**DATA ANALYSIS AND VISUALIZATION LAB**

(Course Code: **22UPCSCSS01**)

A laboratory record submitted to Periyar University, Salem

in partial fulfillment of the requirements for the degree of

**MASTER OF SCIENCE**

in

**MATHEMATICS**

by

**NAME :**

**Reg. No:**

****

**DEPARTMENT OF COMPUTER SCIENCE**

**PERIYAR UNIVERSITY**

**PERIYAR PALKALAI NAGAR**

**SALEM - 636 011**

**(NOV 2022)**

CERTIFICATE

This is to certify that the Programming Lab entitled  
Data analysis and visualization (**22UPCSCSS01**) is a bonafide record work done by  
Mr./Ms. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Register Number \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ as partial  
fulfillment of the requirements for the degree of Master of Science in the Department of  
Mathematics, Periyar University, Salem, during the year 2022-2023.

Faculty In-Charge Head of the Department

Submitted for the practical examination held on \_\_\_\_\_\_\_\_\_\_

Internal Examiner External Examiner

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **DATE** | **TITLE OF THE PROGRAM** | **PAGE**  **No.** | **STAFF**  **SIGNATURE** |
| 01 |  | Basic Operators, Conditional Statements, Looping Statements and String Processing |  |  |
| 02 |  | Exploring and Analyzing of Data |  |  |
| 03 |  | Plot the data using Pygal libraries |  |  |
| 04 |  | Plot the data using Altair libraries |  |  |
| 05 |  | Plot the data using Bokeh libraries |  |  |
| 06 |  | Plot the data using PySimpleGUI libraries |  |  |
| 07 |  | Plot the data using Matplotlib libraries |  |  |
| 08 |  | Plot the data using Seaborn libraries |  |  |
| 09 |  | Plot the data using Plotly libraries |  |  |
| 10 |  | Plot the data using Geoplotlib libraries |  |  |

**CONTENTS**

|  |  |
| --- | --- |
| **EX. NO : 01** | **Basic Operators, Conditional Statements, Looping Statements and String Processing** |
| **DATE:** |

**AIM:**

**ALGORITHM:**

**SOURCE CODE:**

#Arithmetic Operators

x= 4

y= 5

print(x + y)

#Comparison Operators

print('x > y is',x>y)

#Assignment Operators

print ("Line 1 - Value of num1 : ", x)

print ("Line 2 - Value of num2 : ", y)

#compound assignment operator

res = x + y

res += x

print ("Line 1 - Result of + is ", res)

#Logical Operators

a = True

b = False

print('a and b is',a and b)

print('a or b is',a or b)

print('not a is',not a)

#Membership Operators

list = [1, 2, 3, 4, 5 ];

if ( x in list ):

print("Line 1 - x is available in the given list")

else:

print("Line 1 - x is not available in the given list")

if ( y not in list ):

print("Line 2 - y is not available in the given list")

else:

print("Line 2 - y is available in the given list")

#Identity Operators

if ( x is y ):

print("x & y have SAME identity")

if ( x is not y ):

print("x & y have DIFFERENT identity")

#Operator precedence

v = 4

w = 5

x = 8

y = 2

z = 0

z = (v+w) \* x / y;

print("Value of (v+w) \* x/ y is ", z)

# Control Statement

# While Loop

lines=['Hi','I am']

while True:

l =input()

if l:

lines.append(l.upper())

break;

print(lines)

else:

break;

print(lines)

print('looping completed')

# for loop

for l in lines:

print(l)

# elif loop

month=input("Input the month (e.g. January, February etc.): ")

day=int(input("Input the day: "))

if month in('January','February','March'):

season='winter'

elif month in('April','May','June'):

season='spring'

elif month in('July','August','September'):

season='summer'

else:

season ='autumn'

if(month =='March')and(day >19):

season='spring'

elif(month =='June')and(day >20):

season='summer'

elif(month =='September')and(day >21):

season='autumn'

elif(month =='December')and(day >22):

season='winter'

print("Season is",season)

#String operations

str1 = "Periyar University"

print(str1)

print(str1[0])

#String Slicing

print (str1[1:11:3]) #my\_string [start:stop:step]

#String Concatenation

str2 = "Computer Science"

strcon = str1 +' '+ str2 + '.'

print(strcon)

x = 3 \*"Hi!" #x=3\*str1

print(x)

print(str1.capitalize())

print(str2.lower())

print(str2.upper())

print(str1.title())

print(str1.swapcase())

print(str1.count('e'))

print(str2.find('c'))

print(str1.isalnum())

print(str2.isalpha())

print(str2.isdigit()) # all are string basic function

#strip()

str1.lstrip()

str2.rstrip()

str1.strip()

print(str2)

#join()

newstr = ' '.join(['We', 'are', 'Coders'])

print(newstr)

#split()

l = 'we are coders'.split()

l = 'we are coders'.split('e')

print(l)

**OUTPUT:**

9

x > y is False

Line 1 - Value of num1 : 4

Line 2 - Value of num2 : 5

Line 1 - Result of + is 13

a and b is False

a or b is True

not a is False

Line 1 - x is available in the given list

Line 2 - y is available in the given list

x & y have DIFFERENT identity

Value of (v+w) \* x/ y is 36.0

looping completed

Hi

I am

Input the month (e.g. January, February etc.): january

Input the day: 45

Season is autumn

Periyar University

P

ey i

Periyar University Computer Science.

Hi!Hi!Hi!

Periyar university

computer science

COMPUTER SCIENCE

Periyar University

pERIYAR uNIVERSITY

2

10

False

False

False

Computer Science

We are Coders

['w', ' ar', ' cod', 'rs']

**RESULT:**

|  |  |
| --- | --- |
| **EX. NO : 02** | **Exploring and Analyzing of Data** |
| **DATE:** |

**AIM:**

**ALGORITHM:**

**SOURCE CODE:**

#matplotlib inline

import numpy as np

import pandas as pd

import scipy.stats as stats

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.datasets import load\_boston

boston\_dst = load\_boston()

#import seaborn as sns

#iris = sns.load\_dataset('iris')

print("Type of the dataset: ", type(boston\_dst))

print("Keys: ", boston\_dst.keys())

print(boston\_dst.filename)

print(boston\_dst.DESCR)

print("Type of data: ", type(boston\_dst.data))

print("Shape of data: ", boston\_dst.data.shape)

print("Feature Names: ", boston\_dst.feature\_names)

print("Type of target: ", type(boston\_dst.target))

print("Shape of target: ", boston\_dst.target.shape)

boston\_pd = pd.DataFrame(boston\_dst.data)

boston\_pd.head()

boston\_pd.columns = boston\_dst.feature\_names

boston\_pd.head()

boston\_pd['PRICE'] = boston\_dst.target

boston\_pd.head()

#Exploratory Data Analysis

boston\_pd #Overview of the Data

boston\_pd.describe(include='all').transpose() #understand characteristics of the data

and to get a quick summary of it.

boston\_pd.PRICE.value\_counts() #get count of each category in a categorical attributed series of values.

print(boston\_pd.nunique()) #number of unique elements in each column.

boston\_pd[boston\_pd.isnull().any(axis=1)] #Display Rows with Missing Data

boston\_pd.isnull().sum() #Count the number of missing values for each column

total = boston\_pd.isnull().sum().sort\_values(ascending=False)

percent = (boston\_pd.isnull().sum()/boston\_pd.isnull().count()).sort\_values(ascending=False)

missing\_data = pd.concat([total, percent], axis=1,

keys=['Total', 'Percent'])

missing\_data.head(20)

boston\_pd.notna().sum() #count the number of non-missing data

boston\_pd = boston\_pd.dropna() #drop null values

boston\_pd.shape

boston\_pd = boston\_pd.drop((missing\_data[missing\_data['Total'] > 1]).index, 1)

boston\_pd = boston\_pd.drop(boston\_pd.loc[boston\_pd['CRIM'].isnull()].index)

boston\_pd.isnull().sum().max()

boston\_pd.AGE = boston\_pd.AGE.fillna(35)

boston\_pd.fillna(boston\_pd.mean())

#BoxPlot

plt.boxplot(list(boston\_pd.PRICE));

plt.show();

fig, ax = plt.subplots(len(list(boston\_pd.columns)), figsize=(8,40))

#one type

for i, feature\_name in enumerate(list(boston\_pd.columns)):

sns.boxplot(y=boston\_pd[feature\_name], ax=ax[i]);

ax[i].set\_xlabel(feature\_name, fontsize=8);

#ax[i].set\_title("Box plot {} ".format(feature\_name), fontsize=8);

plt.show();

#Scatterplot

for feature\_name in boston\_dst.feature\_names:

plt.figure(figsize=(5, 4));

plt.scatter(boston\_pd[feature\_name], boston\_pd['PRICE']);

plt.ylabel('Price', size=12);

plt.xlabel(feature\_name, size=12);

plt.show();

#implot

sns.lmplot(x = 'RM', y = 'PRICE', data = boston\_pd)

#Histogram

plt.figure(figsize=(8, 6));

plt.hist(boston\_pd['PRICE']);

plt.title('Boston Housing Prices and Count Histogram');

plt.xlabel('price ($1000s)');

plt.ylabel('count');

plt.show();

from scipy.stats import norm

sns.distplot(boston\_pd['PRICE'], fit=stats.norm);

plt.figure(figsize=(8, 6));

res = stats.probplot(boston\_pd['PRICE'], plot=plt);

fig, ax = plt.subplots(len(list(boston\_pd.columns)), figsize=(12,46))

#another one type

for i, feature\_name in enumerate(list(boston\_pd.columns)):

if (feature\_name != 'CHAS'):

sns.distplot(boston\_pd[feature\_name], hist=True, ax=ax[i]);

ax[i].set\_ylabel('Count', fontsize=8);

ax[i].set\_xlabel(" {}".format(feature\_name), fontsize=8);

#ax[i].set\_title("Freq dist "+feature\_name, fontsize=8);

plt.show();

#pairplot

sns.pairplot(boston\_pd); #easy one

#another one type

n = 4

for i in range(0, len(boston\_pd.columns), n):

sns.pairplot(data=boston\_pd,

x\_vars=boston\_pd.columns[i:i+n],

y\_vars=['PRICE']);

#Heatmap: Two-Dimensional Graphical Representation

plt.figure(figsize=(12, 9));

correlation\_matrix = boston\_pd.corr().round(2);

sns.heatmap(correlation\_matrix, cmap="YlGnBu", annot=True);

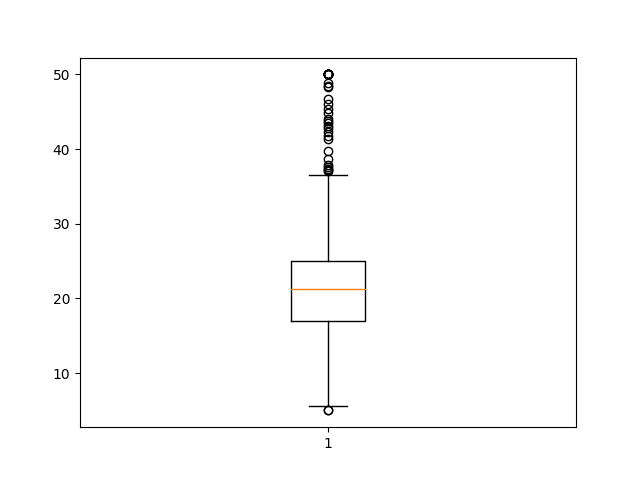
#heatmap

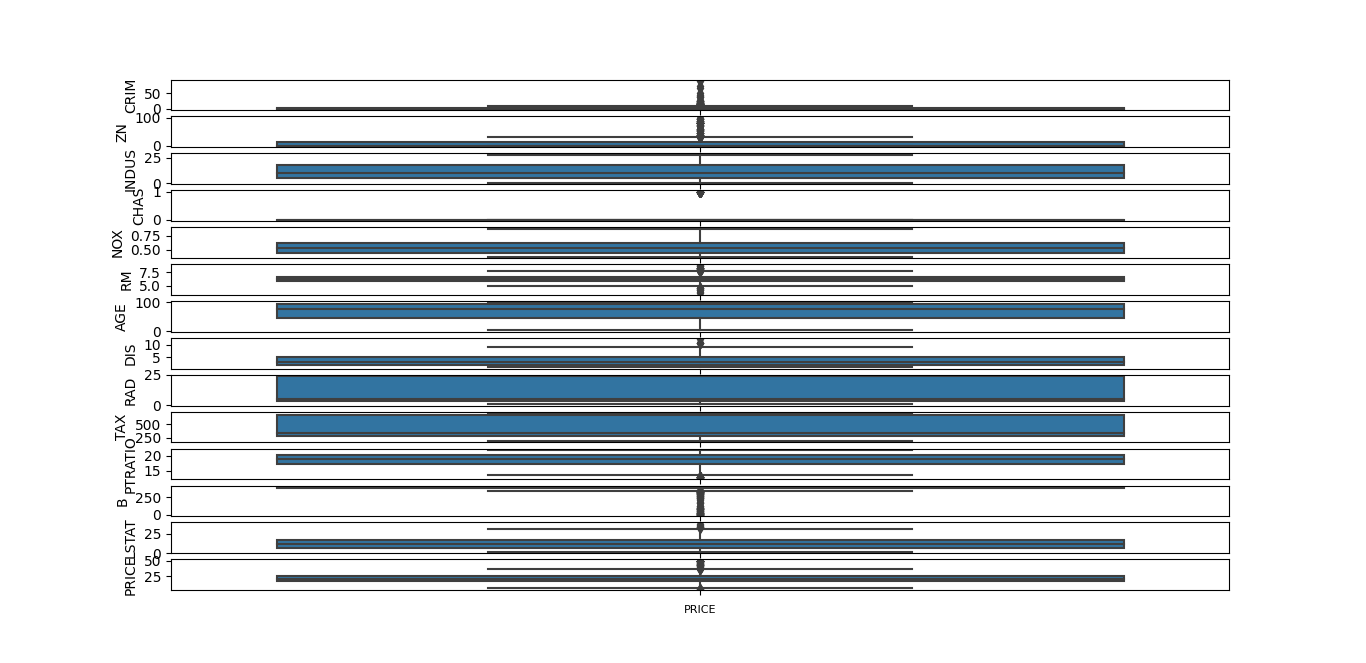
sns.heatmap(correlation\_matrix[(correlation\_matrix >= 0.5) | (correlation\_matrix <= -0.4)],

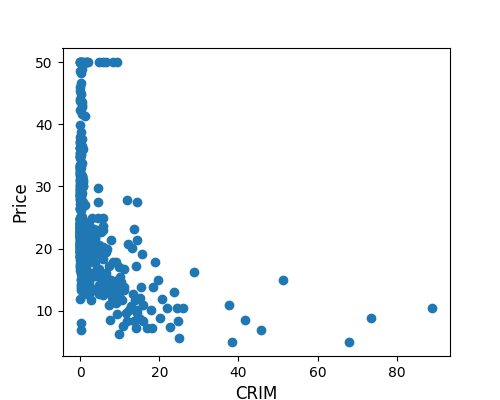
cmap='viridis', vmax=1.0, vmin=-1.0, linewidths=0.1,

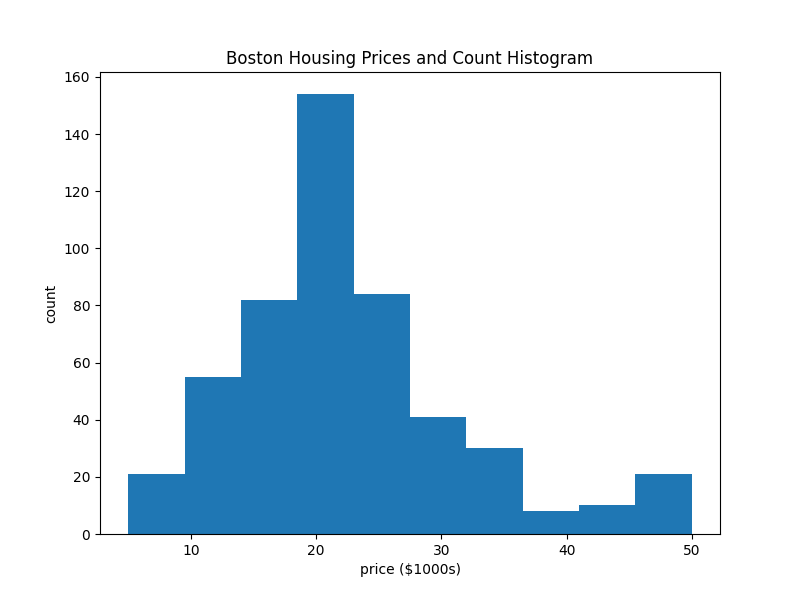
annot=True, annot\_kws={"size": 8}, square=True);

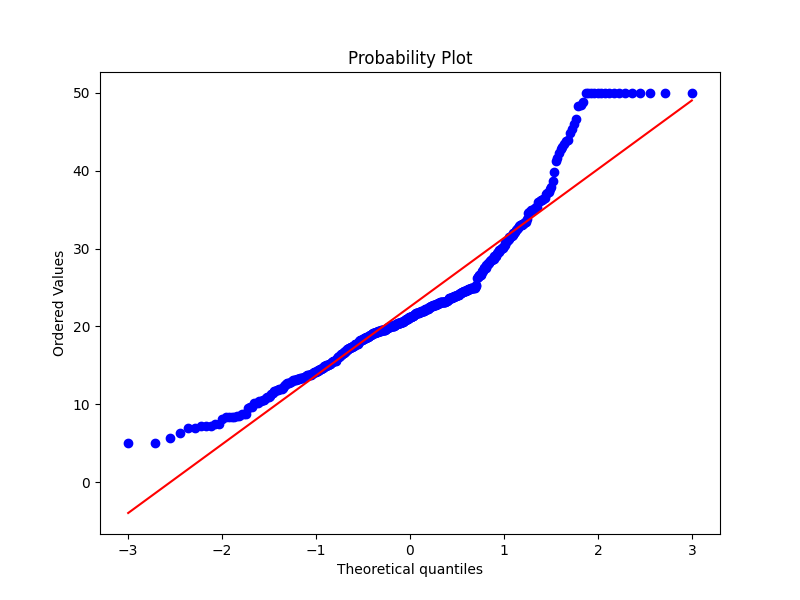
**OUTPUT:**

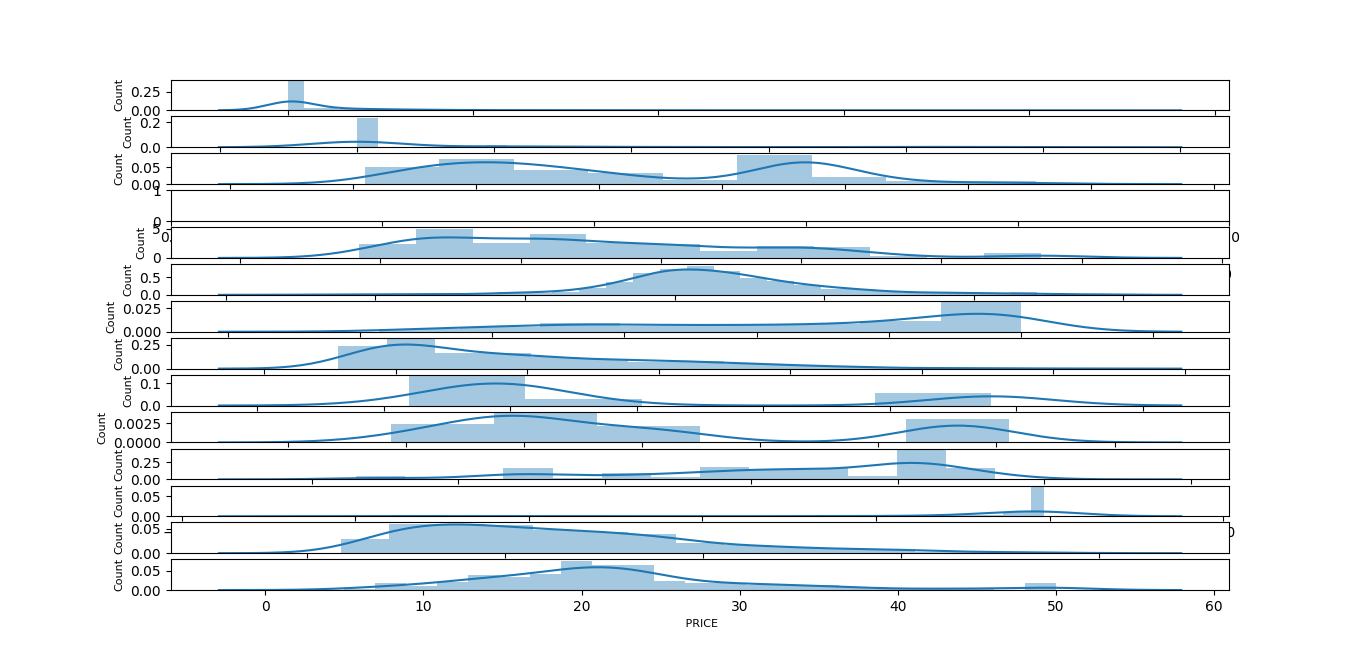






****

****

****

**RESULT:**

|  |  |
| --- | --- |
| **EX. NO : 03** | **Plot the Data using Pygal libraries** |
| **DATE:** |

**AIM:**

**ALGORITHM:**

**SOURCE CODE:**

import pygal

import seaborn as sns # this library for load dataset

df = sns.load\_dataset('tips')

#Bar chart

bar\_chart = pygal.Bar()

bar\_chart.add('Tip', df['tip'])

bar\_chart.render\_to\_file('bar\_chart1.svg')

#double Bar chart

bar\_chart.add('Tip', df['tip'][:10])

bar\_chart.add('Total Bill', df['total\_bill'][:10])

bar\_chart.render\_to\_file('bar\_chart2.svg')

#line chart

line\_chart = pygal.Line()

line\_chart.add('Total Bill', df['total\_bill'][:15])

line\_chart.render\_to\_file('line1.svg')

#double line chart

line\_chart.add('Total Bill', df['total\_bill'][:15])

line\_chart.add('Tips', df['tip'][:15])

line\_chart.render\_to\_file('line2.svg')

#Box plot chart

box\_plot = pygal.Box()

box\_plot.title = 'Tips Dataset'

box\_plot.add('Tips', df['tip'])

box\_plot.render\_to\_file('box1.svg')

#Funnel chart

funnel\_chart = pygal.Funnel()

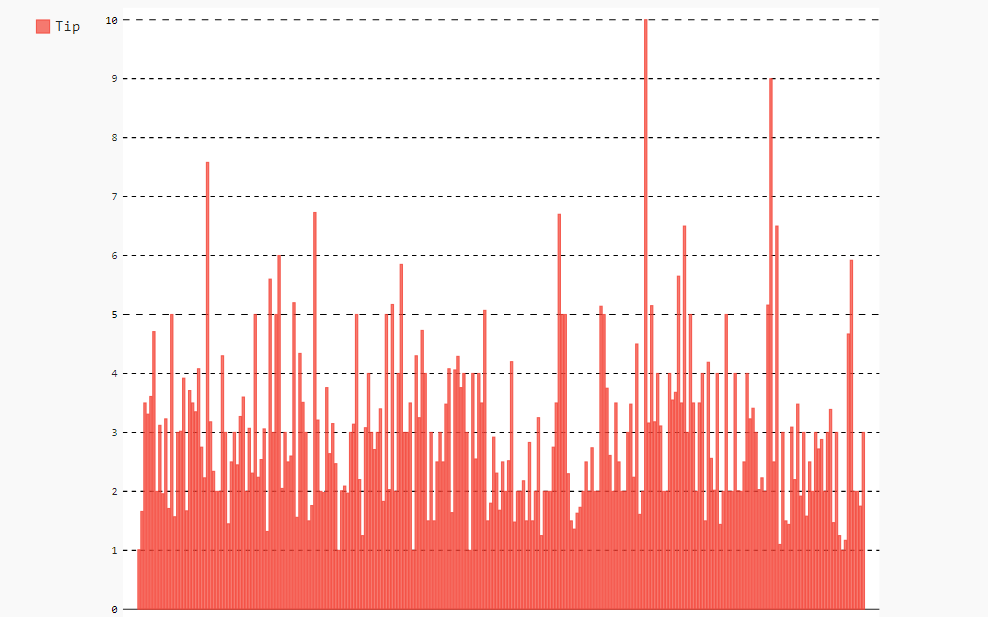
funnel\_chart.title = 'Total Bill'

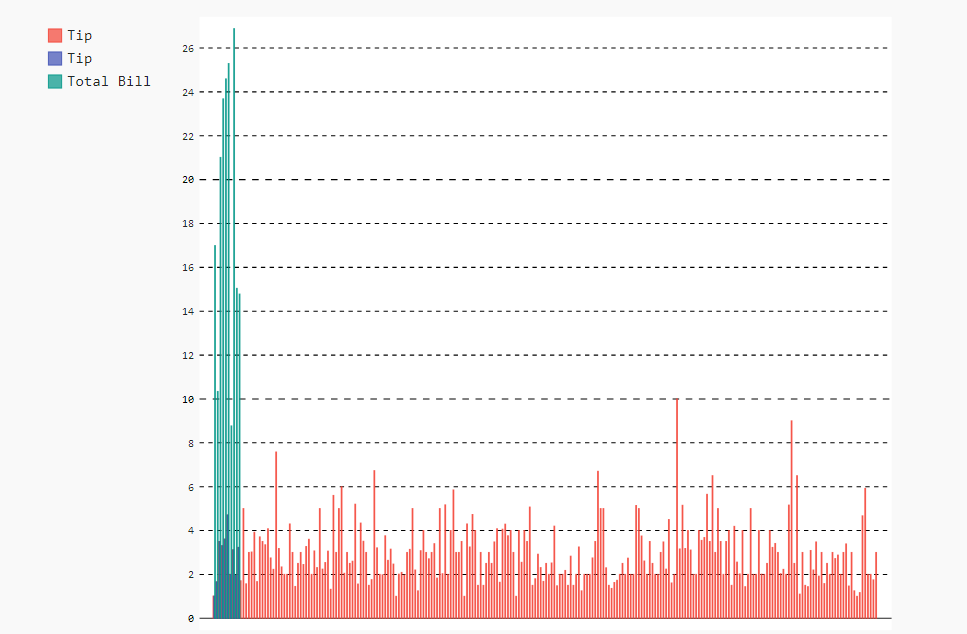
funnel\_chart.add('Total Bill', df['total\_bill'][:15])

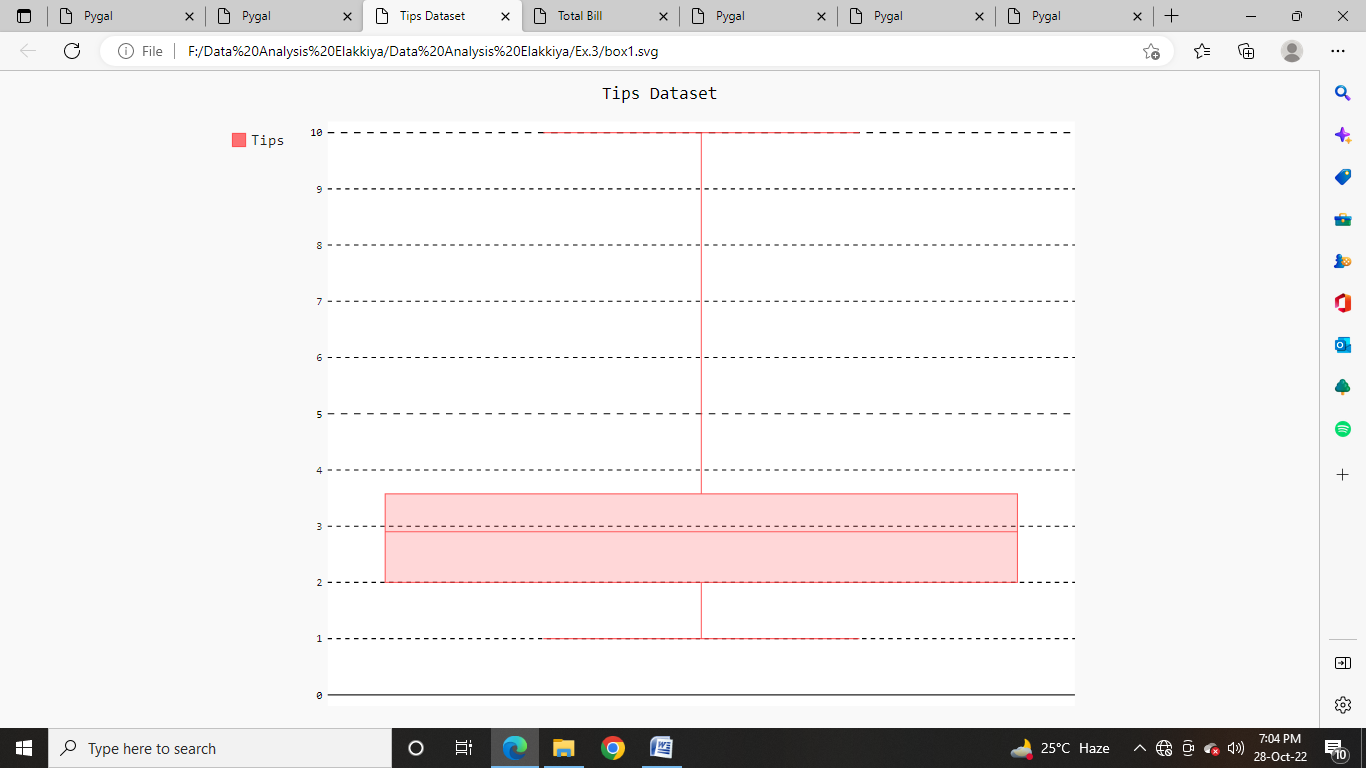
funnel\_chart.add('Tip', df['tip'][:15])

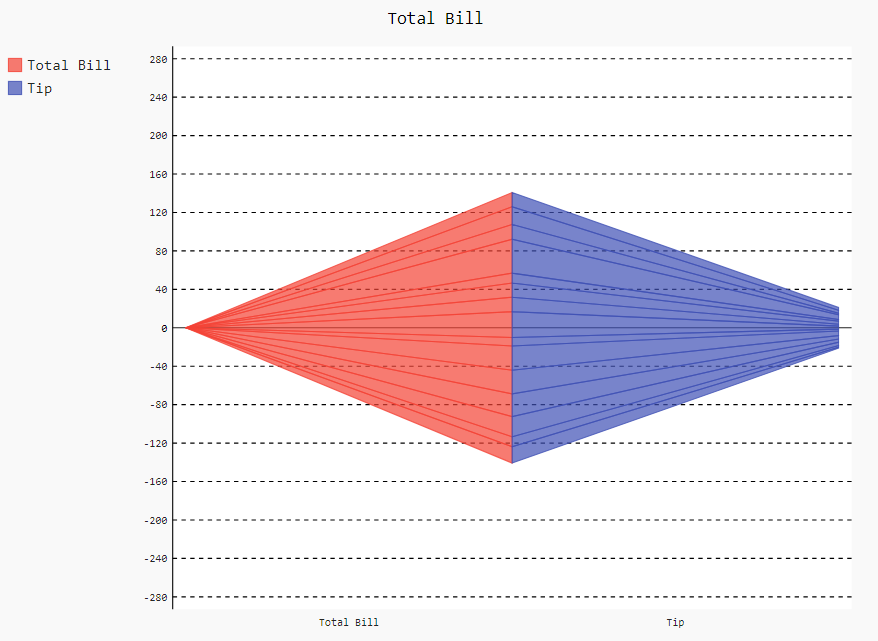
funnel\_chart.render\_to\_file('funnel1.svg')

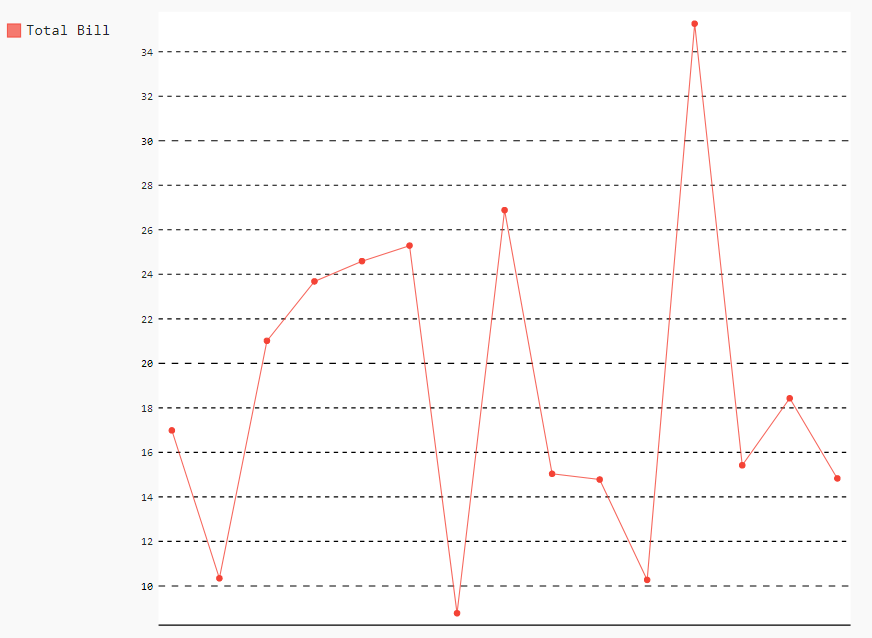
**OUTPUT:**

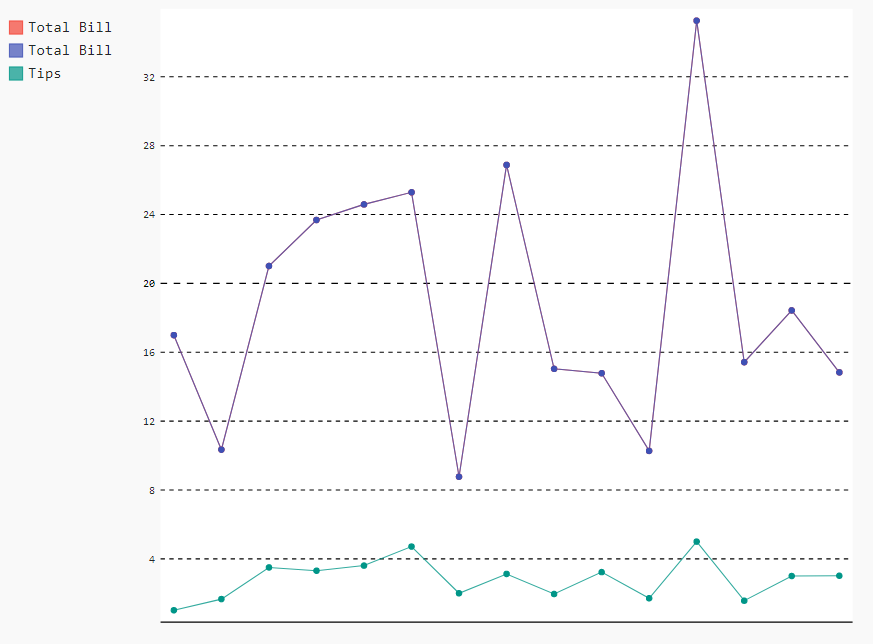


****

****

****

****

****

**RESULT:**

|  |  |
| --- | --- |
| **EX. NO : 04** | **Plot the Data using Altair libraries** |
| **DATE:** |

**AIM:**

**ALGORITHM:**

**SOURCE CODE:**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import altair as alt

df = sns.load\_dataset('mpg') #importing dataset

df.shape

df.keys()

#Scatter chart

chart1=alt.Chart(df).mark\_point().encode(alt.Y('mpg'),alt.X('horsepower'),alt.Color('origin'),alt.OpacityValue(0.7),size='displacement')

chart1.save('altairchart1.html')

#Bubble chart

chart4=alt.Chart(df).mark\_point(filled=True).encode(x='horsepower',y='mpg', size='displacement',color='origin')

chart4.save('altairchart4.html')

#Line chart

chart2=alt.Chart(df).mark\_line().encode(alt.X('horsepower'),alt.Y('acceleration'), alt.Color('origin'))

chart2.save('altairchart2.html')

#bar chart

plot=alt.Chart(df).mark\_bar(size=40).encode(alt.X('cylinders'),alt.Y('mpg'),

alt.Color('origin'))

plot.properties(title='cylinders vs mpg')

plot.save('altairchart4.html')

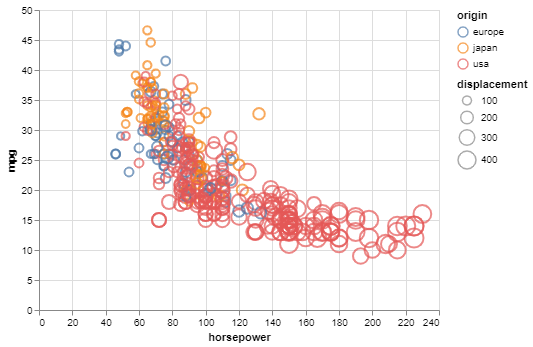
#Strip plots

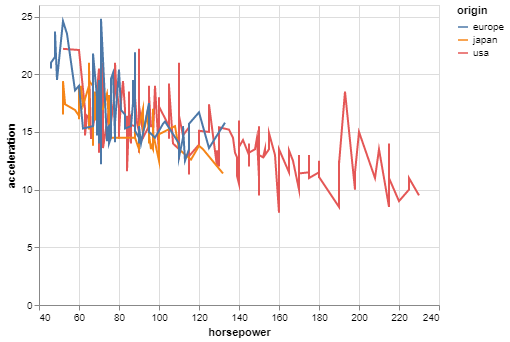
chart3=alt.Chart(df).mark\_tick(filled=True).encode(x='horsepower:Q',y='cylinders:O',

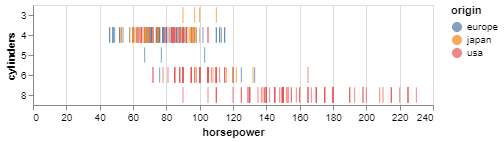
color='origin')

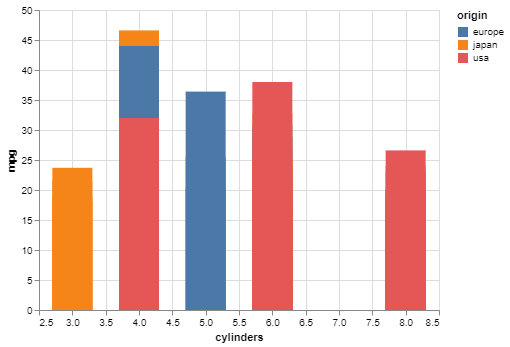
chart3.save('altairchart3.html')

**OUTPUT:**









**RESULT:**

|  |  |
| --- | --- |
| **EX. NO : 05** | **Plot the Data using Bokeh libraries** |
| **DATE:** |

**AIM:**

**ALGORITHM:**

**SOURCE CODE:**

import seaborn as sns

from bokeh.palettes import GnBu5

from bokeh.plotting import figure, show,output\_file

import pandas as pd

import numpy as np

from bokeh.palettes import Spectral7

from bokeh.models import Title

from bokeh.transform import linear\_cmap

from bokeh.models import ColorBar, ColumnDataSource

from bokeh.palettes import Spectral6

df = sns.load\_dataset('diamonds')

data = df.sample(53940, random\_state=34500)

#data plotting bubble chat

y = list(data.price.values)

x = list(data.carat.values)

mapper=linear\_cmap(field\_name="y",palette=Spectral6,low=min(y),high=max(y))

source = ColumnDataSource(dict(x=x,y=y))

p = figure(width=800, height=400)

p.circle(x='x', y='y', line\_color=mapper, color=mapper, fill\_alpha=1, size=12,

source=source)

color\_bar = ColorBar(color\_mapper=mapper['transform'], height=300, width=10)

p.add\_layout(color\_bar, 'right')

output\_file('bohakbubble.html')

# data plotting barchat

from bokeh.palettes import Spectral7

from bokeh.models import Title

# prepare the colors and their value counts

colors = sorted(list(data.color.unique()))

counts = [i for i in data.color.value\_counts().sort\_index()]

p = figure(x\_range=colors, width=800, height=400)

p.vbar(x=colors, top=counts, width=0.9, color=Spectral7)

p.y\_range.start = 0

p.add\_layout(Title(text="Colors", align="center"), "below")

p.add\_layout(Title(text="Color counts", align="center"), "left")

output\_file('bohakcolorbar.html')

# interactive bar chat

colors = list(data.color.unique())

ideal = [data[(data.cut == "Ideal") & (data.color == colors[i])].shape[0] for i in range(len(colors))]

very\_good = [data[(data.cut == "Very Good") & (data.color == colors[i])].shape[0] for i in range(len(colors))]

premium = [data[(data.cut == "Premium") & (data.color == colors[i])].shape[0] for i in range(len(colors))]

good = [data[(data.cut == "Good") & (data.color == colors[i])].shape[0] for i in range(len(colors))]

fair = [data[(data.cut == "Fair") & (data.color == colors[i])].shape[0] for i in range(len(colors))]

cut = list(data.cut.unique())

data\_stacked = {'colors': colors,

'Ideal': ideal,

'Very Good': very\_good,

'Premium': premium,

'Good': good,

'Fair': fair}

p = figure(x\_range=colors, width=800, height=400, title="colors counts by cut",

toolbar\_location=None, tools="hover")

p.vbar\_stack(cut, x='colors', width=0.9, color=GnBu5, source=data\_stacked,

legend\_label=cut)

p.y\_range.start = 0

p.y\_range.end = 1000

p.legend.location = "top\_left"

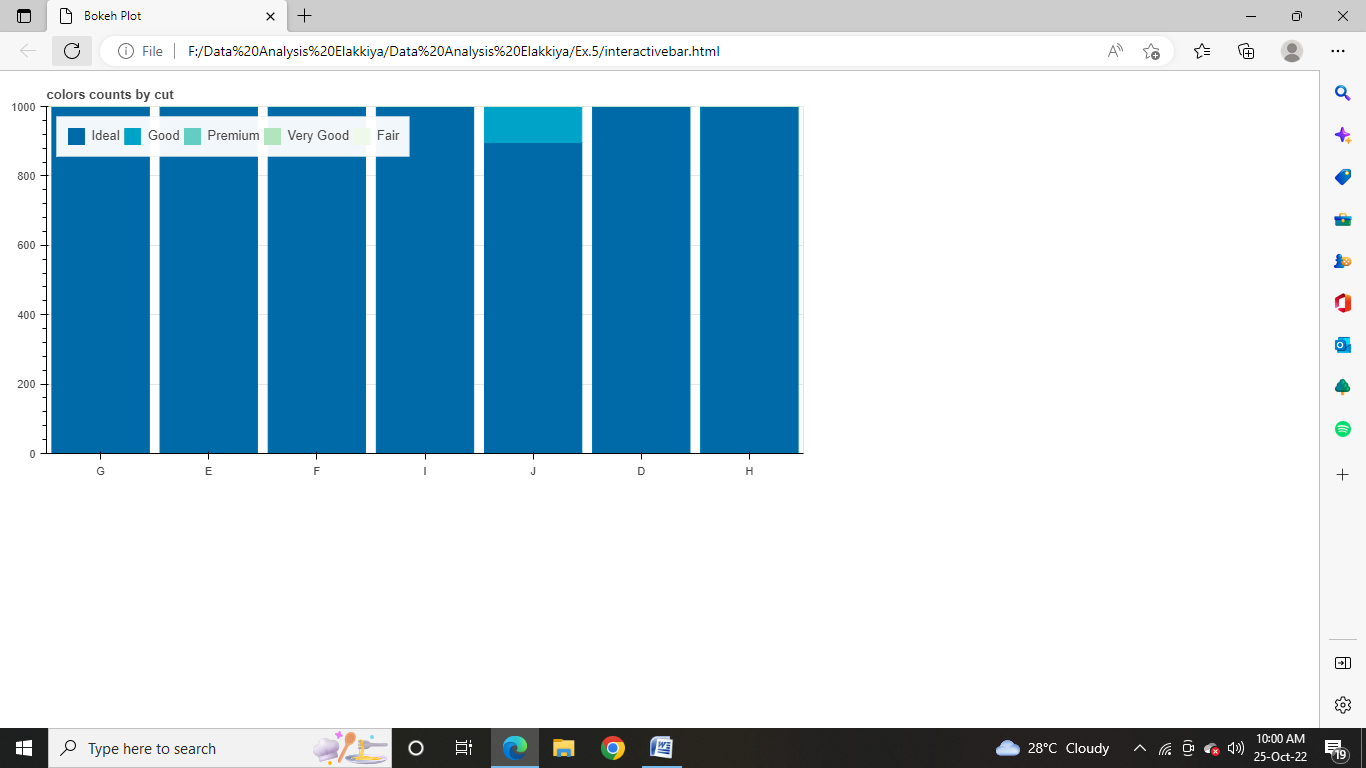
p.legend.orientation = "horizontal"

p.legend.click\_policy="hide"

output\_file('interactivebar.html')

show(p)

**OUTPUT:**



**RESULT:**

|  |  |
| --- | --- |
| **EX. NO : 06** | **Plot the Data using PySimpleGUI libraries** |
| **DATE:** |

**AIM:**

**ALGORITHM:**

**SOURCE CODE:**

import PySimpleGUI as sg

import seaborn as sns

mpg=sns.load\_dataset("mpg")

tips=sns.load\_dataset("tips")

bar\_chart=sns.barplot(x='day',y='total\_bill',data=tips)

bar\_chartfig=bar\_chart.get\_figure()

bar\_chartfig.savefig('pusimpleguibarfig.png')

line\_chart=sns.boxplot(y=mpg["mpg"])

line\_chartfig=line\_chart.get\_figure()

line\_chartfig.savefig('pusimpleguilinefig.png')

barfig\_column=[[sg.Image("pusimpleguibarfig.png")]]

linefig\_column=[[sg.Image("pusimpleguilinefig.png")]]

layout=[

[sg.Column(barfig\_column),

sg.VSeperator(),

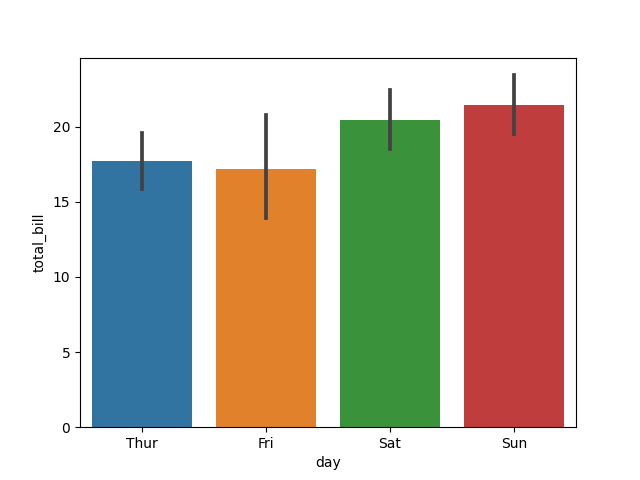
sg.Column(linefig\_column),

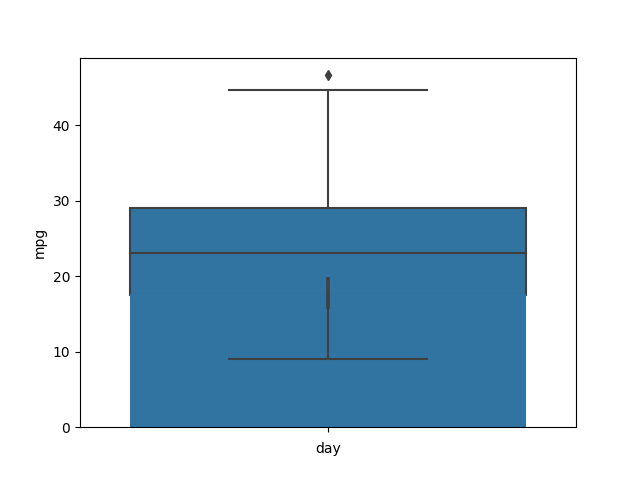
]

]

sg.Window(title='MyWindow',layout=layout).read()

**OUTPUT:**





**RESULT:**

|  |  |
| --- | --- |
| **EX. NO : 07** | **Plot the Data using Matplotlib libraries** |
| **DATE:** |

**AIM:**

**ALGORITHM:**

**SOURCE CODE:**

from sklearn import datasets

import matplotlib.pyplot as plt

import numpy as np

iris = datasets.load\_iris() # load dataset

# scatter plots

x\_iris = iris.data[:, :2] # only take the first two features

y\_iris = iris.target

n\_classes = 3

for i in range(n\_classes):

index = np.where(y\_iris == i)

plt.scatter(x\_iris[index, 0], x\_iris[index, 1],

label=iris.target\_names[i])

plt.legend()

plt.xlabel(iris.feature\_names[0])

plt.ylabel(iris.feature\_names[1])

plt.title('scatter chart Iris average')

plt.show

plt.savefig('matscatter.png')

# bar chart

X\_iris = iris.data

Y\_iris = iris.target

n\_classes = 3

averages = [X\_iris[Y\_iris == i].mean(axis=0) for i in range(n\_classes)]

x = np.arange(len(iris.feature\_names))

fig = plt.figure()

ax = fig.add\_subplot()

bar1 = ax.bar(x - 0.25, averages[0], 0.25, label=iris.target\_names[0])

bar2 = ax.bar(x, averages[1], 0.25, label=iris.target\_names[1])

bar3 = ax.bar(x + 0.25, averages[2], 0.25, label=iris.target\_names[2])

ax.set\_xticks(x)

ax.set\_xticklabels(iris.feature\_names)

plt.legend()

plt.title("Bar Chart Iris Averages")

plt.ylabel("Average")

plt.show

plt.savefig('matbarchart.png')

#Histogram

fig, axs = plt.subplots(2, 2)

axs[0, 0].hist(X\_iris[:, 0])

axs[0, 1].hist(X\_iris[:, 1], color='orange')

axs[1, 0].hist(X\_iris[:, 2], color='green')

axs[1, 1].hist(X\_iris[:, 3], color='red')

i = 0

for ax in axs.flat:

ax.set(xlabel=iris.feature\_names[i], ylabel='Frequency')

i += 1

fig.suptitle("Iris Histograms")

fig.savefig('mathisto.png')

#box plot

plt.boxplot(X\_iris, labels=[iris.feature\_names[0], iris.feature\_names[1],

iris.feature\_names[2], iris.feature\_names[3]])

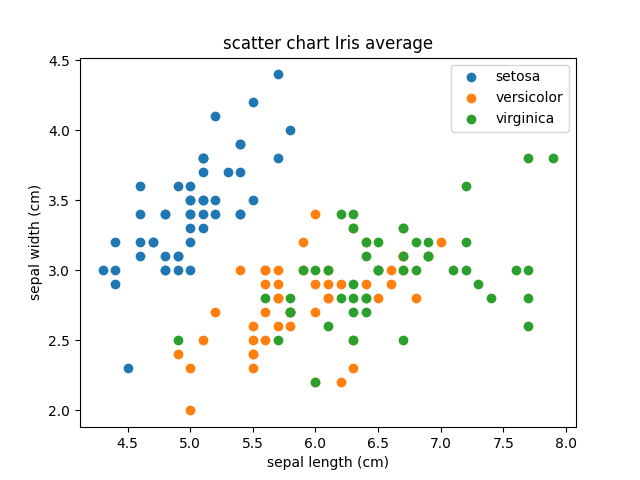
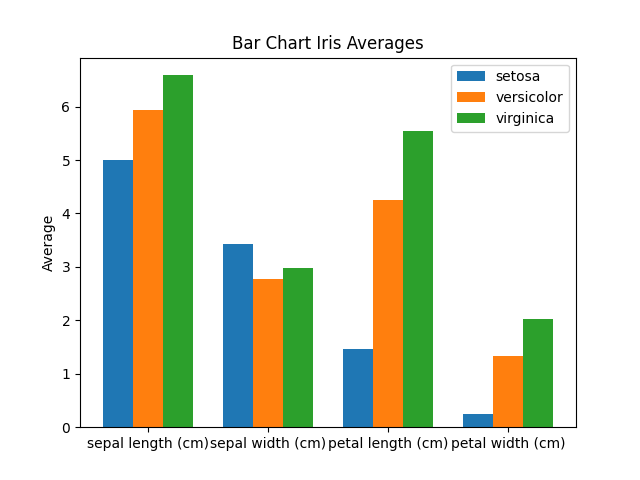
plt.title("Boxplots Iris features")

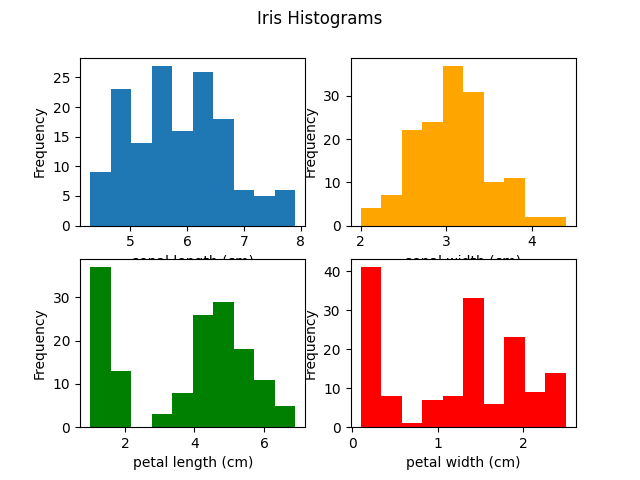
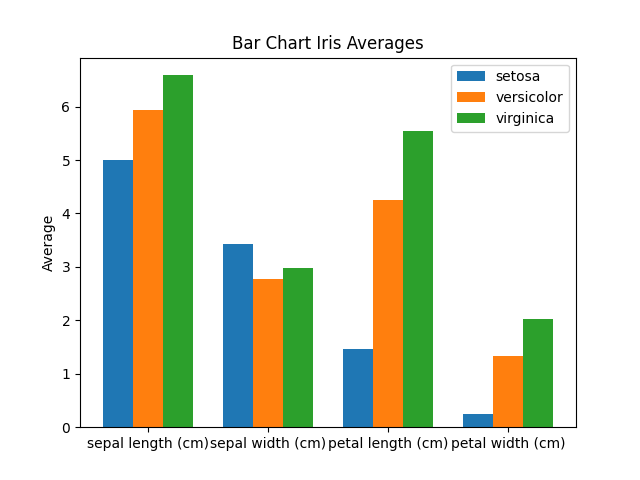
plt.ylabel("cm")

plt.show

plt.savefig('matboxplot.png')

**OUTPUT:**

**RESULT:**

|  |  |
| --- | --- |
| **EX. NO : 08** | **Plot the Data using Seaborn libraries** |
| **DATE:** |

**AIM:**

**ALGORITHM:**

**SOURCE CODE:**

import seaborn as sns

import matplotlib.pyplot as plt

mpg = sns.load\_dataset("mpg")

tips= sns.load\_dataset("tips")

iris = sns.load\_dataset("iris")

fmri = sns.load\_dataset("fmri")

# boxplot

sns.boxplot(y = mpg["mpg"])

plt.savefig("seabornbox.png")

# scatterplot

sns.scatterplot(x = "sepal\_length", y = "petal\_length", data = iris)

plt.savefig("seabornscatter.png")

# trend line

sns.lmplot(x = "sepal\_length", y = "petal\_length", data = iris, scatter = False)

plt.savefig("seaborntrend.png")

# bar chart

sns.barplot(x = "day", y = "total\_bill", data = tips)

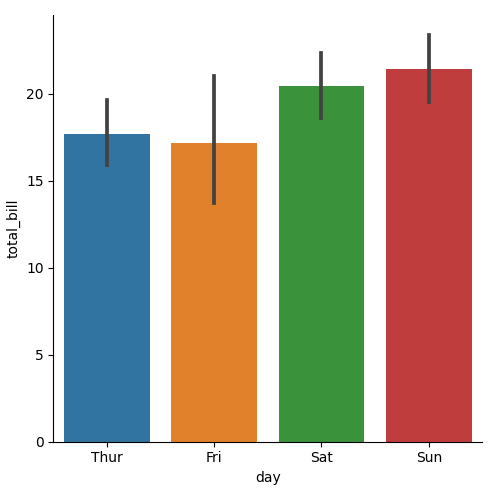
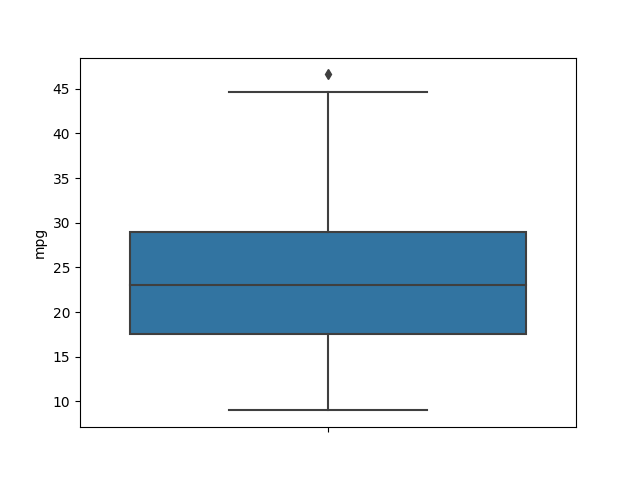
plt.savefig("seabornbar.png")

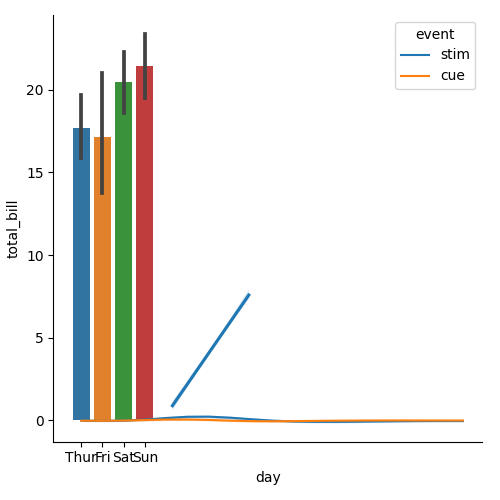
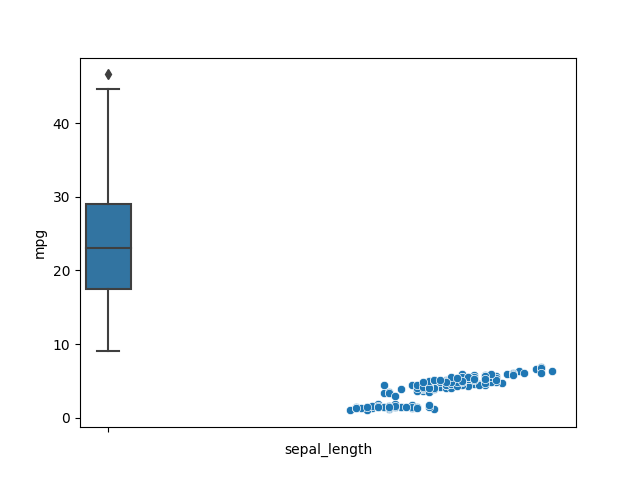
# line chart/time series

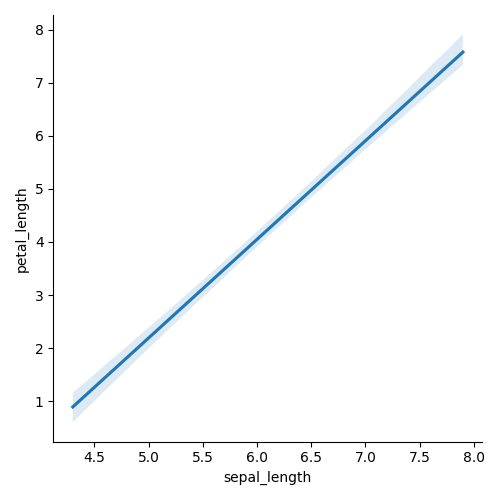
sns.lineplot(x="timepoint", y="signal", hue="event", data=fmri)

plt.savefig("seabornline.png")

**OUTPUT:**

 ****

** **

****

**RESULT:**

|  |  |
| --- | --- |
| **EX. NO : 09** | **Plot the Data using Plotly libraries** |
| **DATE:** |

**AIM:**

**ALGORITHM:**

**SOURCE CODE:**

import numpy as np

import pandas as pd

import plotly.express as px

import plotly.graph\_objects as go

import seaborn as sns

import statsmodels.api as sm

df = sns.load\_dataset('iris')

fig = px.scatter(df, x="sepal\_width", y="sepal\_length", color='species')

fig.show()

fig.write\_html("plotlyscater1.html")

fig = px.scatter(df, y="petal\_length", x="petal\_width", color="species", symbol="species")

fig.update\_traces(marker\_size=10)

fig.write\_html("plotlyscater2.html")

#scatter plot

tips = sns.load\_dataset('tips')

fig = px.scatter(tips, x="total\_bill", y="tip", trendline="ols")

fig.show()

fig.write\_html("plotlyscater3.html")

#line chart

stk = sns.load\_dataset('stock')

fig = px.line(stk, x='date', y=["MSFT","GOOG",'FB',"AMZN"])

fig.show()

fig.write\_html("plotlyline1.html")

# Line plot

gm = px.data.gapminder().query("continent == 'Oceania'")

fig = px.line(gm, x='year', y='pop', color='country',markers=True)

fig.show()

fig.write\_html("plotlyline2.html")

#combined plots

N=100

random\_x = np.linspace(0, 5, N)

random\_y0 = np.random.randn(N) + 5

random\_y1 = np.random.randn(N)

random\_y2 = np.random.randn(N) - 5

fig = go.Figure()

# Add traces

fig.add\_trace(go.Scatter(x=random\_x, y=random\_y0,

mode='lines+markers',

name='lines+markers'))

fig.add\_trace(go.Scatter(x=random\_x, y=random\_y1,

mode='markers',

name='markers'))

fig.add\_trace(go.Scatter(x=random\_x, y=random\_y2,

mode='lines',

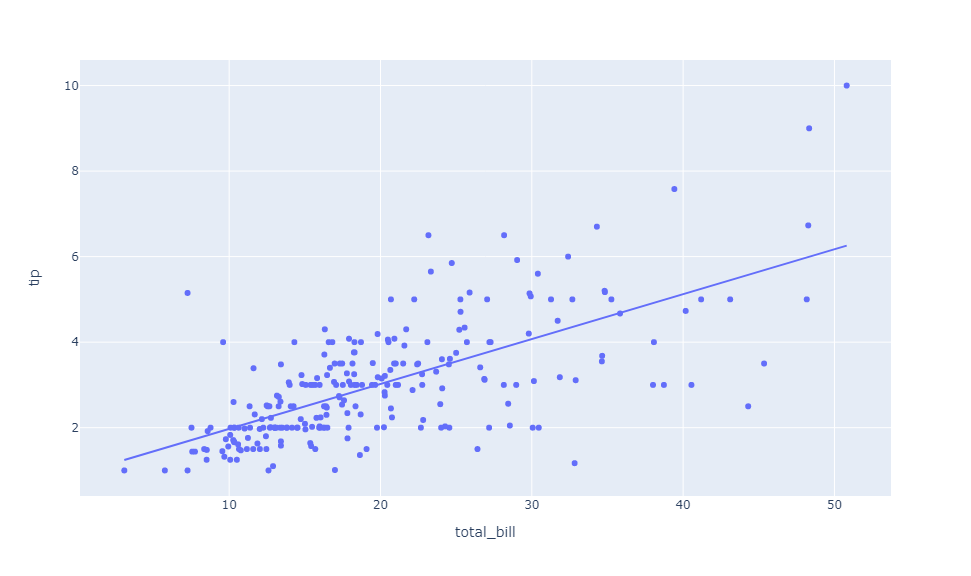
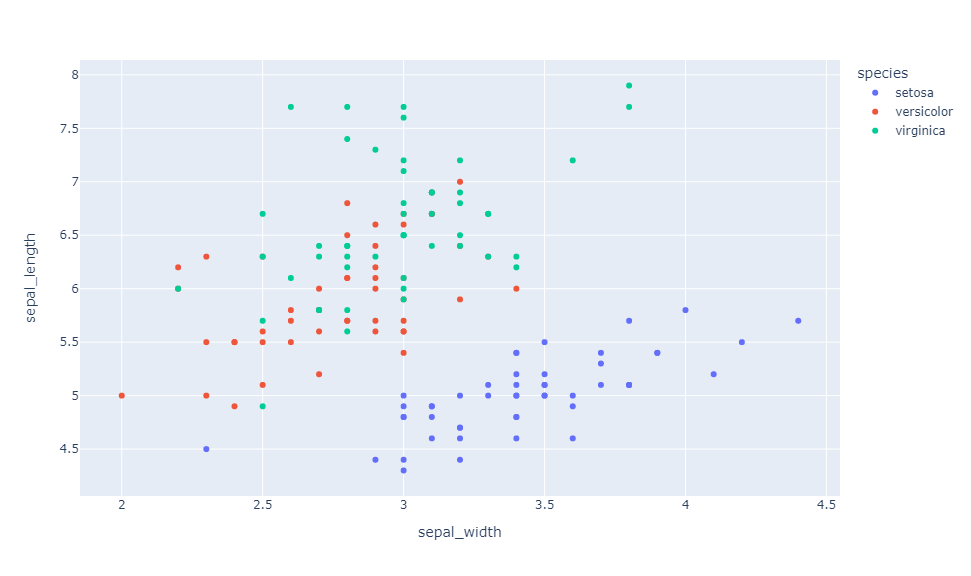
name='lines'))

fig.show()

fig.write\_html("plotlyinteraction.html")

**OUTPUT:**





**RESULT:**

|  |  |
| --- | --- |
| **EX. NO : 10** | **Plot the Data using Geoplotlib libraries** |
| **DATE:** |

**AIM:**

**ALGORITHM:**

**SOURCE CODE:**

#import required packages

from geoplotlib.utils import read\_csv, BoundingBox, DataAccessObject

import geoplotlib

import pandas as pd

#dot map

data = read\_csv('bus.csv')

geoplotlib.dot(data)

geoplotlib.show()

# Heat Map

df = pd.read\_csv('flights.csv')

geoplotlib.kde(df, bw=7, cut\_below=1e-4)

geoplotlib.set\_bbox(BoundingBox.KBH)

geoplotlib.show()

#histogram

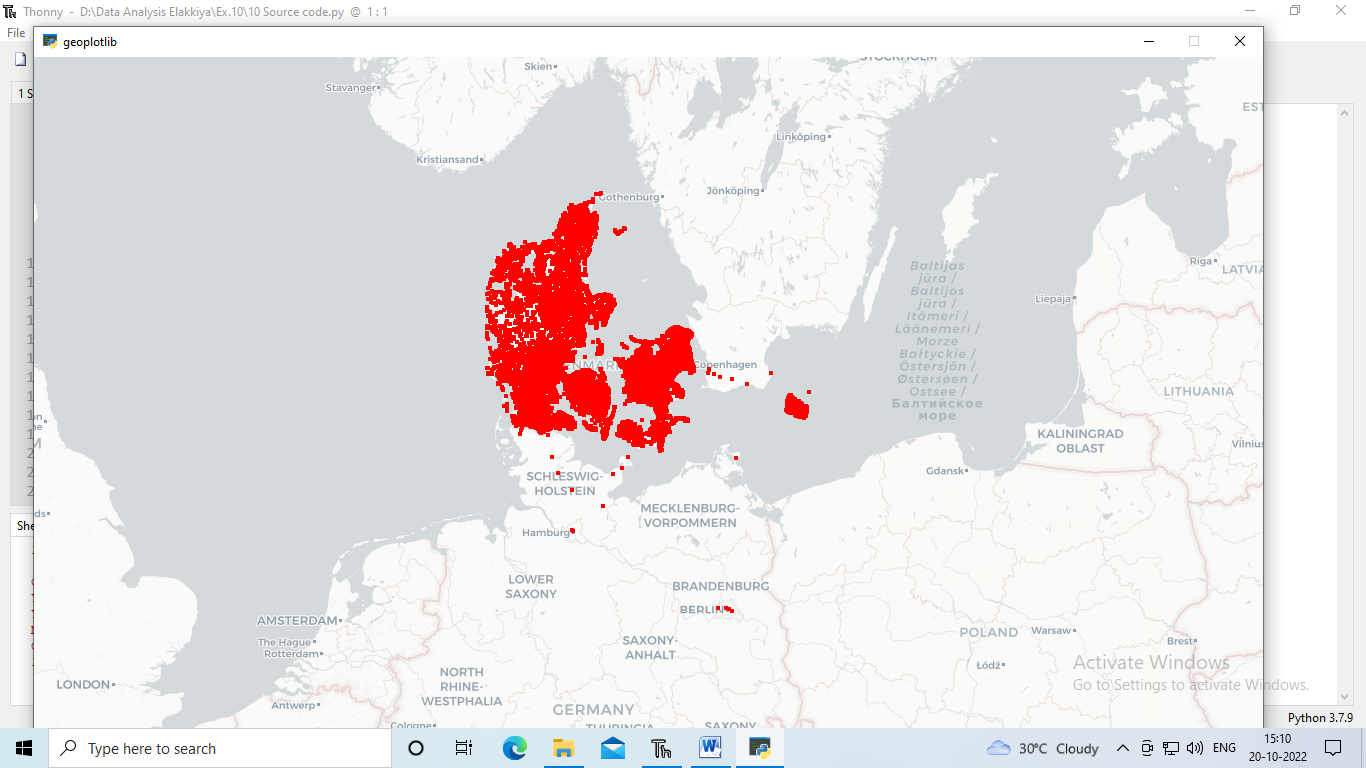
data1 = read\_csv('metro.csv')

geoplotlib.hist(data1, colorscale='sqrt', binsize=8)

geoplotlib.set\_bbox(BoundingBox.DK)

geoplotlib.show()

**OUTPUT:**



**RESULT:**