Exploratory Data Analysis

Practical 1&2 – W4

Learning Objectives

Learn various visualisation methods, such as

1. Line Graph
2. Bar-Graph
3. Pie-Graph
4. Histogram
5. Boxplot

1. Data Visualisation

Visualisation is the key to data analysis. The most popular Python package for visualisation is matplotlib and seaborn, but sometimes pandas will be handy for you. Pandas also provide some visualisation plots quickly. For the fundamental analysis part, it will be easy to use.

1.1 Line plot

The line plot is the simplest of all graphical plots. A line plot is utilised to follow changes continuously and show information as a series. For creating a line plot in pandas, we use .plot(), two columns' names for the argument.

```
dict_line = {
    'year': [2016, 2017, 2018, 2019, 2020, 2021],
    'price': [200, 250, 260, 220, 280, 300]
}

df_line = pd.DataFrame(dict_line)

# use plot() method on the dataframe
df_line.plot('year', 'price');
```

This code will create a line chart. Observe the output.

Task: Pandas Method:

Create a line graph for `'Payment_method'` features. Visualise the frequencies of different payment methods.

- Create a sub-data frame by selecting only one feature. Count the unique values and their frequencies using the value_counts() method.

```
pay_df = pd.DataFrame(df[['Payment_method']].value_counts())
print(pay_df)
```

- value_counts() method returns a series. Reindex the index of the newly created data frame.
  ```
  pay_df = pay_df.reset_index()
  ```
- Rename the column names. Give a name to the newly created column of frequency counts. It could be "Count" or any useful label.
  ```
  pay_df.columns = ['Payment_method', 'Count']
  print(pay_df)
  ```

- Create a line graph using the plot function.
  ```
  pay_df.plot('Payment_method', 'Count', kind='line')
  ```

Alternate Method:

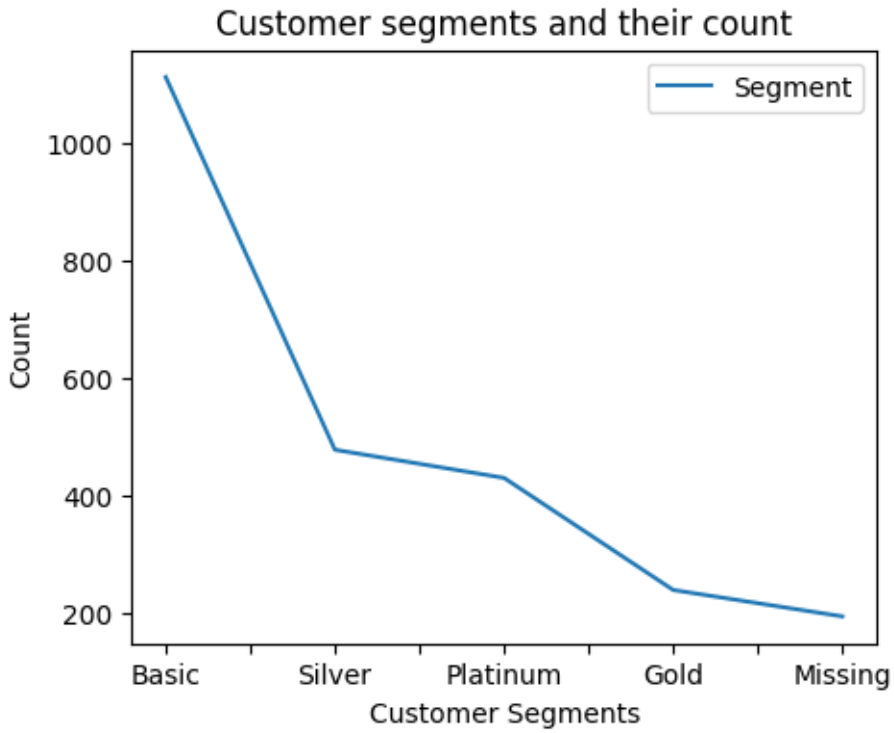
The same could be achieved by calling the

```
df['Payment_method'].value_counts().plot(kind='line',title="Payment_method")
```

Task:

Create a Line graph for the column 'Segments'. Read Panda's documentation to learn to set x and y labels and figure titles. Your figure should also have a legend. Your figure size should be L*W = 5*4.

Your figure should look like the one below.



1.2 Bar plot

A bar plot/bar chart, shows quantitative or qualitative values for different category items. In a bar, plot data are represented in the form of bars. Bars' length or height are used to describe the quantitative value for each item. Bar plots can be plotted horizontally or vertically. For creating these plots, look below.

For vertical bar: (default)

```
df['Employees_status'].value_counts().plot(kind='bar');
```

- The title of the figure could be set using title = "Your title". Moreover, there are various arguments that you can select to make your figure more readable.

- You can also set the length and width of your figure using the figsize=[10,10] argument.

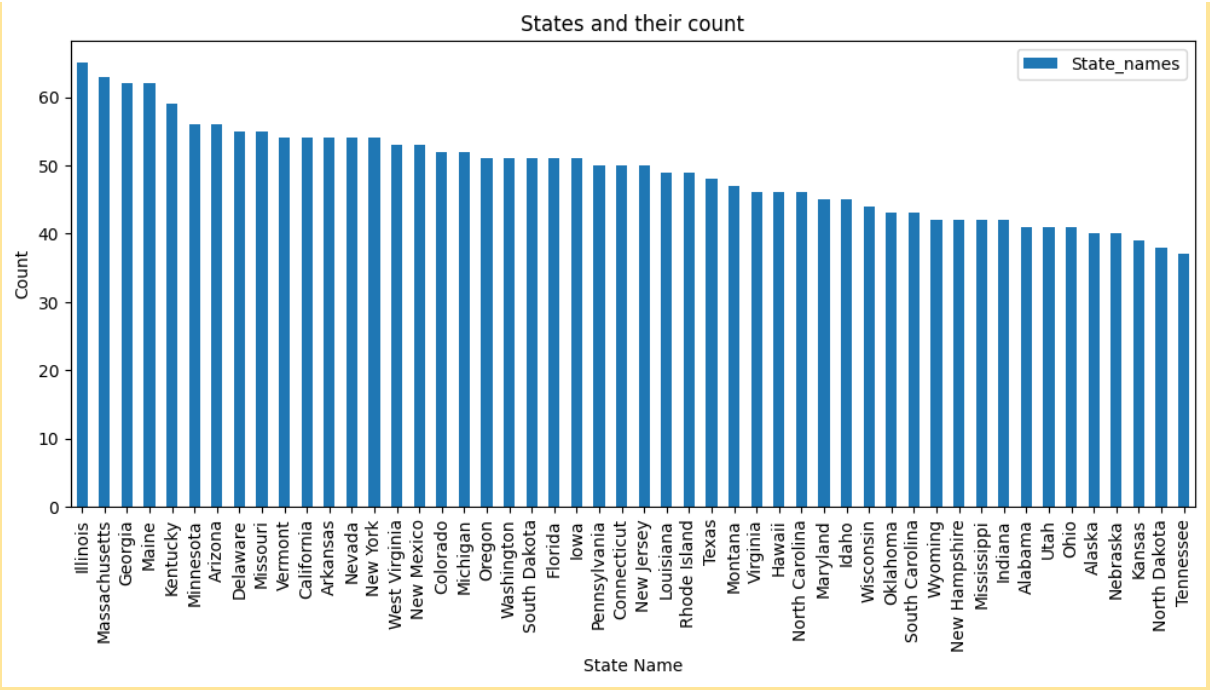
```
df['Employees_status'].value_counts().plot(kind='bar',
title="Employee_Status", figsize= [5,5]);
```

For horizontal bar:

```
df['Employees_status'].value_counts().plot(kind='barh');
```

Task: Create a bar graph for states and their count. Your figure should look like the figure below.

Hint: Try using a bigger length for figure size for all state's names to appear readable in the figure.

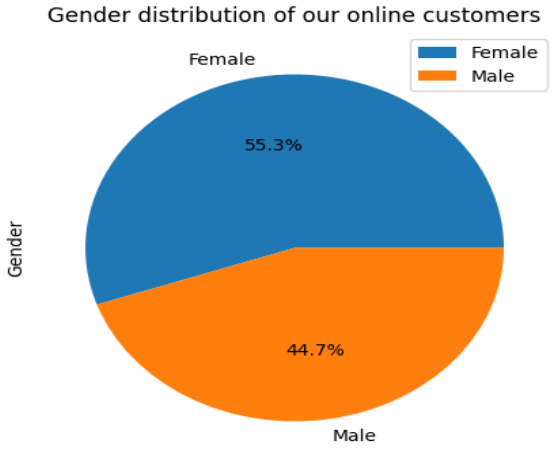


1.3 Pie plot

A pie plot/pie chart is a circular graph representing the total value with its components. The area of a circle represents the total value, and the different sectors represent the different parts. In this plot, the data are expressed as percentages. Each component is described as a percentage of the total value.

In pandas, we use kind=pie in plot() function in data frame column or series to create a pie plot. `df['Segment'].value_counts().plot(kind='pie');`

Task: Create a pie plot for gender feature. Also, show the percentage of each category on the pie plot. Your figure should look like the figure given below.



1.4 Histogram

A histogram shows the frequency and distribution of quantitative measurement across grouped values for data items. It is commonly used in statistics to show how many of a certain type of variable occurs within a specific range or bucket. Below, we will plot a histogram for looking at Age distribution.

```
df.plot(
    y='Age',
    kind='hist',
    bins=10
);
```

Task: Create a histogram of the age feature with a bin size of 5 and 20. Have you observed any difference between the three visualisations?

1.5 Box Plot

A box plot is also known as a box and whisker plot. This plot is used to show the distribution of a variable based on its quartiles. Box plot displays the five-number summaries of a set of data. The five-number summaries are: minimum, first quartile, median, third quartile, and maximum values. They are popular to identify outliers.

We can plot this by one column or multiple columns. For multiple columns, we need to pass the column name in the y variable as a list.

```
df.plot(y=['Amount_spent'], kind='box');
```

Task: The Amount spent feature luckily has no outliers. Run the following code, and it will add some rows into our data with extreme amount spent values.

```
df = df.append({'Transaction_ID': 160000, 'Gender': 'Male','Age': 40,
'Marital_status': 'Married','State_names': 'Hawaii', 'Segment':
'Basic','Employees_status': 'workers', 'Payment_method':
'PayPal','Referal': 1.0, 'Amount_spent': 4000}, ignore_index=True)

df = df.append({'Transaction_ID': 160001, 'Gender': 'Male','Age': 40,
'Marital_status': 'Married','State_names': 'Hawaii', 'Segment':
'Basic','Employees_status': 'workers', 'Payment_method':
'PayPal','Referal': 1.0, 'Amount_spent': 5000}, ignore_index=True)

df = df.append({'Transaction_ID': 160002, 'Gender': 'Male','Age': 40,
'Marital_status': 'Married','State_names': 'Hawaii', 'Segment':
'Basic','Employees_status': 'workers', 'Payment_method':
'PayPal','Referal': 1.0, 'Amount_spent': 6000}, ignore_index=True)

df = df.append({'Transaction_ID': 160003, 'Gender': 'Male','Age': 40,
'Marital_status': 'Married','State_names': 'Hawaii', 'Segment':
'Basic','Employees_status': 'workers', 'Payment_method':
'PayPal','Referal': 1.0, 'Amount_spent': 7000}, ignore_index=True)

df = df.append({'Transaction_ID': 160004, 'Gender': 'Male','Age': 40,
'Marital_status': 'Married','State_names': 'Hawaii', 'Segment':
'Basic','Employees_status': 'workers', 'Payment_method':
'PayPal','Referal': 1.0, 'Amount_spent': 8000}, ignore_index=True)

df = df.append({'Transaction_ID': 160005, 'Gender': 'Male','Age': 40,
'Marital_status': 'Married','State_names': 'Hawaii', 'Segment':
'Basic','Employees_status': 'workers', 'Payment_method':
'PayPal','Referal': 1.0, 'Amount_spent': 9000}, ignore_index=True)
```

Now, recreate the boxplot and see the difference.
The new boxplot has outliers. Remove the outliers and replot the boxplot to remove the extreme values.

In a box plot, we can compare the distribution of categorical variables against a numerical variable.

Let's plot it with the `Employees_status` and `Amount_spent` columns with the

pandas `boxplot()` method:

```
import matplotlib.pyplot as plt

np.warnings.filterwarnings('ignore', category=np.VisibleDeprecationWarning)
fig, ax = plt.subplots(figsize=(6,6))

df.boxplot(by ='Employees_status', column =['Amount_spent'],ax=ax, grid =
False)
```