**Experiment No. 3**

**Title: Exploratory data analysis using PANDAS and Matplotlib visualization**

**Batch:** B2 **RollNo:** 1914078 **ExperimentNo.:** 3

### Aim: To use Pandas in built visualization and Matplotlib visualization to perform exploratory data analysis

**Resources needed:** Python IDE

### Theory:

# Pandas Built-in Data Visualization

Pandas have got built-in capabilities for data visualization. It's built-off of matplotlib, but it baked into pandas for easier usage!

importnumpy as np

import pandas as pd

importmatplotlib as mp

%matplotlib inline

There are some fake data csv files you can read in as dataframes:

df1**=**pd.read\_csv('df1',index\_col**=**0)

df1.head().

#Bar plot for df1 can be plot using

df2.plot.bar(stacked=True)

#plotinghistogram of only one column with 50 bins are setting bar width less than 1.

df1['A'].plot.hist(bins=50, rwidth=0.8)

#line plot in pandas

df1.plot.line()

#scatter plot with color and colormaps

df1.plot.scatter()

#boxplot of data frame will helps us to spot the outliers(mild and extream both)

df2.plot.box()

#density plots- to explore symmetric or assymetric nature of your dataset.

df2.plot.density()

# Mathplotlibfor Data Visualization

Matplotlib is the "grandfather" library of data visualization with Python. It was created by John Hunter. He created it to try to replicate MatLab's (another programming language) plotting capabilities in Python. So if you happen to be familiar with matlab, matplotlib will feel natural to you.

It is an excellent 2D and 3D graphics library for generating scientific figures.

Some of the major Pros of Matplotlib are:

* Generally easy to get started for simple plots
* Support for custom labels and texts
* Great control of every element in a figure
* High-quality output in many formats
* Very customizable in general

Matplotlib allows you to create reproducible figures programmatically

## Installation

You'll need to install matplotlib first with either:

conda install matplotlib

or pip install matplotlib

## Importing

Import the matplotlib.pyplot

**Basic Matplotlib Commands**

We can create a very simple line plot using the following

### plt.plot(x, y, 'r') # 'r' is the color red

### #setting x and y axis labels, title of plot

### plt.xlabel('X Axis Title Here')

### plt.ylabel('Y Axis Title Here')

### plt.title('StringTitlehere')

### Using subplot a grid of plots can be created as shown below. Also we can set marker and linestyle along with color of plot.

### plt.subplot(1,2,1)

### plt.plot(x, y, 'r.--') #

### plt.subplot(1,2,2)

### plt.plot(y, x, 'g\*-.');

### Matplotlib’s object oriented api:

### The main idea in using the more formal Object Oriented method is to create figure objects and then just call methods or attributes off of that object. This approach is nicer when dealing with a canvas that has multiple plots on it.

### # Create Figure object to represent an empty canvas

### fig = plt.figure()

### # Add set of axes to figure(manually)

### axes = fig.add\_axes([0.1, 0.1, 0.8, 0.8]) # left, bottom, width, height (range 0 to 1)

### # Plot on that set of axes

### axes.plot(x, y, 'b')

### axes.set\_xlabel('Set X Label') # Notice the use of set\_ to begin methods

### axes.set\_ylabel('Set y Label')

### axes.set\_title('Set Title')

### axes.set\_legend(loc=1)

### Figure size, aspect ratio and DPI

Matplotlib allows the aspect ratio, DPI and figure size to be specified when the Figure object is created. You can use the figsize and dpi keyword arguments.

* figsize is a tuple of the width and height of the figure in inches
* dpi is the dots-per-inch (pixel per inch).

For example:

fig**=**plt.figure(figsize**=**(8,4), dpi**=**100)

### \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Activities:

### Use Pandas AndMatplotlib For Following Activities

### 1. Download data set with atleast 1500 rows and 10-20 columns(numeric and non numeric) from valid data sources

### 2. Visualization to summarize your data set(density, frequency plot) 3. Measures of central tenancy of data set (mean, median etc) 4. Determining presence of outliers in your dataset(boxplot) 5. Correlation of attributes in your dataset( scatter plot and line plot on 2-3 pairs which are correlated) 6. Comparison of data ploted on same scale using barplot( 3 plots for 3 different columns pairs) 7. Use different, colors, styles, markers,marker with different size, legends, labels, colormaps dpi, figsizeetc in the plot

### 8. Save these plots 9. Write down your comment on each of these plots 10. place legends at appropriate location on the plot

### 11. Write down observation for your dataset for each of above listed task of analysis.

### Result: (script and output)

### Description of dataset:-

### Dataset name:- Cleaned\_Ott\_Movies.csv

|  |  |
| --- | --- |
| Number of Rows and Columns in the Dataset | (16774, 16) |
| Therefore, total no of values in the Dataset | 2,67,904 |
| Total no of movies | 16774 |
| Columns | ['ID', 'Title', 'Year', 'Age', 'IMDb', 'Rotten Tomatoes', 'Netflix', 'Hulu', 'Prime Video', 'Disney+', 'Type', 'Directors', 'Genres', 'Country', 'Language', 'Runtime'] |

### importing required modules and datadet :-

### import pandas as pd

### import numpy as np

### import matplotlib.pyplot as plt

### %matplotlib inline

### movies\_ds = pd.read\_csv("Cleaned\_Ott\_Movies.csv")

### movies\_ds.drop('Unnamed: 0',axis=1,inplace=True)

### movies\_ds.head()

### 

### 

### 2. Visualization to summarize your data set(density, frequency plot)

### #plotting yearwise frequency-count of movies for last 20 years

### movies\_ds['Year'].value\_counts().head(20).plot(color="red",kind="bar")

### 

### A bar graph was plotted for yearwise count of movies(for past 20years).

### We found that in past 20 years, 2017 had the most number of movies and 2001 had least number of movies.

### Measures of central tenancy of data set (mean, median etc)

### runtime= movies\_ds[movies\_ds["Runtime"]!="NOT MENTIONED"].head(1000)

### runtime["Runtime"]= runtime["Runtime"].apply(lambda x: x[0:-2]).astype(int)

### max\_r = runtime["Runtime"].max()+1

### min\_r = runtime["Runtime"].min()

### plt.hist(runtime["Runtime"],bins=20,range=(min\_r,max\_r), color="red")

### plt.savefig("Runtime\_hist.png")

### 

### A histogram was plotted to estimate various central tendies for runtime(only 1000 top-rated values are considered) of movies.

### Since the histogram is symmetric, We can estimate that the mean could be 100 and the median could be 100.

### Determining presence of outliers in your dataset(boxplot)

### plt.figure(figsize=(6,6))

### plt.boxplot(movies\_ds["IMDb"])

### plt.ylabel("IMDb rating")

### plt.title("finding outliers in IMDb ratings")

### plt.savefig("IMDb\_boxplot.png")

### 

### A box plot was used to detect the presence of outliers in IMDb ratings.

### We can conclude that the imdb ratings have fair amount of outliers.

### Correlation of attributes in your dataset( scatter plot and line plot on 2-3 pairs which are correlated)

figure = plt.figure(figsize=(7,7),dpi=80)

axes = figure.add\_axes([0.1,0.1,0.7,0.7])

axes.scatter(movies\_ds["Year"],movies\_ds["IMDb"],color='green',s=6,label="IMDb rating")

axes.scatter(movies\_ds["Year"],movies\_ds["Rotten Tomatoes"],color='orange',s=6,label="Rotten Tomatoes rating")

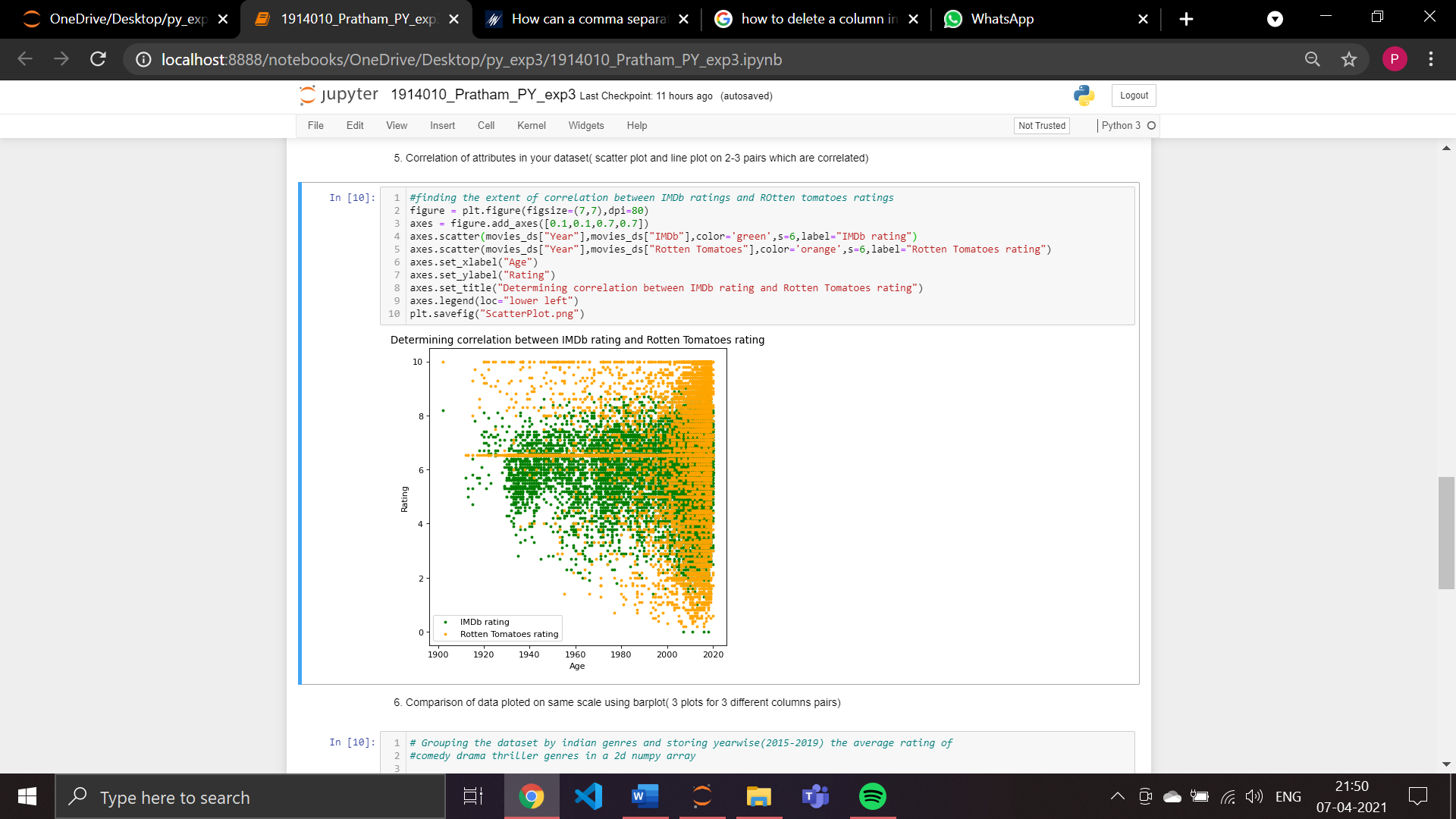
axes.set\_xlabel("Age")

axes.set\_ylabel("Rating")

axes.set\_title("Determining correlation between IMDb rating and Rotten Tomatoes rating")

axes.legend(loc="lower left")

plt.savefig("ScatterPlot.png")



A scatter plot was used to determine the extent of correlation between IMDb ratings and Rotten tomatoes ratings.

We can conclude that the points are quite scattered and the degree of correlation between imdb rating and rotten tomatoes rating is quite low.

### Comparison of data ploted on same scale using barplot( 3 plots for 3 different columns pairs)

### #step1 : Grouping the dataset by indian genres and storing yearwise(2015-2019) the average rating of comedy drama thriller genres in a 2d numpy array

### group\_by\_indian\_directors = movies\_ds[(movies\_ds["Country"]=="India")].groupby(["Genres","Year"])["IMDb"].mean().reset\_index()

### group\_by\_indian\_directors.sort\_values("IMDb",ascending=False,inplace=True)

### abc = group\_by\_indian\_directors[(group\_by\_indian\_directors["Year"]>2014) & (group\_by\_indian\_directors["Year"]<2020)]

### abc = abc[abc["Genres"].isin(["Comedy","Drama","Thriller"])].sort\_values(["Genres","Year"])

### data = np.array(abc["IMDb"])

### data = data.reshape(3,5)

### #step2: comparing the year-wise average rating of drama comedy thriller genres

### x=np.arange(5)

### fig = plt.figure(figsize=(7,8))

### axs = fig.add\_axes([0,0,1,1])

### w =0.25

### axs.bar(x, data[0],label="Comedy", color = 'b',width=w)

### axs.bar([i+0.25 for i in x ], data[1],label="Drama", color = 'c',width=w)

### axs.bar([i+0.5 for i in x ], data[2],label="Thriller", color = 'g',width=w)

### axs.set\_xticks([i+0.25 for i in x ])

### axs.set\_xticklabels(["2015","2016","2017","2018","2019"])

### axs.set\_xlabel("Year")

### axs.set\_ylabel("IMDb ratings")

### axs.set\_title("Average rating comparison of few popular genre")

### axs.legend()

### plt.savefig("barPlot.png")

### plt.style.use('dark\_background')

### 

### 

### A horizontally-stacked bar plot was used to compare the yearwise average imdb rating of indian movie’s comedy drama and thriller genre(past 5 year)

### It was observed that drama genre has performed really well as compared to other two genres. It’s average rating is not only 6+ but also higher than other 2 genres. The performance of Thriller genre movies was really poorly in 2016.

### Outcomes: Use python libraries like matplotlib, numpy, pandas, scipy for data visualization and scientific-mathematical data computing.

**Conclusion:** (Conclusion to be based on the objectives and outcomes achieved)

Through this experiment we learnt various Visualization methods were used to analyze the

dataset.

References:

1. <https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html>