Data visualization (Basics)

- Representation of the data in a pictorial or graphical format
- Allow us to get the intuitive understanding of the data
- Helps to visualize the patterns in the data
- Python offers multiple great graphing libraries that come packed with lots of different features.
- Matplotlib: low level, provides lots of freedom
- Pandas Visualization: easy to use interface, built on Matplotlib
- Seaborn: high-level interface, great default styles
- Plotly: can create interactive plots
- Bokeh: used creating interactive visualizations for modern web browsers"

Primary Objects of matplotlib

No description has been provided for this image

- The [figure] is the overall figure space that can contain one or more plots
- The [axes] is the individual plots that are rendered within the figure

Anatomy of a figure

No description has been provided for this image

Using matplotlib.pyplot library to plot a chart

Installing library -

!pip install matplotlib

Importing library -

In [ ]:

```
import matplotlib.pyplot as plt
```

Use plt.show() as the last line of the code. It suppresses the memory address information

In [ ]:

```
import numpy as np
dates = np.arange('2019-01', '2022-01', dtype='datetime64[M]')
sales = np.array([42390, 77560, 77385, 76039, 42968, 53833, 47205, 68936, 51175, 48
profits = np.array([ 7206.3 ,  8531.6 , 13155.45,  9885.07,  7304.56,  9689.94, 566
```

In [ ]:

Resize the figure

In [ ]:

Customise plots

- color, labels, title, legends, grid

In [ ]:

Subplots

Visualise sales and profits as subplots

In [ ]:

Additional chart types

In [ ]:

```
import numpy as np
products = np.array(['Caffe Latte', 'Cappuccino', 'Colombian', 'Darjeeling', 'Decaf
sales = np.array([52248.0, 14068.0, 71060.0, 60014.0, 69925.0, 27711.0, 19231.0, 24
profits = np.array([17444.0, 5041.0, 28390.0, 20459.0, 23432.0, 7691.0, -2954.0, 71
target_profits = np.array([15934.0, 4924.0, 31814.0, 19649.0, 24934.0, 8461.0, 7090
target_sales = np.array([48909.0, 13070.0, 80916.0, 57368.0, 66906.0, 30402.0, 1821
```

Bar chart

In [ ]:

Horizantal bar chart

In [ ]:

Stacked bar chart

In [ ]:

Side by Side Bar Chart

In [ ]:

Pie chart

In [ ]:

Bullet chart

In [ ]:

Scatter Plot

In [ ]:

Introduction to Pandas Library

- Pandas is an open source library in python which is know for its rich applications and utilities for all kinds of mathematical, financial and statistical functions
- It is useful in data manipulation and analysis
- It provides fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data

Installing pandas

In [ ]:

```
!pip install pandas
```

Importing pandas

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Series

Series -

- are one-dimensional ndarray with axis labels (homogenous data)
- labels need not be unique but must be of immutable type

Creating Series

Ex. Create series using the given list of names

In [ ]:

Extracting elements from series

Indexing based on index number

In [ ]:

Indexing based on index name

In [ ]:

Assigning names as index to marks

In [ ]:

In [ ]:

Filtering Series / Conditional Indexing

In [ ]:

Operations on Series

In [ ]:

Ranking and Sorting

series.sort_values(`ascending=True` , `inplace=False` , `na_option = {"first","last"}`)
series.sort_index(`ascending=True` , `inplace=False`)
series.rank(`ascending=False` , `method={"average","min","dense"}` , `na_option = {"top","bottom"}`)

In [ ]:

Working with NULLs

In [ ]:

Dataframe

A DataFrame is two dimensional data structure where the data is arranged in the tabular format in rows and columns

DataFrame features:

- Columns can be of different data types
- Size of dataframe can be changes
- Axes(rows and columns) are labeled
- Arithmetic operations can be performed on rows and columns

Creating Dataframes

In [ ]:

```
employees = {"Name" : ["Jack", "Bill", "Lizie", "Jane", "George"],
             "Designation" : ["HR", "Manager", "Developer", "Intern", "Manager"],
             "Salary": [40000, 60000, 25000, 12000, 70000]}

df = pd.DataFrame(employees)
df
```

Accessing Dataframes

In [ ]:

Operations on dataframes

Ex. Average Salary

In [ ]:

Ex. Average Salary of managers

In [ ]:

Concataneting and Merging Dataframes

In [ ]:

```
df_jan = pd.DataFrame({"Order ID" : range(101, 111), "Sales" : np.random.randint(10
df_feb = pd.DataFrame({"Order ID" : range(111, 121), "Sales" : np.random.randint(10
df_mar = pd.DataFrame({"Order ID" : range(121, 131), "Sales" : np.random.randint(10
```

In [ ]:

In [ ]:

Concatenate

pd.concat(`tuple of dfs` , `ignore_index = False` , `axis=0`)

In [ ]:

Merging Dataframes

`df1.merge(df2, how="", left_on="", right_on="", left_index= "" , right_index="")`

In [ ]:

```
df_emp = pd.DataFrame({"Name" : ["Jack", "Bill", "Lizie", "Jane", "George"],
            "Designation" : ["HR", "Manager", "Developer", "Intern", "Manager"]})
df_emp
```

In [ ]:

```
base_salaries = pd.DataFrame({"Designation" : ["HR", "Developer", "Manager", "Senio
            "Salary": [40000, 25000, 70000, 1000000]})
base_salaries
```

Inner Merge

In [ ]:

Left Merge

In [ ]:

Right Merge

In [ ]:

Outer Merge

In [ ]:

Reading data from Data Sources

In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Reading data from MYSQL or SQLITE3

In [ ]:

```
!pip install sqlalchemy
```

Examples using Coffee Shop Dataset

In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Ex. Read data from `coffee_sales.csv`

In [ ]:

```

```

Drop a column or row from dataframe

In [ ]:

```

```

Drop null rows

df.dropna(`axis = 0` , `how = "any"` , `inplace = False`)

- axis 0 for row or 1 for column
- how - {any or all}

In [ ]:

```

```

Renaming a Columns

Rename Columns (column 5 - 8 are not accessible)

In [ ]:

```
headers = ["ShopID", "Year/Month", "Product", "Product Type", "State", "Target Prof
```

Rename Single Column

In [ ]:

```

```

Understanding Data in Dataframe

- `df.shape` - gives the size of the dataframe in the format (row_count x column_count)
- `df.dtypes` - returns a Series with the data type of each column
- `df.info()` - prints information about a DataFrame including the index dtype and columns, non-null values and memory usage
- `df.head()` - prints the first 5 rows of you dataset including column header and the content of each row
- `df.tail()` - prints the last 5 rows of you dataset including column header and the content of each row

In [ ]:

```
df_coffee.shape
```

```
In [ ]: df_coffee.dtypes
```

```
In [ ]: df_coffee.info()
```

```
In [ ]: df_coffee.head()
```

```
In [ ]: df_coffee.head(3)
```

```
In [ ]: df_coffee.tail()
```

```
In [ ]: df_coffee.tail(3)
```

Cleaning data

`df.apply()`

Convertinf Franchise into str

```
In [ ]:
```

Ex. Converting Sales and Profits columns to float types

```
In [ ]:
```

Working with null values

`df.isna()` - Detect missing values. Return a boolean same-sized object indicating if the values are NA.

`df.fillna(value=None, inplace=False)` - Fill NA/NaN values using the specified method.

```
In [ ]:
```

Ex. Identify Sales made by 'Caffe Latte'

```
In [ ]:
```

Removing Duplicate Data

```
In [ ]:
```

Replacing values

df.replace(old_value, new_value, inplace=True)

In [ ]:

Adding a new Column by calculation

Ex. Create columns showing `Sales` and `Profit` targets achieved

In [ ]:

Ex. Count the number times Targets are achieved

In [ ]:

Creating a bar chart to view Target Status

using matplpotlib

In [ ]:

using pandas

In [ ]:

using seaborn

In [ ]:

Setting and Resetting Index

Seting Index

`df.set_index(keys, drop=True, inplace=False,)` - Set the DataFrame index (row labels) using one or more existing columns or arrays (of the correct length). The index can replace the existing index or expand on it.

In [ ]:

Resetting Index

`df.reset_index(level=None, drop=False, inplace=False,)` - Reset the index of the DataFrame, and use the default one instead. If the DataFrame has a MultiIndex, this method can remove one or more levels.

In [ ]:

Indexing and Slicing using loc and iloc

Using loc to retrive data

- loc is label-based
- specify the name of the rows and columns that we need to filter out

Ex. Extract data for franchise 203

In [ ]:

Ex. Extract `City` column

In [ ]:

Ex. Extract `Sales` column for `Franchise - 203`

In [ ]:

Ex. Extract `Sales` and `Profit` column for `Franchise - 203, 504`

In [ ]:

Using iloc to retrive data

- iloc is integer index-based
- specify rows and columns by their integer index.

Ex. Extract row at index 2

In [ ]:

Ex. Extract rows at index position 2,3,4

In [ ]:

Ex. Extract column at index 0

In [ ]:

Ex. Extract column from index 0 to 2

In [ ]:

Ex. Extra rows 0 to 2 and columns 0 to 2

In [ ]:

Working with dates

In [ ]:

Insert a column in between

df.insert(`index` , `column_name` , `default_value`)

Create columns Year and Month - extract data using pd.DatetimeIndex

In [ ]:

Extract data for 2018

In [ ]:

Extract data for Jan - 2018

In [ ]:

Extract data for Jan - 2018 and 2019

In [ ]:

Extract data starting from April - 2019

In [ ]:

Extract data from Jan-2019 to Apr-2019

In [ ]:

Ranking and Sorting Dataframes

Ex. Rank the products in descending order of `Sales`

In [ ]:

Ex. Sort the data in ascending order of `Rank`

In [ ]:

Grouping Dataframes

```
df.groupby(by=None, as_index=True, sort=True, dropna=True)
```

Ex. Find product wise total Sales - bar chart

In [ ]:

use of `agg()`

Ex. Extract Monthly Sales and Profit

In [ ]:

Analysing Dataframes

- univariate analysis - boxplot, histogram, value_counts(), countplot, describe()
- bivariate analysis
    - categorial X numerical - barchart, piechart
    - 2 numerical - scatter plot
    - 2 categorial - crosstab
- multivariate - pivot table

Univariate Analysis

Summary Statistics

`df.describe()` - Generates descriptive statistics. Descriptive statistics include those that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding NaN values. Analyzes both numeric and object series, as well as DataFrame column sets of mixed data types. The output will vary depending on what is provided.

In [ ]:

`df.value_counts(normalize = False)` - returns a Series containing counts of unique rows in the DataFrame

In [ ]:

Histogram

In [ ]:

Box and Whisker Plot

No description has been provided for this image

Bivariate Analysis

`pd.crosstab(index, columns, values=None, aggfunc=None normalize=False)` - Computes a simple cross tabulation of two (or more) factors. By default computes a

frequency table of the factors unless an array of values and an aggregation function are passed.

Ex. Number of franchise where a product is sold across each state

In [ ]:

Ex. Product and the number of time Sales Target achieved

In [ ]:

`df.pivot_table(values=None, index=None, columns=None, aggfunc='mean')` **- creates a spreadsheet-style pivot table as a DataFrame. The levels in the pivot table will be stored in MultiIndex objects (hierarchical indexes) on the index and columns of the result DataFrame.**

In [ ]:

Barchart

Display Sales across products

In [ ]:

Piechart

In [ ]:

Scatter Plot

In [ ]:

Line Chart

Ex. Plot line chart showing monthly sales

using pandas

In [ ]:

using seaborn

In [ ]: