Users\hp\AppData\Local\Programs\Python\Python36\python.exe
Data Analysis and Plotting using Python
Help
Generate Result

```
Predicted Value = 9.80
```

Welcome to Data Analysis and Plotting using Python

Select Mode : PREDICTION

Selcet Year : ALL

Select Month : APR

Select Car : BELANO

Graph Type : PIE CHART

Enter year to predict value for : 2019

bar shows the option of Help as a dropdown. The user can easily select the values he needs to generate result for.

5.1.2 Prediction Mode Screen

Fig. 5.2 : Screen for Prediction Mode

This is the screen for the prediction mode and the console in the background shows the predicted value as 9.80 for Belano car in the month of April for the year 2019.

This value has been calculated as the previous data recored in the CSV files.

•

5.1.3 Pie Chart Plotting Output Screen

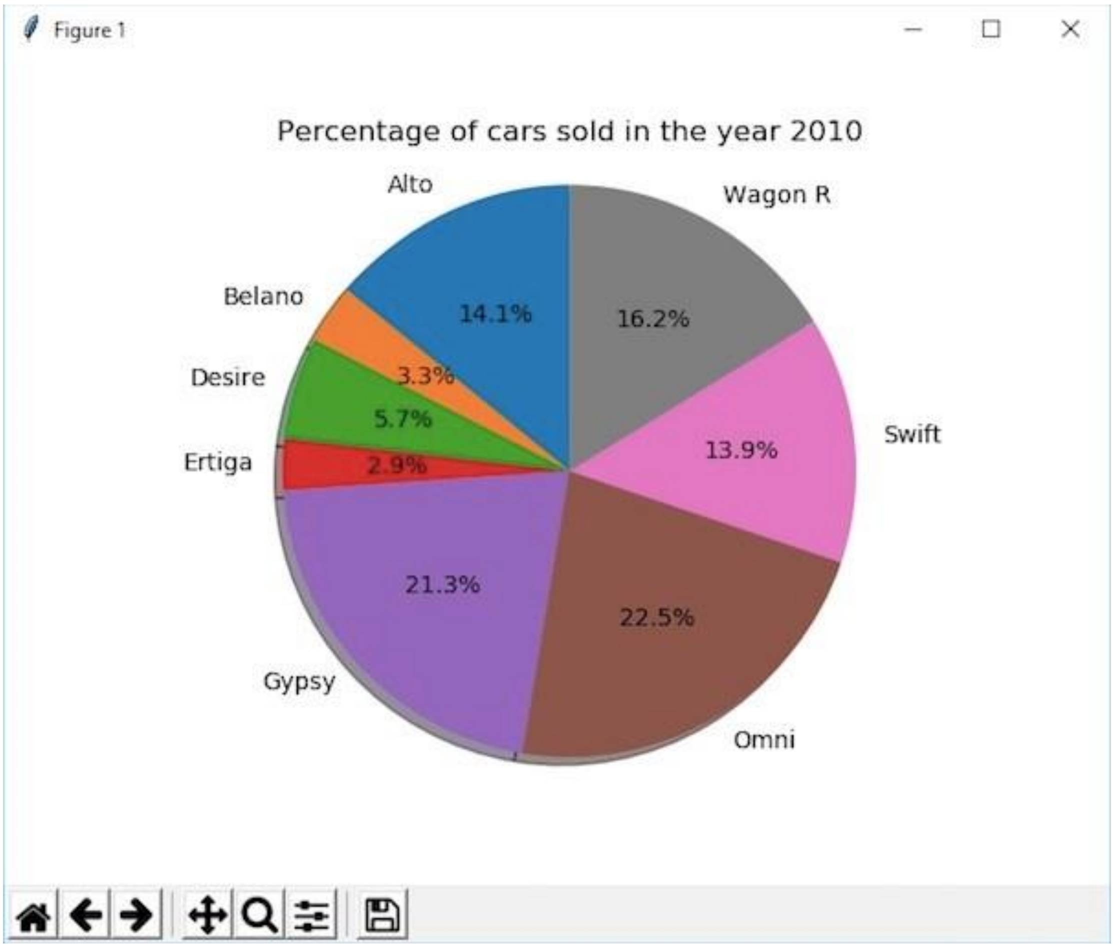


Fig. 5.3 : Plotting of Pie Chart

This is the output generated while generating pie chart for the analysing cars sold in the year 2010 across all the months. The piechart shows the percentages of all the cars correct to one decimal place.

5.1.4 Bar Graph Plotting Output Screen

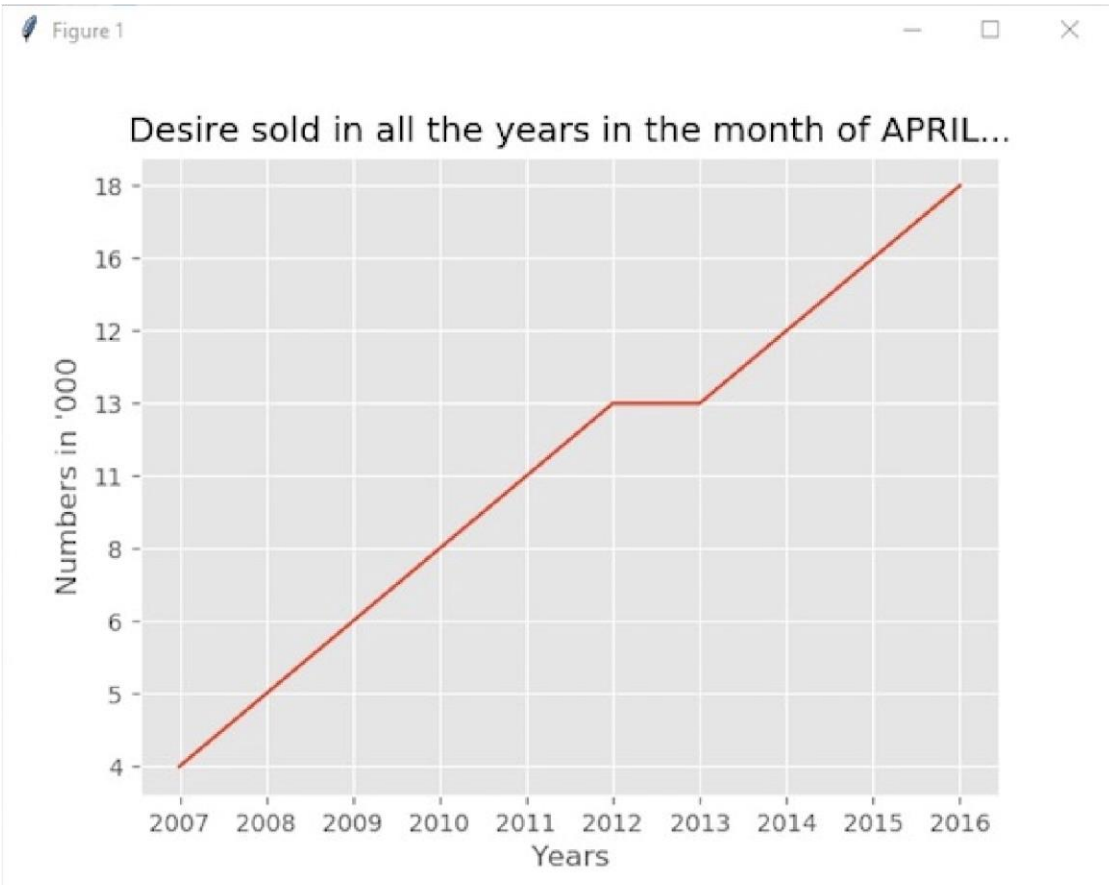


Fig. 5.4 : Plotting of Bar Graph

This is the output screen for the bar graph generated for showing all the Desires sold in the month of April for all the year.

CLI_CODE

```
import csv
import matplotlib.pyplot as plt
import matplotlib.style as stl
import numpy as np

def function(mode, year, month, car, pre_year, graph):

    if(mode==1):
        file=""
        counter=1
        r=0
        data=[]
        pre_year=pre_year%2016

        if(month==0):
            file="cars.csv"
        else:
            if (car==1):
                file="Alto.csv"
            if (car==2):
                file="Belano.csv"
            if (car==3):
                file="Desire.csv"
            if (car==4):
                file="Ertiga.csv"
            if (car==5):
                file="Gypsy.csv"
            if (car==6):
```

```
            file="Omni.csv"
        if (car==7):
            file="Swift.csv"
        if (car==8):
            file="WagonR.csv"

    with open(file) as csvfile:
        read=csv.reader(csvfile, delimiter=',')
        next(read)

        if(month==0):
            for row in read:
                if(counter<car):
                    counter=counter+1
                    continue
                else:
                    for i in range(1,11):
                        data.append(int(row[i]))
                    break
            for i in range(1,10):
                r=r+data[i]/data[i-1]
            r=r/9
            value=data[-1]*pow(r,pre_year)
            return (value)

        else:
            for row in read:
                if(counter<month):
                    counter=counter+1
                    continue
                else:
                        for i in range(1,11):
                            data.append(int(row[i]))
                        break
                for i in range(1,10):
                    r=r+data[i]/data[i-1]
                r=r/9
                value=data[-1]*pow(r,pre_year)
                return (value)

    else:
        car_labels = np.loadtxt('cars.csv', dtype='str', delimiter=',', skiprows = 1, usecols =
(0,))

        if(graph==1):
            if(year==0 and month!=0 and car!=0):
                if (car==1):
                    file="Alto.csv"
                if (car==2):
                    file="Belano.csv"
                if (car==3):
                    file="Desire.csv"
                if (car==4):
                    file="Ertiga.csv"
                if (car==5):
                    file="Gypsy.csv"
                if (car==6):
                    file="Omni.csv"
                if (car==7):
                    file="Swift.csv"
                if (car==8):
                    file="WagonR.csv"
                data = np.loadtxt(file, dtype='str', delimiter=',', usecols =
(1,2,3,4,5,6,7,8,9,10))
                month_labels = np.loadtxt(file, dtype='str', delimiter=',', usecols = (0,))
                month_name=month_labels[month]
                car_name=car_labels[car-1]
                values=[]
                years=[]
                for i in range(0,10):
                    years.append(data[0][i])
                    values.append(data[month][i])
                stl.use("ggplot")
                plt.plot(years,values)
                plt.title("%s sold in all the years in the month of
%s..."%(car_name,month_name))
                plt.xlabel("Years")
                plt.ylabel("Numbers in '000")
                plt.show()
            if(year!=0 and month==0 and car==0):
                data = np.loadtxt('cars.csv', delimiter=',', skiprows = 1, usecols = range(1,11))
                data=data.transpose()
                year=year%2007
                values=[]
                for i in range(0,8):
                    values.append(data[year][i])
                stl.use("ggplot")
                plt.plot(car_labels,values)
                plt.title("Cars sold in the year %d..."%(2007+year))
                plt.xlabel("Cars")
                plt.ylabel("Numbers in '000")
                plt.show()
            elif(year!=0 and month!=0 and car==0):
```

```
    year=year%2007
    values=[]
    for i in range(1,9):
        if (i==1):
            file="Alto.csv"
        if (i==2):
            file="Belano.csv"
        if (i==3):
            file="Desire.csv"
        if (i==4):
            file="Ertiga.csv"
        if (i==5):
            file="Gypsy.csv"
        if (i==6):
            file="Omni.csv"
        if (i==7):
            file="Swift.csv"
        if (i==8):
            file="WagonR.csv"
        data = np.loadtxt(file, dtype='str', delimiter=',',skiprows = 1, usecols = range(1,11))
        values.append(data[month-1][year])
    month_labels = np.loadtxt(file, dtype='str', delimiter=',', usecols = (0,))
    month_name=month_labels[month]
    stl.use("ggplot")
    plt.plot(car_labels,values)
    plt.title("Cars sold in the year %d, in the month of %s..."%((2007+year),month_name))
    plt.xlabel("Cars")
    plt.ylabel("Numbers in '000")
    plt.show()
    else:
        values=[]
        year=year%2007
        if (car==1):
            file="Alto.csv"
        if (car==2):
            file="Belano.csv"
        if (car==3):
            file="Desire.csv"
        if (car==4):
            file="Ertiga.csv"
        if (car==5):
            file="Gypsy.csv"
        if (car==6):
            file="Omni.csv"
        if (car==7):
            file="Swift.csv"
        if (car==8):
            file="WagonR.csv"

        if(car==0):
            data = np.loadtxt("cars.csv", dtype='str', delimiter=',', skiprows=1, usecols = range(1,11))
            car_labels = np.loadtxt('cars.csv', dtype='str', delimiter=',', skiprows = 1, usecols = (0,))
            for i in range(0,8):
                values.append(data[i][year])
            plt.pie(values, labels=car_labels, autopct='%1.1f%%', shadow=True, startangle=90)
            plt.axis('equal')
            plt.title("Percentage of cars sold in the year %d"%(2007+year))
```

```
                plt.show()
            else:
                data = np.loadtxt(file, dtype='str', delimiter=',', skiprows=1, usecols =
range(1,11))
                month_labels = np.loadtxt(file, dtype='str', delimiter=',', skiprows=1, usecols
= (0,))
                for i in range(0,12):
                    values.append(data[i][year])
                car_name=file[0:-4]
                plt.pie(values, labels=month_labels, autopct='%1.1f%%', shadow=True,
startangle=90)
                plt.axis('equal')
                plt.title("Percentage of %ss sold in all the months of year
%d"%(car_name,2007+year))
                plt.show()

        return (0)

mode=int(input("Enter mode...0 for Plotting and 1 for Prediction...\n"))

if (mode==0):
    pre_year=0
    graph=int(input("Enter Graph Type...0 for Pie Chart and 1 for Bar Graph...\n"))
    if(graph==1):
        year=int(input("Enter Year...0 for all...\n"))
        month=int(input("Enter Month Number...0 for all...\n"))
        car=int(input("Enter Car's Serial Number...0 for all...\n"))
    else:
        year=int(input("Enter Year...0 for all...\n"))
        car=int(input("Enter Car's Serial Number...0 for all...\n"))
        month=0
```

```
else:
    year=0;
    graph=0;
    car=int(input("Enter Car's Serial Number...0 for all...\n"))
    month=int(input("Enter Month Number...0 for all...\n"))
    pre_year=int(input("Enter Year for Prediction...\n"))

f=function(mode, year, month, car, pre_year, graph)
print("Predicted Value = %d\n" %(f))
```

INTERFACE_CODE

```
#Python Interface Program Using Tkinter and Other External Libraries to plot and predict data using given datasets
#importing required libraries(both internal and external)
import csv
import matplotlib.pyplot as plt
import matplotlib.style as stl
import numpy as np
from tkinter import *
#intialising global variables for use in all the function along with the "global" keyword before variable names
md=0
yr=0
mn=0
cr=0
py=0
gr=0
#making the window for the interface
window=Tk()
window.title("Data Analysis and Plotting using Python")
window.geometry('500x350') #setting default size of the window
#intialising variables for getting values from Tkinter widgets
mode=StringVar(window)
year=StringVar(window)
month=StringVar(window)
car=StringVar(window)
graph=StringVar(window)
#function for About option in the menu
def about():
```

```
about=Tk()
about.title("About")
about.geometry('235x165')
msg_about=Message(about, text="This application can be used to predict and plot data as per the data entered by the user. The prediction is done using the annual growth rate model. Plotting is done after reading the data from the files and plotting them on bar-charts or pie-charts, as per the user requirement.")
msg_about.pack()
about.mainloop()
#function for User Instructions option in the menu
def user_ins():
userins=Tk()
userins.title("User Instructions")
userins.geometry('340x260')
msg_user=Message(userins, text="\nSelect Plotting or Prediction of data.\n\nSelect year for which you want to import data. Select \"ALL\" for all years or any specific value for a particular year.\n\nSelect month as per your requirement,\"All\" or any specific value for the desired month.\n\nSelect car accordingly or select \"All\" for all cars.\n\nSelect type of graph you want to plot or select the year for which you want to predict data for.\n\nTo predict data for any specific month, select that month from the month dropdown.")
msg_user.pack()
userins.mainloop()
#function to get values entered by the user using the Tkinter widgets
def value_func(*args):
	global md, yr, mn, cr, py, gr #importing global variables as values we will be set to them accordingly
	m=mode.get() #xyz.get() : function to retrieve values from the variable in the widgets
	y=year.get()
	mon=month.get()
```

```
    c=car.get()
    g=graph.get()
    py=int(pre_year.get()) #converting to integer type
    #setting md variable to 0 or 1 for the mode of operation of the code selected by the user
    if(m=="PLOTTING"):
        md=0
    elif(m=="PREDICTION"):
        md=1
    #setting the yr variable as per the value of the year selected by the user
    if(y=="ALL"):
        yr=0
    if(y=="2007"):
        yr=1
    if(y=="2008"):
        yr=2
    if(y=="2009"):
        yr=3
    if(y=="2010"):
        yr=4
    if(y=="2011"):
        yr=5
    if(y=="2012"):
        yr=6
    if(y=="2013"):
        yr=7
    if(y=="2014"):
        yr=8
    if(y=="2015"):
        yr=9
    if(y=="2016"):
```

```
    yr=10
#setting the mn variable as per the value of the month selected by the user
if(mon=="ALL"):
    mn=0
if(mon=="JAN"):
    mn=1
if(mon=="FEB"):
    mn=2
if(mon=="MAR"):
    mn=3
if(mon=="APR"):
    mn=4
if(mon=="MAY"):
    mn=5
if(mon=="JUN"):
    mn=6
if(mon=="JUL"):
    mn=7
if(mon=="AUG"):
    mn=8
if(mon=="SEP"):
    mn=9
if(mon=="OCT"):
    mn=10
if(mon=="NOV"):
    mn=11
if(mon=="DEC"):
    mn=12
#setting the cr variable as per the car selected by the user
if(c=="ALL"):
    cr=0
```

```
        if(c=="ALTO"):
                cr=1
        if(c=="BELANO"):
                cr=2
        if(c=="DESIRE"):
                cr=3
        if(c=="ERTIGA"):
                cr=4
        if(c=="GYPSY"):
                cr=5
        if(c=="OMNI"):
                cr=6
        if(c=="SWIFT"):
                cr=7
        if(c=="WOGONR"):
                cr=8
        #setting the gr variable for the graph type required by the user
        if(g=="PIE CHART"):
                gr=-1
        else:
                gr=1
#main function of the application
def function():
    global md, yr, mn, cr, py, gr #importing the global variables with their changed values
    #working on prediction mode
    if(md==1):
        py=py%2016 #to see how years into the future the user wants to make the prediction for
        fl="" #empty string variable for storing the name of the file to be opened
        counter=1 #counter variable used for skipping rows in the data imported
        r=0 #variable to calculate and store the avaerage annual graowth rate
```

```
data=[] #list to read and store the data read from the CSV file
#selecting file to be opened, and writing its name to fl variable
if(mn==0):
   fl="cars.csv"
else:
   if (cr==1):
      fl="Alto.csv"
   if (cr==2):
      fl="Belano.csv"
   if (cr==3):
      fl="Desire.csv"
   if (cr==4):
      fl="Ertiga.csv"
   if (cr==5):
      fl="Gypsy.csv"
   if (cr==6):
      fl="Omni.csv"
   if (cr==7):
      fl="Swift.csv"
   if (cr==8):
      fl="WagonR.csv"
#opening file as csvfile into the variable read
with open(fl) as csvfile:
   read=csv.reader(csvfile, delimiter=',')
   next(read) #skipping the header rows
   #code for yearly prediction value
   if(mn==0):
      for row in read: #traversing rows of the imported data in the read variable
         if(counter<cr): #skipping to the car from the top of the read data
            counter=counter+1
            continue
```

```
                    else:
                        for i in range(1,11):
                            data.append(int(row[i])) #reading and appending the values to the data list
                        break
                for i in range(1,10): #traversing the values in the list
                    r=r+data[i]/data[i-1] #calculating cummulative sum of growth index for every year
                r=r/9 #finding average of the growth rate
                value=data[-1]*pow(r,py) #multiplying the value for year 2016 into growth factor for the required year
            #code for month wise yearly prediction value
            else:
                for row in read: #traversing rows of the imported data in the read variable
                    if(counter<mn): #skipping to the month from the top of the read data
                        counter=counter+1
                        continue
                    else:
                        for i in range(1,11):
                            data.append(int(row[i])) #reading and appending the values to the data list
                        break
                for i in range(1,10): #traversing the values in the list
                    r=r+data[i]/data[i-1] #calculating cummulative sum of growth index for every year
                r=r/9 #finding average of the growth rate
                value=data[-1]*pow(r,py) #multiplying the value for year 2016 into growth factor for the required year
        #printing the predicted value
        print("Predicted Value = %1.2f"%value)
    #code for plotting of data
    else:
        #making labels for car names into the variable car_labels
        car_labels = np.loadtxt('cars.csv', dtype='str', delimiter=',', skiprows = 1, usecols = (0,))
        #code for plotting of bar-graphs
        if(gr==1):
            #plotting for a particular car sold in a particular month of all years
            if(yr==0 and mn!=0 and cr!=0):
                #assinging file name to fl varaible
                if (cr==1):
                    fl="Alto.csv"
                if (cr==2):
                    fl="Belano.csv"
                if (cr==3):
                    fl="Desire.csv"
                if (cr==4):
                    fl="Ertiga.csv"
                if (cr==5):
                    fl="Gypsy.csv"
                if (cr==6):
                    fl="Omni.csv"
                if (cr==7):
                    fl="Swift.csv"
                if (cr==8):
                    fl="WagonR.csv"
                #importing data into data named variable
                data = np.loadtxt(fl, dtype='str', delimiter=',', usecols = (1,2,3,4,5,6,7,8,9,10))
                #making month names into the varaible month_labels
                month_labels = np.loadtxt(fl, dtype='str', delimiter=',', usecols = (0,))
                #assinging month name needed to the month_name variable
                month_name=month_labels[mn]
                #assinging car name needed to the car_name variable
                car_name=car_labels[cr-1]
                values=[] #list for values
                years=[] #list for year
                for i in range(0,10): #traversing through the data values in data variable
                    years.append(data[0][i]) #appending years to the years list
                    values.append(data[mn][i]) #appending values to the values variable
                #plotting code
                stl.use("ggplot") #using style as ggplot
                plt.plot(years,values) #plotting years on X-axis and values on Y-axis
                plt.title("%s sold in all the years in the month of %s..."%(car_name,month_name)) #giving title to the plot
                plt.xlabel("Years") #label on X_axis
                plt.ylabel("Numbers in '000") #label on Y_axis
                plt.show() #showing the plot, finally
            #plotting for all cars sold in all the months of a particular year
            if(yr!=0 and mn==0 and cr==0):
                #importing the data into the data variable from the cars.csv file
                data = np.loadtxt('cars.csv', delimiter=',', skiprows = 1, usecols = range(1,11))
                data=data.transpose() #transposing the data imported
                yr=yr%2007 #finding index for the year after doing modulus from 2007
                values=[] #creating empty list for the values
                for i in range(0,8): #looping for finding the values
                    values.append(data[yr][i]) #appending cars sold in a particular year for every car
                stl.use("ggplot") #using style as ggplot
                plt.plot(car_labels,values) #plotting Cars on X-axis and values on Y-axis
                plt.title("Cars sold in the year %d..."%(2007+yr)) #giving title to the plot
                plt.xlabel("Cars") #label on X_axis
                plt.ylabel("Numbers in '000") #label on Y_axis
                plt.show() #showing the plot, finally
```

```
#plotting for all cars sold in a particular month in a particular year
elif(yr!=0 and mn!=0 and cr==0):
    yr=yr%2007 #finding year index
    values=[] #empty list for values
    for i in range(1,9): #looping for the files, one by one
        if (i==1):
            fl="Alto.csv"
        if (i==2):
            fl="Belano.csv"
        if (i==3):
            fl="Desire.csv"
        if (i==4):
            fl="Ertiga.csv"
        if (i==5):
            fl="Gypsy.csv"
        if (i==6):
            fl="Omni.csv"
        if (i==7):
            fl="Swift.csv"
        if (i==8):
            fl="WagonR.csv"
        #importing all files, one by one
        data = np.loadtxt(fl, dtype='str', delimiter=',',skiprows = 1, usecols = range(1,11))
        #appending values for the asked month and the asked year
        values.append(data[mn-1][yr])
    #making month labels
    month_labels = np.loadtxt(fl, dtype='str', delimiter=',', usecols = (0,))
    #assinging month name to the month_name variable
    month_name=month_labels[mn]
    stl.use("ggplot") #using style as ggplot
```

```
            plt.plot(car_labels,values) #plotting Cars on X-axis and values on Y-axis
            plt.title("Cars sold in the year %d, in the month of %s..."%((2007+yr),month_name)) #giving title to the plot
            plt.xlabel("Cars") #label on X_axis
            plt.ylabel("Numbers in '000") #label on Y_axis
            plt.show() #showing the plot, finally
      #code for pie-charts
      elif(gr==-1):
         values=[] #empty list for the values
         #assinging file name to the fl variable
         if (cr==1):
            fl="Alto.csv"
         if (cr==2):
            fl="Belano.csv"
         if (cr==3):
            fl="Desire.csv"
         if (cr==4):
            fl="Ertiga.csv"
         if (cr==5):
            fl="Gypsy.csv"
         if (cr==6):
            fl="Omni.csv"
         if (cr==7):
            fl="Swift.csv"
         if (cr==8):
            fl="WagonR.csv"
         #plotting for all cars sold in a particular year
         if(cr==0):
            #reading cars.csv and giving cars names to the car_labels variable
            data = np.loadtxt("cars.csv", dtype='str', delimiter=',', skiprows=1, usecols = range(1,11))
            car_labels = np.loadtxt('cars.csv', dtype='str', delimiter=',', skiprows = 1,
usecols = (0,))
            #traversing the data
            for i in range(0,8):
                values.append(data[i][yr]) #appending values to the values list
            #making pie-chart
            plt.pie(values, labels=car_labels, autopct='%1.1f%%', shadow=True,
startangle=90)
            plt.axis('equal') #making axis equal
            plt.title("Percentage of cars sold in the year %d"%(2007+yr)) #title of the plot
            plt.show() #showing the plot, finally
        #plotting for a particular car sold in all months of a particular year
        else:
            #reading the file required and giving month names to the month_labels
variable
            data = np.loadtxt(fl, dtype='str', delimiter=',', skiprows=1, usecols =
range(1,11))
            month_labels = np.loadtxt(fl, dtype='str', delimiter=',', skiprows=1, usecols =
(0,))
            for i in range(0,12):
                values.append(data[i][yr]) #appending values to the values list
            car_name=fl[0:-4] #assinging car name to the car_name variable
            #making pie-chart
            plt.pie(values, labels=month_labels, autopct='%1.1f%%', shadow=True,
startangle=90)
            plt.axis('equal') #making axis equal
            plt.title("Percentage of %ss sold in all the months of year
%d"%(car_name,2007+yr)) #title of the plot
            plt.show() #showing the plot, finally
#code for interface of the program
#making menubar
```

```
menubar=Menu(window)
helpmenu=Menu(menubar, tearoff=0)
helpmenu.add_command(label="About", command=about)
helpmenu.add_command(label="User Instructions", command=user_ins)
helpmenu.add_separator() #adding seperator
helpmenu.add_command(label="Quit", command=window.quit)
menubar.add_cascade(label="Help", menu=helpmenu)
window.config(menu=menubar)
#making the frame for the heading
header=Frame(window, height=50, width=350, relief=RAISED)
heading=Label(header, text="Welcome to Data Analysis and Plotting using Python", font=("Helvetica", 15), pady=10)
heading.pack()
header.pack()
#making the frame for all other widgets
base_frame=Frame(window, height=425, width=300, relief=RAISED, padx=15, pady=15)
#making first frame for selecting the mode of operation
first_frame=Frame(base_frame)
select_mode=Label(first_frame, text="Select Mode :")
select_mode.pack(side=LEFT)
mode_frame=Frame(first_frame)
mode_value=OptionMenu(first_frame,mode,"PLOTTING","PREDICTION",command=value_func)
mode_value.pack(side=RIGHT)
mode_frame.pack(side=RIGHT)
first_frame.pack()
#making second frame for the selection of year
second_frame=Frame(base_frame)
year_label=Label(second_frame, text="Selcet Year :")
year_label.pack(side=LEFT)
```

```
year_value=OptionMenu(second_frame,year,"ALL","2007","2008","2009","2010","2011","2012","2013","2014","2015","2016",command=value_func)
year_value.pack(side=RIGHT)
second_frame.pack()
#making third frame for the selection of month
third_frame=Frame(base_frame)
month_label=Label(third_frame, text="Select Month :")
month_label.pack(side=LEFT)
month_value=OptionMenu(third_frame,month,"ALL","JAN","FEB","MAR","APR","MAY","JUN","JUL","AUG","SEP","OCT","NOV","DEC",command=value_func)
month_value.pack(side=RIGHT)
third_frame.pack()
#making fourth frame for the selection of car
fourth_frame=Frame(base_frame)
car_label=Label(fourth_frame, text="Select Car :")
car_label.pack(side=LEFT)
car_value=OptionMenu(fourth_frame,car,"ALL","ALTO","BELANO","DESIRE","ERTIGA","GYPSY","OMNI","SWIFT","WAGONR",command=value_func)
car_value.pack(side=RIGHT)
fourth_frame.pack()
#making fifth frame for the selection of graph type
fifth_frame=Frame(base_frame)
graph_label=Label(fifth_frame, text="Graph Type :")
graph_label.pack(side=LEFT)
graph_value=OptionMenu(fifth_frame,graph,"PIE CHART","BAR GRAPH",command=value_func)
graph_value.pack(side=RIGHT)
fifth_frame.pack()
#making sixth frame for the selection of year of prediction type
sixth_frame=Frame(base_frame)
pre_label=Label(sixth_frame, text="Enter year to predict value for :")
```

```
pre_label.pack(side=LEFT)
pre_year=Spinbox(sixth_frame, from_=2017, to=2030, command=value_func)
pre_year.pack(side=RIGHT)
sixth_frame.pack()
#making button for result generation
generate_result=Button(base_frame, text="Generate Result", command=function)
generate_result.pack()
base_frame.pack()
window.mainloop()
```

Chapter 6

References/Bibliography

6.1 Books Consulted

1. Introduction to Programming Using Python (Daniel Liang)
2. Learning Python, 5th Edition (Mark Lutz)
3. Python Essential Reference (David M. Beazley)

6.2 Websites Visited

1. www.google.com
2. https://www.python.org/downloads/release/python-360/
3. https://www.activestate.com/activetcl/downloads
4. https://plot.ly/python/pie-charts/
5. https://stackoverflow.com/questions/20714966/how-to-create-asimple-pie-chart-using-python
6. https://docs.python.org/2/library/csv.html
7. https://plot.ly/python/bar-charts/
8. https://pythonspot.com/tk-dropdown-example/

Chapter 7

Appendices

7.1 Coding Templates 7.1.1

Tkinter Code

```
from tkinter import *
#making the window for the interface
window=Tk()
window.title("Data Analysis and Plotting using Python") window.geometry('500x350')
#setting default size of the window
#making menubar menubar=Menu(window)
helpmenu=Menu(menubar, tearoff=0)
helpmenu.add_command(label="About",                              command=about)
helpmenu.add_command(label="User Instructions", command=user_ins)
helpmenu.add_separator() #adding seperator
helpmenu.add_command(label="Quit", command=window.quit)
menubar.add_cascade(label="Help", menu=helpmenu)
window.config(menu=menubar) #making the frame for the
heading
header=Frame(window, height=50, width=350, relief=RAISED)
heading=Label(header, text="Welcome to Data Analysis and Plotting using Python",
font=("Helvetica", 15), pady=10) heading.pack() header.pack()
#making the frame for all other widgets
base_frame=Frame(window, height=425, width=300, relief=RAISED, padx=15,
pady=15) base_frame.pack() window.mainloop()
```

7.1.2 Prediction Code

```
#code for yearly prediction value if(mn==0):
for row in read:
if(counter<cr):
counter=counter+1 continue
else:
```

```
for i in range(1,11):
data.append(int(row[i]))
break for i in
range(1,10):
r=r+data[i]/data[i-1]
r=r/9
```

```
value=data[-1]*pow(r,py)
```

7.1.3 Code for Plotting of Bar Graphs

```
#importing data into data named variable
data = np.loadtxt(fl, dtype='str', delimiter=',', usecols = (1,2,3,4,5,6,7,8,9,10))
#making month names into the varaible month_labels
month_labels = np.loadtxt(fl, dtype='str', delimiter=',', usecols = (0,)) #assinging
month name needed to the month_name variable
month_name=month_labels[mn]
#assinging car name needed to the car_name variable
car_name=car_labels[cr-1] values=[] #list for values
years=[] #list for year
for i in range(0,10): #traversing through the data values in data variable
years.append(data[0][i]) #appending years to the years list values.append(data[mn][i])
#appending values to the values variable
#plotting code
stl.use("ggplot") #using style as ggplot
plt.plot(years,values) #plotting years on X-axis and values on Y-axis plt.title("%s
sold in all the years in the month of %s..."%(car_name,month_name))
plt.xlabel("Years") #label on X_axis plt.ylabel("Numbers in '000") #label on
Y_axis plt.show() #showing the plot, finally
```