Lab2.1_Plotting 2D graphs(class2)

January 19, 2022

[1]:
```
from datetime import date
today=date.today()
print("Today's date:",today)
```

```
Today's date: 2021-12-31
```

1 Plots- 2D and 3D, Graph Customization

1.1 Table of Content

1. **Plots using Matplotlib**
2. **Plots using Seaborn**

There are different visualization libraries in python that provides an interface for drawing various graphics. Some most widely used libraries are Matplotlib and Seaborn.

Import the required libraries

[5]:
```
# import the library NumPy
import numpy as np

# import the library Pandas
import pandas as pd

# import the library matplotlib
import matplotlib.pyplot as plt

# import thelibrary seaborn
import seaborn as sns

# to suppress warnings
# import the library warnings
import warnings
# filterwarnings() filters a warning
# action: the action to be taken if a warning is shown
warnings.filterwarnings(action = 'ignore')
```

Seaborn library provides a variety of datasets. Plot different visualization plots using various libraries for the 'tips' dataset.

[34]:
```
# load the 'tips' dataset from seaborn
df_tips = sns.load_dataset('tips')

# display head() of the dataset
df_tips.head()
```

[34]:
```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female     No  Sun  Dinner     4
```

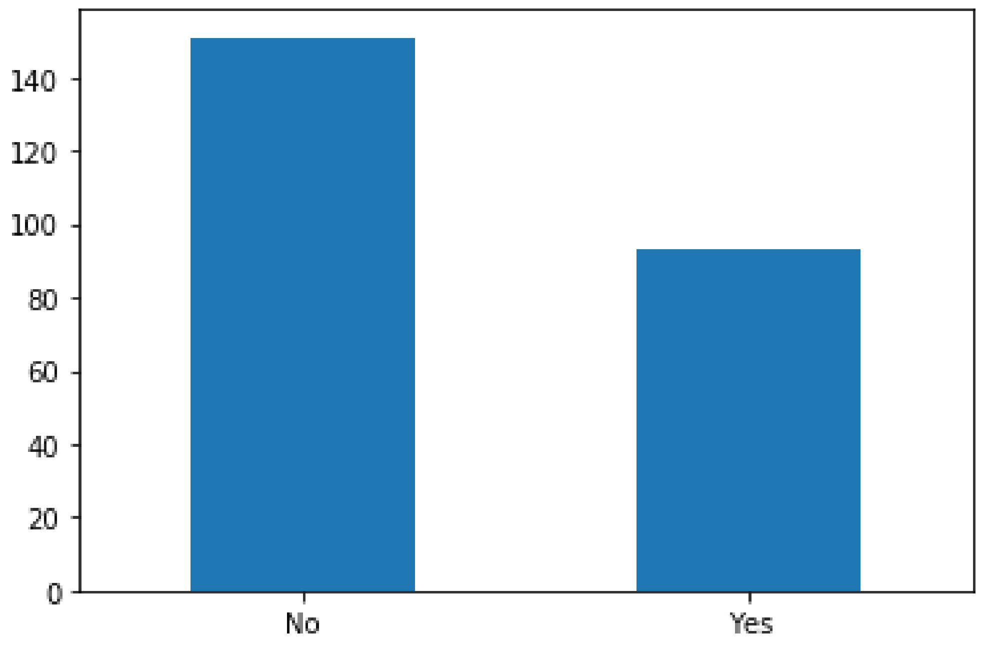
1.1.1 1.3 Bar Plot

To display categorical data use a bar plot with bars having lengths proportional to the values that they represent. The comparison between different categories of a categorical variable can be made by studying a bar plot. In the vertical bar plot, the X-axis displays the categorical variable, and Y-axis contains the values corresponding to different categories.

[41]:
```
# the variable 'smoker' is categorical
# check categories in the variable
set(df_tips['smoker'])
```

[41]:
```
{'No', 'Yes'}
```

[42]:
```
# bar plot to get the count of smokers and non-smokers in the data

# kind='bar' plots a bar plot
# 'rot = 0' returns the categoric labels horizontally
df_tips.smoker.value_counts().plot(kind='bar', rot = 0)

# display the plot
plt.show()
```



Let us add the count of smokers and non-smokers, axes labels and title to the above plot.

1.1.2 1.4 Pie Plot

Pie plot is a graphical representation of univariate data. It is a circular graph divided into slices displaying the numerical proportion. For the categorical variable, each slice of the pie plot corresponds to each of the categories.

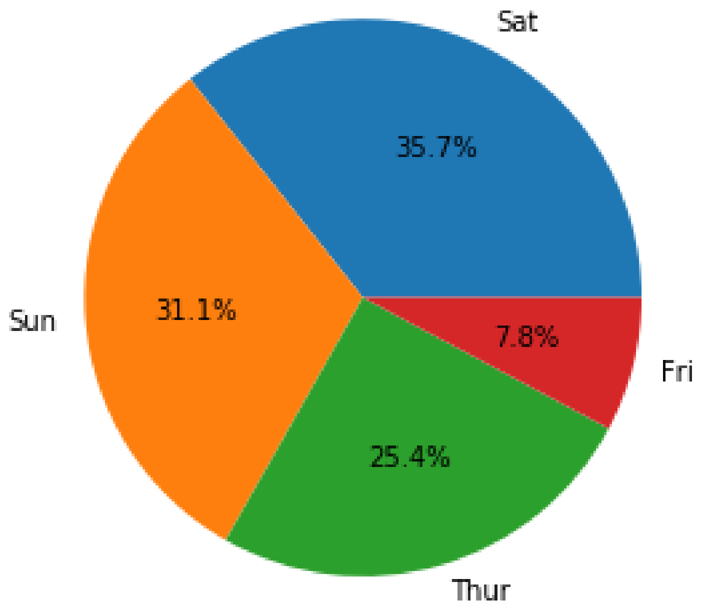
[44]:
```
# check the head() of the tips dataset
df_tips.head()
```

[44]:
```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female     No  Sun  Dinner     4
```

[45]:
```
# categories in the 'day' variable
df_tips.day.value_counts()
```

[45]:
```
Sat     87
Sun     76
Thur    62
Fri     19
Name: day, dtype: int64
```

[46]:
```
# plot the occurrence of different days in the dataset

# 'autopct' displays the percentage upto 1 decimal place
# 'radius' sets the radius of the pie plot
# 'labels' are the labels given to each section
plt.pie(df_tips.day.value_counts(), autopct = '%.1f%%', radius = 1.2, labels =␣
 ↪['Sat', 'Sun','Thur','Fri'])

# display the plot
plt.show()
```

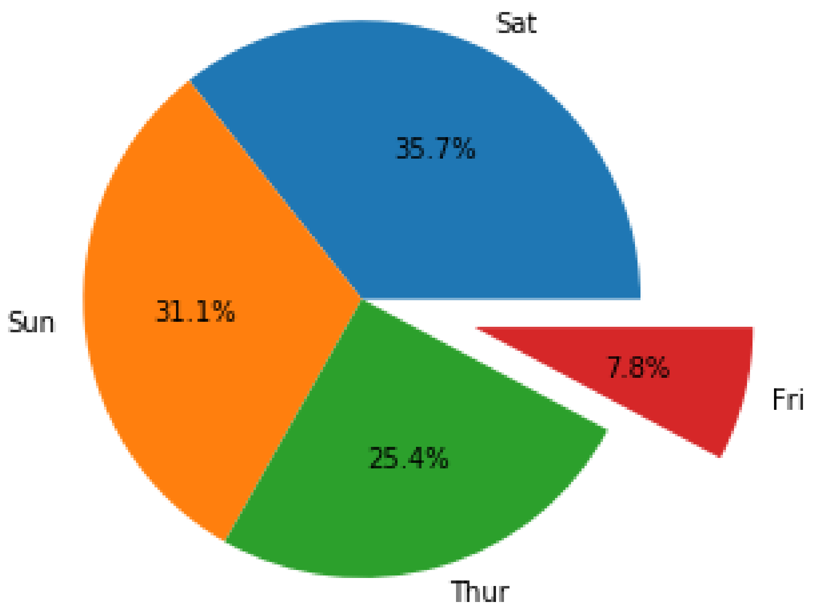


From the above pie plot, we see that the data has a high proportion for Saturday followed by Sunday.

Exploded pie plot is a plot in which one or more sectors are separated from the disc

[47]:
```
# plot the occurrence of different days in the dataset

# 'autopct' displays the percentage upto 1 decimal place
# 'radius' sets the radius of the pie plot
# 'labels' are the labels given to each section
# 'explode' specifies the radius with which to offset each section
plt.pie(df_tips.day.value_counts(), autopct = '%.1f%%', radius = 1.2, labels =␣
 ↪['Sat', 'Sun','Thur','Fri'],
        explode = [0,0,0,0.5])

# display the plot
```

```
plt.show()
```



Donut pie plot is a type of pie plot in which there is a hollow center representing a doughnut.

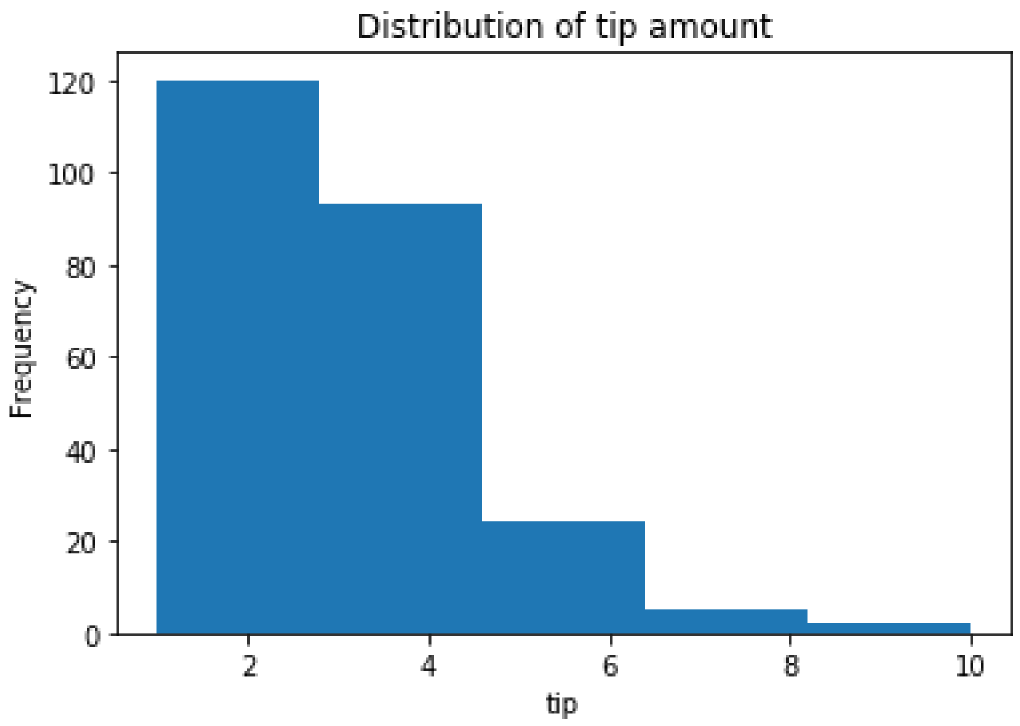
1.1.3 1.5 Histogram

A histogram is used to display the distribution and spread of the continuous variable. One axis represents the range of variable, and the other axis shows the frequency of the data points. In a histogram, there are no gaps between the bars.

In tips dataset, 'tip' is the continuous variable. Let us plot the histogram to understand the distribution of the variable.

[49]:
```
# plot the histogram
# specify the number of bins, using 'bins' parameter
plt.hist(df_tips['tip'], bins= 5)

# add the graph title
plt.title('Distribution of tip amount')

# add th x-axis label
plt.xlabel('tip')

# # add th y-axis label
plt.ylabel('Frequency')

# display the plot
```

```
plt.show()
```



From the above plot, we can see that the tip amount is positively skewed.

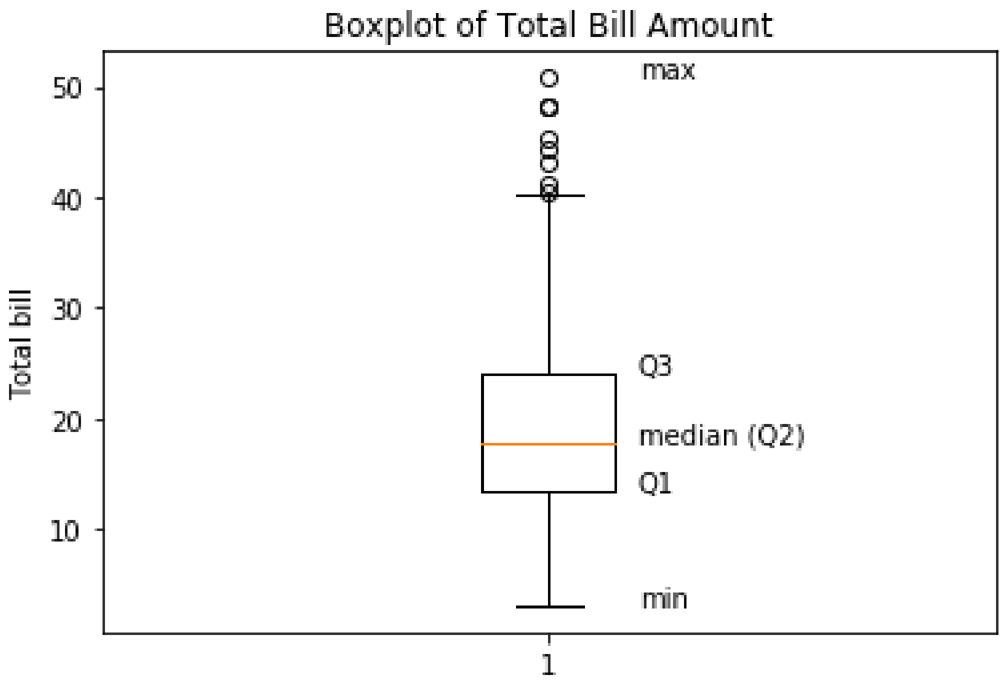
1.1.4 1.6 Box Plot

Boxplot is a way to visualize the five-number summary of the variable. The five-number summary includes the numerical quantities like minimum, first quartile (Q1), median (Q2), third quartile (Q3), and maximum. It gives information about the outliers in the data. Detecting and removing outliers is one of the essential steps in exploratory data analysis. Boxplots also tells about the distribution of the data.

Plot the boxplot of 'total_bill' to check the distribution and presence of outliers in the variable.

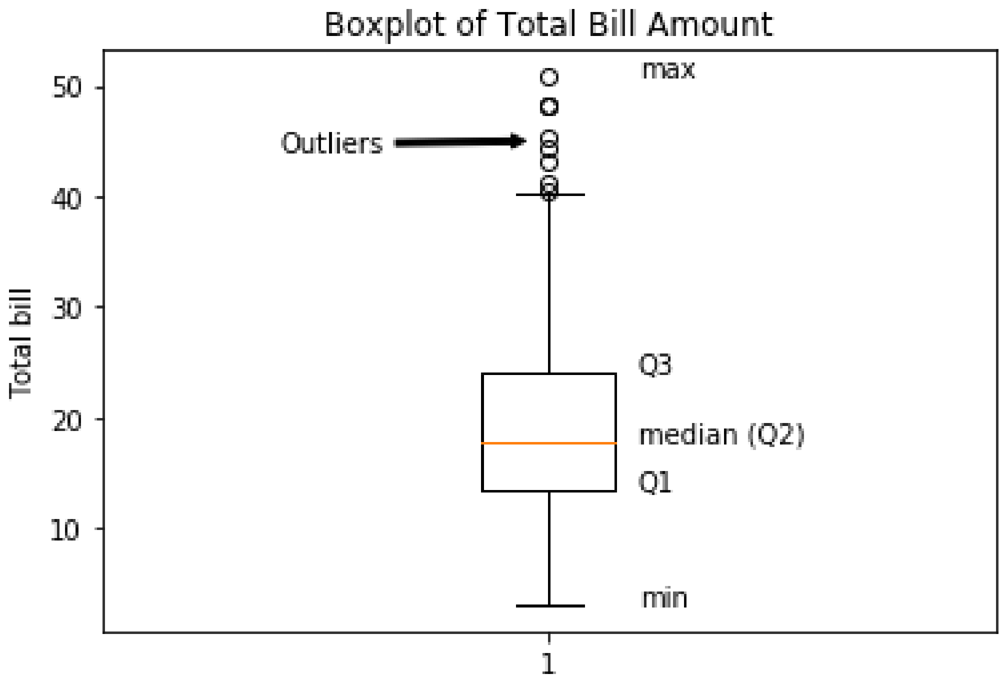
[50]:
```
# plot a boxplot of total bill
plt.boxplot(df_tips['total_bill'])

# add labels for five number summary using text()
# x is location on x-axis
# y is the location on y-axis
# s is the text
plt.text(x = 1.1, y = df_tips['total_bill'].min(), s ='min')              # for␣
 ↪minimun
```

```
plt.text(x = 1.1, y = df_tips.total_bill.quantile(0.25), s ='Q1')         # for␣
 ↪quartile 1
plt.text(x = 1.1, y = df_tips['total_bill'].median(), s ='median (Q2)')   # for␣
 ↪quartile 2
plt.text(x = 1.1, y = df_tips.total_bill.quantile(0.75), s ='Q3')         # for␣
 ↪quartile 3
plt.text(x = 1.1, y = df_tips['total_bill'].max(), s ='max')              # for␣
 ↪maximum


# add the graph title
plt.title('Boxplot of Total Bill Amount')

# add the y-axis label
plt.ylabel('Total bill')

# display the plot
plt.show()
```



The above boxplot clearly shows the presence of outliers above the horizontal line. We can add an arrow to showcase the outliers. Also, the median (Q2) is represented by the orange line, which is nearer to Q1 than Q3, indicating that the total bill is positively skewed.

```
[51]: # plot a distribution of total bill
plt.boxplot(df_tips['total_bill'])

# add labels for five number summary using text()
# x is location on x-axis
# y is the location on y-axis
# s is the text
plt.text(x = 1.1, y = df_tips['total_bill'].min(), s ='min')              # for␣
 ↪minimun
plt.text(x = 1.1, y = df_tips.total_bill.quantile(0.25), s ='Q1')         # for␣
 ↪quartile 1
plt.text(x = 1.1, y = df_tips['total_bill'].median(), s ='median (Q2)')   # for␣
 ↪quartile 2
plt.text(x = 1.1, y = df_tips.total_bill.quantile(0.75), s ='Q3')         # for␣
 ↪quartile 3
plt.text(x = 1.1, y = df_tips['total_bill'].max(), s ='max')              # for␣
 ↪maximum

# add an arrow (annonate) to show the outliers
# xy is the location of the head of the arrow
# xytext is the location of the text
# arrowprops specify the properties of the arrow
# facecolor specifies the arrow color
# arrowstyle specifies the arrow stype
plt.annotate('Outliers', xy = (0.98,45),xytext=(0.7, 44), arrowprops =␣
 ↪dict(facecolor='black', arrowstyle = 'simple'))

# add the graph title
plt.title('Boxplot of Total Bill Amount')
# add the y-axis label
plt.ylabel('Total bill')

# display the plot
plt.show()
```



1.1.5 1.7 Heatmap

Heatmap is a graphical representation of two-dimensional data. It represents the individual values that are contained in a matrix are represented as colours. Generally, the heatmap is used to the correlation matrix among numeric variables of a dataset. Each square in the heatmap shows the correlation between the two variables on each axis.

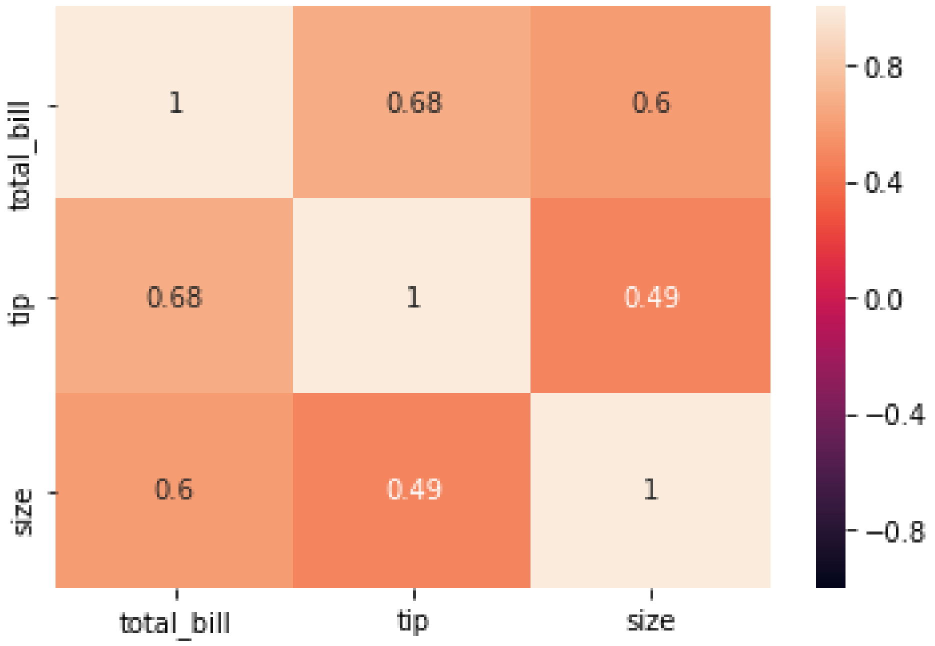
Compute correlation between the variables using .corr() function. Plot a heatmap of the correlation matrix.

[68]:
```
# compute correlation
corr_matrix = df_tips.corr()

# display the correlation matrix
corr_matrix
```

[68]:
```
            total_bill       tip      size
total_bill    1.000000  0.675734  0.598315
tip           0.675734  1.000000  0.489299
size          0.598315  0.489299  1.000000
```

[69]:
```
# plot heatmap
# 'annot=True' returns the correlation values
# vmin is the minimum range of values
sns.heatmap(corr_matrix, annot = True, vmin = -1)
```

```
# display the plot
plt.show()
```



The above plot shows that there is a moderate correlation between 'total_bill' and 'tip' (0.68). The diagonal values are '1' as it is the correlation of the variable with itself.

The above boxplot clearly shows the presence of outliers above the horizontal line. We can add an arrow to showcase the outliers. Also, the median (Q2) is represented by the orange line, which is nearer to Q1 than Q3, indicating that the total bill is positively skewed.