

Laboratory Certificate

This is to certify that Smt./Sri PURVIKA S Reg No 20201ISE0050 has
satisfactorily completed the course of Experiments in DATA HANDLING
AND VISUALIZATION Prescribed by the PRESIDENCY UNIVERSITY in the
Laboratory of this College in the year 2023 - 2024

	Signature of the Lecturer
DATE:	in Charge

INDEX

SI NO.	Name of the Experiment
01	Introduction to Numpy
02	Working to Pandas
03	Data Cleaning
04	Zscore Normalization
05	Outlier Detection with IQR
06	Matplotlib
07	Interacting with Web API
08	Colormaps
09	Heatmaps
10	Seaborn color pallette
11	Univariate, Bivariate, Multivariate Visualization
12	Text
13	Time Series

LABSHEET 1

```
from matplotlib import pyplot as plt
plt.style.use('seaborn-whitegrid')
import numpy as np
print("step 1")
```

step 1
<ipython-input-4-240c5389bdd3>:2: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6,
 plt.style.use('seaborn-whitegrid')

fig = plt.figure()
ax = plt.axes()
ax.grid()

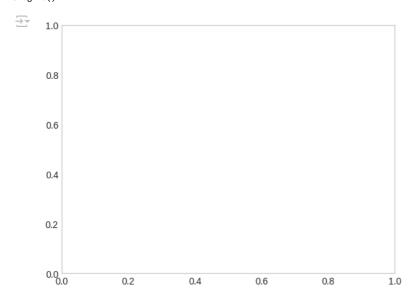
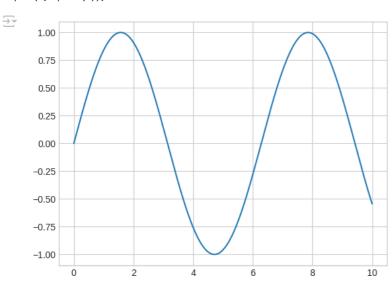


fig = plt.figure()
ax = plt.axes()

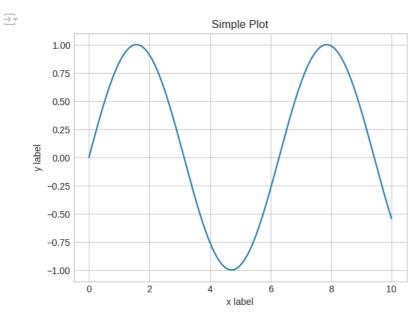
x = np.linspace(0, 10, 1000)
ax.plot(x, np.sin(x));



```
# Lets add a title and labels to the plot

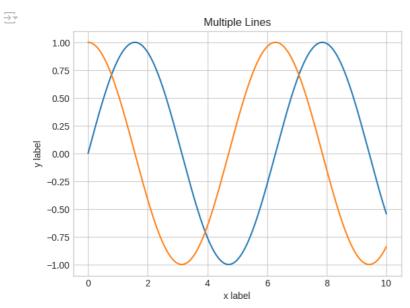
fig = plt.figure()
ax = plt.axes()

x = np.linspace(0, 10, 1000)
ax.plot(x, np.sin(x))
ax.set_title('Simple Plot')  # Add a title
ax.set_xlabel('x label')  # Add x label
ax.set_ylabel('y label');  # Add y label
```



```
# Lets add a title to the plot above
fig = plt.figure()
ax = plt.axes()

x = np.linspace(0, 10, 1000)
ax.plot(x, np.sin(x))
ax.plot(x, np.cos(x))
#ax.plot(x, np.tan(x))
ax.set_title('Multiple Lines');
ax.set_xlabel('x label')
ax.set_ylabel('y label')
plt.show()
```



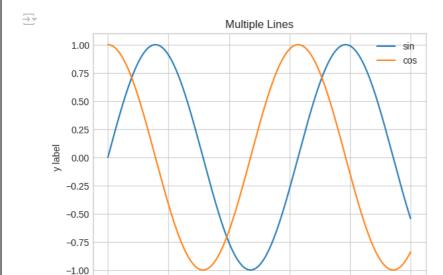
```
5/15/24, 4:13 PM
fig = plt.figure()
```

```
rig = pit.rigure()
ax = plt.axes()

x = np.linspace(0, 10, 1000)
ax.plot(x, np.sin(x), label = 'sin')
ax.plot(x, np.cos(x), label = 'cos')
ax.set_title('Multiple Lines');
ax.set_vlabel('x label')
ax.set_ylabel('y label')
ax.legend()
# ax.legend(loc=1)
plt.show()
```

0

2



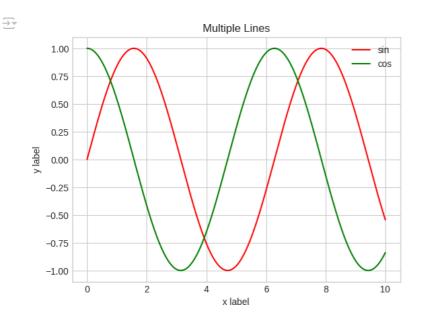
```
fig = plt.figure()
ax = plt.axes()

x = np.linspace(0, 10, 1000)
ax.plot(x, np.sin(x), label = 'sin', color = 'red')  # specify color by name
ax.plot(x, np.cos(x), label = 'cos', color = 'g')  # short color code (rgbcmyk)
ax.set_title('Multiple lines');
ax.set_xlabel('x label')
ax.set_ylabel('y label')
ax.legend();
```

x label

8

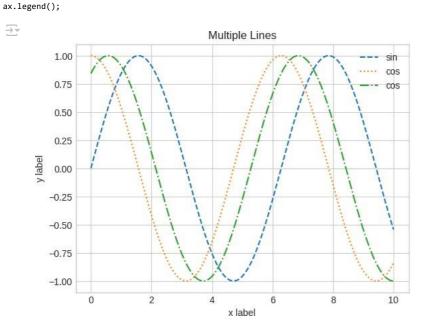
10



```
5/15/24, 4:13 PM
```

```
fig = plt.figure()
ax = plt.axes()
# ax.grid(linestyle = '--')

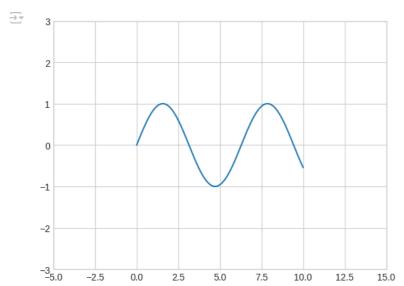
x = np.linspace(0, 10, 1000)
ax.plot(x, np.sin(x), label = 'sin', linestyle = 'dashed')
ax.plot(x, np.cos(x), label = 'cos', linestyle = 'dotted')
ax.plot(x, np.sin(x+1), label = 'cos', linestyle = 'dashdot')
ax.set_title('Multiple Lines');
ax.set_xlabel('x label')
ax.set_ylabel('y label')
```



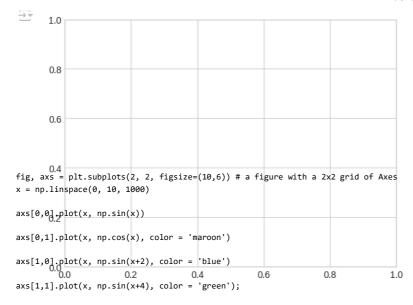
```
fig = plt.figure()
ax = plt.axes()

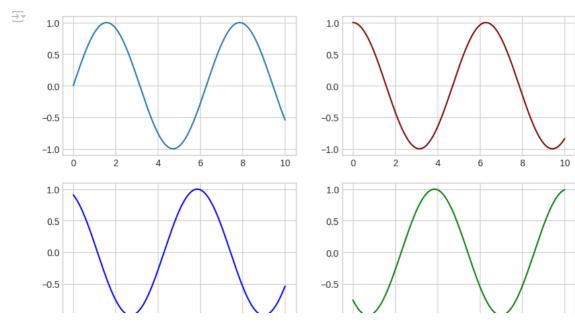
x = np.linspace(0, 10, 1000)
ax.plot(x, np.sin(x))

ax.set_xlim(-5, 15)
ax.set_ylim(-3, 3);
```



fig, ax = plt.subplots() # a figure with a single Axes





□ LABSHEET 2

□ pandas

```
import pandas as pd
data=pd.read_csv(r'C:\Users\Thejas Venugopal\Downloads\nyc_weather.csv')
data.head()
```

$\overrightarrow{\Rightarrow}$		EST	Temperature	DewPoint	Humidity	Sea Level PressureIn	VisibilityMiles	WindSpeedMP
	0	1/1/2016	38	23	52	30.03	10	8.
	1	1/2/2016	36	18	46	30.02	10	7.
	2	1/3/2016	40	21	47	29.86	10	8.
	3	1/4/2016	25	9	44	30.05	10	9.
	4							+

□ pandas series

□ with d being a dictionary

```
d={'a':1.,'b':2,'c':3}
s=pd.Series(d,index=['b','c','d'])
s

b     2.0
     c     3.0
     d     NaN
     dtype: float64
```

□ changing the index

□ dtype = float

```
n=np.array([1,2,3])
s1=pd.Series(n,dtype=float)
s1
```

```
syntax
```

```
pd.Series(data,index=[],dtype=, name=, copy=,)
```

□ combining 2 arrays to make an object

□ handling missing values

```
d={'a':1.,'b':2,'c':3}
s=pd.Series(d,index=['b','c','d'])
print(s)
 <u>→</u> b
           2.0
     С
           3.0
         NaN
     dtype: float64
s.isna().sum()
<u>⇒</u> 1
s.dropna()
b 2.0
c 3.0
     dtype: float64
d={'a':1.,'b':2,'c':3}
s=pd.Series(d,index=['b','c','d'])
print(s)
 <u></u> b
           2.0
     c 3.0
d NaN
     dtype: float64
s.fillna(2)
<u>⇒</u> b
     b
c 3.c
2.0
           2.0
     dtype: float64
```

□ accessing elements from the index

```
series=pd.Series([1,2,3,4,5],index=['a','b','c','d','e'])
series[1]

2
series[:3]

a    1
b    2
c    3
```

→ 978

□ DATA FRAME

```
import pandas as pd
data = {'Name':['Alice', 'Bob', 'Claire', 'David'],
       'Age':[20, 21, 20, 22]}
df = pd.DataFrame(data)
print(df)
         Name Age
    0 Alice
               20
     1
         Bob
               21
     2 Claire
               20
        David
               22
\ensuremath{\text{\#}} creating a dataframe from a list of dictionary
df = pd.DataFrame(data)
print(df)
         Name Age
               20
       Alice
               21
     1
         Bob
    2 Claire
               20
       David 22
```

pd.DataFrame(df)



Start coding or generate with AI.

LABSHEET 3

Data Cleaning and Data Preprocessing:

- 1. Data cleaning is the process of changing or eliminating garbage, incorrect, duplicate, corrupted, or incomplete data in a dataset.
- 2. There's no such absolute way to describe the precise steps in the data cleaning process because the processes may vary from dataset to dataset.



□ Data Cleaning Cycle



Missing Values:

```
# import the pandas library
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
'h'],columns=['one', 'two', 'three'])
# df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
# print (df)
\overline{\rightarrow}
              one
                         two
                                  three
     a 0.375319 -0.763927 -0.762393
     c -1.093644 1.335944 -0.668966
     e -0.013401 0.155461 -0.843651
     f 0.423813 0.900266 -0.828664
     h -0.644593 2.654895 1.211697
```

Check for Missing Values:

To make detecting missing values easier (and across different array dtypes), Pandas provides the

isnull() and notnull() functions, which are also methods on Series and DataFrame objects –

```
5/15/24, 3:25 PM
                                                   2.1Data_Cleaning.ipynb - Colab
   import pandas as pd
   import numpy as np
   df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
   'h'],columns=['one', 'two', 'three'])
   df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
   # print (df['one'].isnull())
   # print(df)
   print(df["one"].isnull())
               False
         h
               True
               False
         C
         d
               True
         е
               False
         f
               False
               True
         g
```

Replacing the Missing Values

Name: one, dtype: bool

False

h

```
#Replace the missing values by 0
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(3, 3), index=['a', 'c', 'e'],columns=['one',
'two', 'three'])
df = df.reindex(['a', 'b', 'c'])
print (df)
print ("NaN replaced with '0':")
print (df.fillna(0))
             one
                               three
                      two
     a -0.961858 -1.671248 0.556286
             NaN
                       NaN
                                  NaN
     c -0.386504 -0.709324 0.622838
     NaN replaced with '0':
             one
                        two
                               three
     a -0.961858 -1.671248 0.556286
     b 0.000000 0.000000 0.000000
     c -0.386504 -0.709324 0.622838
```

Fill NA Forward and Backward

```
# Method Action
pad/fill Fill methods Forward
bfill/backfill Fill methods Backward
```

```
5/15/24, 3:25 PM
                                               2.1Data_Cleaning.ipynb - Colab
   import pandas as pd
   import numpy as np
   df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
   'h'],columns=['one', 'two', 'three'])
   df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
   print(df)
   print (df.fillna(method='pad'))
                                    three
                 one
                           two
         a 0.109813 -1.940379 -0.444834
        b
                 NaN
                           NaN
                                      NaN
        c -0.208020 0.309864
                                0.819870
                 NaN
                           NaN
                                      NaN
        e -0.465764 0.215614 1.031519
        f 1.189843 3.814140 0.954030
                 NaN
                           NaN
        g
         h 0.480653 0.552598 -0.888482
                 one
                           two
         a 0.109813 -1.940379 -0.444834
         b 0.109813 -1.940379 -0.444834
        c -0.208020 0.309864 0.819870
        d -0.208020 0.309864 0.819870
        e -0.465764 0.215614 1.031519
        f 1.189843 3.814140 0.954030
        g 1.189843 3.814140 0.954030
           0.480653 0.552598 -0.888482
   import pandas as pd
   import numpy as np
   df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
   'h'],columns=['one', 'two', 'three'])
   df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
   print (df.fillna(method='bfill'))
    \overline{\rightarrow}
                 one
                           two
                                    three
         a -1.204446 2.137228 -0.388020
         b 1.327178 2.355456 -1.347412
         c 1.327178 2.355456 -1.347412
         d -0.228600 1.300295 0.939832
         e -0.228600 1.300295 0.939832
         f -0.938383 2.278881 -0.098408
         g 0.726762 0.456629 -1.167753
         h 0.726762 0.456629 -1.167753
```

Drop Missing Values:

Use dropna function along with the axis argument.

By default, axis=0, i.e., along row, which means that if any value within a row is NA then the whole row is excluded.

```
5/15/24, 3:25 PM
                                                 2.1Data_Cleaning.ipynb - Colab
   import pandas as pd
   import numpy as np
   df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
   'h'],columns=['one', 'two', 'three'])
   print(df)
   df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
   print(df)
   print (df.dropna())
    \overline{\Rightarrow}
                  one
                            two
                                     three
         a -0.481989 -1.249458 -2.316982
         c 1.119240 -1.054186 -0.972090
         e -0.991040 -0.749165
                                 0.259387
         f -1.300768 -0.000567 -0.056870
           0.497341 0.984014 -1.094049
                             two
         a -0.481989 -1.249458 -2.316982
         b
                  NaN
                            NaN
           1.119240 -1.054186 -0.972090
         d
                  NaN
                            NaN
         e -0.991040 -0.749165
                                 0.259387
         f -1.300768 -0.000567 -0.056870
                  NaN
                            NaN
         g
            0.497341 0.984014 -1.094049
                                     three
                  one
                             two
         a -0.481989 -1.249458 -2.316982
           1.119240 -1.054186 -0.972090
         e -0.991040 -0.749165 0.259387
         f -1.300768 -0.000567 -0.056870
```

Replace Missing (or) Generic Values:

0.497341 0.984014 -1.094049

We can achieve this by applying the replace method.

Replacing NA with a scalar value is equivalent behavior of the fillna() function.

```
import pandas as pd
import numpy as np
df = pd.DataFrame({'one':[10,20,30,40,50,2000],
'two':[1000,0,30,40,50,60]})
print(df)
print (df.replace({1000:10,2000:60}))
\rightarrow
          one
                 two
      0
           10
                1000
           20
      1
                    0
      2
           30
                  30
      3
           40
                  40
                  50
      4
           50
         2000
                  60
         one
               two
      0
          10
                10
      1
          20
```

4 50 50 5 60 60

□ Data Preprocessing

- 1. Load data in Pandas
- 2. Drop columns that aren't useful
- 3. Drop rows with missing values
- 4. Create dummy variables
- 5. Take care of missing data
- 6. Convert the data frame to NumPy

Download Titanic-Dataset from Kaggle.com.

Here we are going to use train.csv dataset for preprocessing.

```
import pandas as pd
import numpy as np
from google.colab import drive
drive.mount('/content/drive')
```

→ Mounted at /content/drive

memory usage: 83.7+ KB

df = pd.read_csv(r"C:\Users\Thejas Venugopal\Downloads\train (1).csv")
df.info()

<<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):

#	Column	Non-Null Count	Dtype
0	PassengerId	891 non-null	int64
1	Survived	891 non-null	int64
2	Pclass	891 non-null	int64
3	Name	891 non-null	object
4	Sex	891 non-null	object
5	Age	714 non-null	float64
6	SibSp	891 non-null	int64
7	Parch	891 non-null	int64
8	Ticket	891 non-null	object
9	Fare	891 non-null	float64
10	Cabin	204 non-null	object
11	Embarked	889 non-null	object
dtype	es: float64(2), int64(5), obj	ect(5)

Creating Dummy variables

memory usage: 55.6+ KB

Instead of wasting our data, let's convert the Pclass, Sex and Embarked to columns in Pandas and drop them after conversion.

```
dummies = []
cols = ['Pclass', 'Sex', 'Embarked']
for col in cols:
    dummies.append(pd.get_dummies(df[col]))
```

Transfor the eigth columns

```
titanic_dummies = pd.concat(dummies, axis=1)
```

Concatenate the values with data frame

```
df = pd.concat((df,titanic_dummies), axis=1)
```

Remove the unwanted cols

```
df = df.drop(['Pclass', 'Sex', 'Embarked'], axis=1)
```

Take care of Missing data

Let's compute a **median or interpolate()** all the ages and fill those missing age values. Pandas has an interpolate() function that will replace all the missing NaNs to interpolated values.

Min Max Scaler and Standardization

Normalization is a rescaling of the data from the original range so that all values are within the new range of O and 1.

```
A value is normalized as
```

□ Data Standardization

Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is O and the standard deviation is 1.

A value is standardized as follows:

$$y = (x - mean) /$$

[1.

1.

standard_deviation Where the

mean is calculated as:

```
mean = sum(x) / count(x)
And the standard_deviation is calculated as:
standard_deviation = sqrt(sum((x - mean)^2)/
count(x)
from numpy import asarray
from sklearn.preprocessing import StandardScaler
# define data
data = asarray([[100, 0.001],
[8, 0.05],
[50, 0.005],
[88, 0.07],
[4, 0.1]])
print(data)
# define standard scaler
scaler = StandardScaler()
# transform data
scaled = scaler.fit transform(data)
```

□ LABSHEET 4

import numpy as np

```
import pandas as pd
# Example dataset
data = {
    'Feature1': [10, 20, 30, 40, 50],
    'Feature2': [5, 15, 25, 35, 45]
# Create a DataFrame
df = pd.DataFrame(data)
# Display the original data
print("Original Data:")
print(df)
 → Original Data:
        Feature1 Feature2
     0
               10
                          5
               20
                          15
               30
                          25
      3
               40
                          35
               50
# Function to normalize data using Z-score
def zscore_normalization(df):
    normalized_df = df.copy()
    for column in normalized\_df.columns:
        mean = normalized_df[column].mean()
        std = normalized_df[column].std()
        normalized_df[column] = (normalized_df[column] - mean) / std
    return normalized_df
\ensuremath{\text{\#}} Normalize the DataFrame
normalized_df = zscore_normalization(df)
\# Display the normalized data
print("\nNormalized Data (Z-score):")
print(normalized_df)
     Normalized Data (Z-score):
        Feature1 Feature2
     0 -1.264911 -1.264911
     1 -0.632456 -0.632456
     2 0.000000 0.000000
     3 0.632456 0.632456
     4 1.264911 1.264911
```

□ LABSHEET 5

from google.colab import files df = files.upload()

Choose Files No file chosen enable. Saving train.csv to train.csv Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to

import pandas as pd import numpy as np

data = pd.read csv('./train.csv')

data.head()

_														
_		PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	
	0	493	0	1	Molson, Mr. Harry Markland	male	55.0	0	0	113787	30.5000	C30	S	
	1	53	1	1	Harper, Mrs. Henry Sleeper (Myna Haxtun)	female	49.0	1	0	PC 17572	76.7292	D33	С	
	2	388	1	2	Buss, Miss. Kate	female	36.0	0	0	27849	13.0000	NaN	S	
	3	192	0	2	Carbines, Mr. William	male	19.0	0	0	28424	13.0000	NaN	S	
	4	687	0	3	Panula, Mr. Jaako Arnold	male	14.0	4	1	3101295	39.6875	NaN	S	

cols = ['Name', 'Ticket', 'Cabin'] filtered_data = data.drop(cols, axis = 1) filtered_data.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 712 entries, 0 to 711 Data columns (total 9 columns):

#	Column	Non-Null Count	Dtype
0	PassengerId	712 non-null	int64
1	Survived	712 non-null	int64
2	Pclass	712 non-null	int64
3	Sex	712 non-null	object
4	Age	566 non-null	float64
5	SibSp	712 non-null	int64
6	Parch	712 non-null	int64
7	Fare	712 non-null	float64
8	Embarked	710 non-null	object

dtypes: float64(2), int64(5), object(2)
memory usage: 50.2+ KB

data = data.dropna() data.info()

<class 'pandas.core.frame.DataFrame'> Int64Index: 148 entries, 0 to 695 Data columns (total 12 columns):

Ducu	coramiis (coc	ar ir coramis).	
#	Column	Non-Null Count	Dtype
0	PassengerId	148 non-null	int64
1	Survived	148 non-null	int64
2	Pclass	148 non-null	int64
3	Name	148 non-null	object
4	Sex	148 non-null	object
5	Age	148 non-null	float64
6	SibSp	148 non-null	int64
7	Parch	148 non-null	int64
8	Ticket	148 non-null	object
9	Fare	148 non-null	float64
10	Cabin	148 non-null	object
11	Embarked	148 non-null	object
	63 164/0		

dtypes: float64(2), int64(5), object(5)

memory usage: 15.0+ KB

data.head()

```
5/15/24, 3:31 PM
                                                                      Outlier Detection.ipynb - Colab
             PassengerId Survived Pclass
                                                                             Sex Age SibSp Parch
                                                                                                        Ticket
                                                                                                                   Fare
                                                                                                                             Cabin Embarked
          0
                                  0
                                                  Molson, Mr. Harry Markland
                                                                                  55.0
                                                                                                        113787 30.5000
                                                                                                                               C30
                                                                            male
                                          1 Harper, Mrs. Henry Sleeper (Myna female 49.0
          1
                       53
                                  1
                                                                                            1
                                                                                                      PC 17572 76.7292
                                                                                                                               D33
                                                                   Haxtun)
          9
                      752
                                          3
                                                        Moor, Master. Meier
                                                                           male
                                                                                  6.0
                                                                                            0
                                                                                                   1
                                                                                                        392096 12.4750
                                                                                                                              E121
                                                                                                          WE/P
                                                      Crosby, Miss. Harriet R female 36.0
                                                                                                   2
          10
                      541
                                          1
                                                                                            0
                                                                                                                71.0000
                                                                                                                               B22
                                                                                                          5735
   dummies = []
   cols = ['Pclass', 'Sex', 'Embarked']
   for col in cols:
     dummies.append(pd.get_dummies(data[col]))
   dummies
    \overline{\geq}
               1 2 3
         [
                  0
                    0
               1
               1 0
          1
                    0
          2
               0 1 0
          3
               0 1
                     0
          4
               0 0
                     1
          707
              0 0 1
          708
                    0
              1 0
          709
               0 0
                    1
          710 0 1 0
          711 1 0 0
          [712 rows x 3 columns],
               female male
          0
                    0
                          1
          1
                    1
                          0
          2
                    1
                          0
          3
                    0
                          1
          4
                    0
                          1
          707
                    1
                          0
          708
                    0
                          1
          709
                    0
                          1
          710
                    0
                          1
          711
                    0
                          1
          [712 rows x 2 columns],
              C Q
0 0
                    S
1
               1 0
          1
                    0
```

2

3 4

709

710

0 0 1

0 0 1 1 0 0 707 1 0 0 708

1 0 0

0 0

0 0 1

titanic_dummies = pd.concat(dummies, axis = 1) titanic_dummies

S

С

S

S

 $\overline{\rightarrow}$

	1	2	3	female	male	С	Q	S
0	1	0	0	0	1	0	0	1
1	1	0	0	1	0	1	0	0
2	0	1	0	1	0	0	0	1
3	0	1	0	0	1	0	0	1
4	0	0	1	0	1	0	0	1
707	0	0	1	1	0	1	0	0
708	1	0	0	0	1	1	0	0
709	0	0	1	0	1	0	0	1
710	0	1	0	0	1	0	0	1
711	1	0	0	0	1	0	0	1

712 rows x 8 columns

data.drop(['Pclass', 'Sex', 'Embarked'], axis = 1)

	PassengerId	Survived	Name	Age	SibSp	Parch	Ticket	Fare	Cabin
	0 493	0	Molson, Mr. Harry Markland	55.0	0	0	113787	30.5000	C30
	1 53	1	Harper, Mrs. Henry Sleeper (Myna Haxtun)	49.0	1	0	PC 17572	76.7292	D33
	2 388	1	Buss, Miss. Kate	36.0	0	0	27849	13.0000	NaN
	3 192	0	Carbines, Mr. William	19.0	0	0	28424	13.0000	NaN
	4 687	0	Panula, Mr. Jaako Arnold	14.0	4	1	3101295	39.6875	NaN
7	07 859	1	Baclini, Mrs. Solomon (Latifa Qurban)	24.0	0	3	2666	19.2583	NaN
7	08 65	0	Stewart, Mr. Albert A	NaN	0	0	PC 17605	27.7208	NaN
7	09 130	0	Ekstrom, Mr. Johan	45.0	0	0	347061	6.9750	NaN
7	10 21	0	Fynney, Mr. Joseph J	35.0	0	0	239865	26.0000	NaN
7	11 476	0	Clifford, Mr. George Quincy	NaN	0	0	110465	52.0000	A14

712 rows x 9 columns

data['Age'] = data['Age'].interpolate() print(data)

\rightarrow		Passeng	erId	Survive	d Pcla	SS			Name	9 /
	0		493		0	1	M	olson,	Mr. Harry Markland	1
	1		53		1	1 Harper	r, Mrs. He	nry Sle	eper (Myna Haxtun)	,
	2		388		1	2			Buss, Miss. Kate	h
	3		192		0	2		Car	bines, Mr. William	1
	4		687		0	3		Panula	, Mr. Jaako Arnolo	1
										,
	707		859		1	3 Bac	clini, Mrs	. Solom	on (Latifa Qurban)	
	708		65		0	1		Ste	wart, Mr. Albert A	
	709		130		0	3			Ekstrom, Mr. Johar	l
	710		21		0	2		Fy	nney, Mr. Joseph 3	J
	711		476		0	1	Cl	ifford,	Mr. George Quincy	,
		Sex	Age	SibSp	Parch	Ticket	Fare	Cabin E	mbarked	
	0	male	55.0	0	0	113787	30.5000	C30	S	
	1	female	49.0	1	0	PC 17572	76.7292	D33	С	
	2	female	36.0	0	0	27849	13.0000	NaN	S	
	3	male	19.0	0	0	28424	13.0000	NaN	S	
	4	male	14.0	4	1	3101295	39.6875	NaN	S	
	707	female	24.0	0	3	2666	19.2583	NaN	С	
	708	male	34.5	0	0	PC 17605	27.7208	NaN	С	
	709	male	45.0	0	0	347061	6.9750	NaN	S	
	710	male	35.0	0	0	239865	26.0000	NaN	S	
	711	male	35.0	0	0	110465	52.0000	A14	S	

[712 rows x 12 columns]

Outlier Detection.ipynb - Colab

..19

□ LABSHEET 6

```
import matplotlib.pyplot as plt
# import seaborn as sn
# print a empty figure
# linespace 10 points with 1000 data points
# styles
# sin x and cos x
\mbox{\tt\#} legend values, colors, setting x,\;y title and other stuff
# line styles (different styles for each line)
# setting access limits (interval limits)
# subplot (printing multiple plots)
# 0 1 y = sin and then 0 1 x = sin
                                                                 ■ Code ]

    Tex

# print a empty figure
fig = plt.figure()
plt.show()

→ <Figure size 640x480 with 0 Axes>
# print sin wave until 4pi
import numpy as np
x = np.linspace(0, 4*np.pi, 1000)
y = np.sin(x)
z = np.cos(x)
a = np.tan(x)
plt.plot(x, y, color="green", linestyle="dotted")
plt.plot(x, z, color="blue")
# Set the x-axis and y-axis limits
plt.xlim(0, 4*np.pi)
plt.ylim(-1, 1)
# Set the x-axis and y-axis labels
plt.xlabel('x')
plt.ylabel('sin(x) and cos(x)')
# Show the plot
# plt.show()
 \rightarrow Text(0, 0.5, 'sin(x) and cos(x)')
            0.75
            0.50
       in(x) and cos(x)
            0.25
            0.00
          -0.25
          -0.50
          -0.75
          -1.00
```

plt.xlabel('empty grid')

8

6

10

12

```
Text(0.5, 0, 'empty grid')

1.0

0.8 -

0.4 -

0.2 -

0.0

0.0

0.2

0.4

0.6

0.8

1.0
```

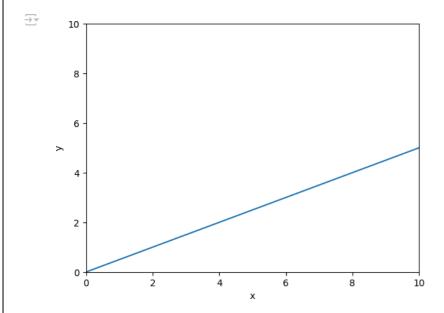
empty grid

```
x = np.linspace(0, 10, 1000)
y = np.linspace(0, 5, 1000)
# plt.plot(np.sin(x), np.cos(y))
plt.plot(x, y)

# Set the x-axis and y-axis limits
plt.xlim(0, 10)
plt.ylim(0, 10)

# Set the x-axis and y-axis labels
plt.xlabel('x')
plt.ylabel('y')

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



```
# printing a subplot
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(2, 1, 1)
plt.plot(x,y)

#plot 2:
#x = np.array([0, 1, 2, 3])
#y = np.array([10, 20, 30, 40])

#plt.subplot(2, 1, 2)
#plt.plot(x,y)
```

```
[<matplotlib.lines.Line2D at 0x7a4d87f00ca0>]
```

```
10 - 8 - 6 - 4 - 2 - 0.0 0.5 1.0 1.5 2.0 2.5 3.0
```

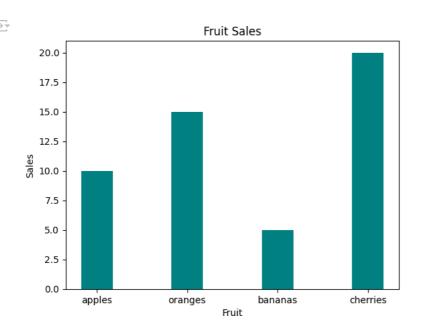
```
# barchar example with dictionary
import matplotlib.pyplot as plt

# Define the data
data = {'apples': 10, 'oranges': 15, 'bananas': 5, 'cherries': 20}

# Create a bar chart
plt.bar(list(data.keys()), list(data.values()), width=0.35, color="teal")

# Add title and axis labels
plt.title('Fruit Sales')
plt.xlabel('Fruit')
plt.ylabel('Sales')

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



```
# example of horizontal barchart with dictionary

# Define the data
data = {'apples': 10, 'oranges': 15, 'bananas': 5, 'cherries': 20}

# Create a horizontal bar chart
plt.barh(list(data.keys()), list(data.values()), color="purple", height=0.3)

# Add title and axis labels
plt.title('Fruit Sales')
# plt.xlabel('Sales')
# plt.ylabel('Fruit')

# Show the plot
show_plot = plt.show()
```

 $\overline{\geq}$

AttributeError Traceback (most recent call last)
<ipython-input-56-dbd46437747f> in <cell line: 16>()
 14 # Show the plot
 15 show_plot = plt.show()
---> 16 show_plot.set_xlabel('something')

AttributeError: 'NoneType' object has no attribute 'set_xlabel'

```
# Example data
```

fig, ax = plt.subplots()

people = ('Tom', 'Thejas', 'Harry', 'Slim', 'Jim')
y_pos = np.arange(len(people))
performance = 3 + 10 * np.random.rand(len(people))
error = np.random.rand(len(people))

hbars = ax.barh(y_pos, performance, xerr=error, align='cen')

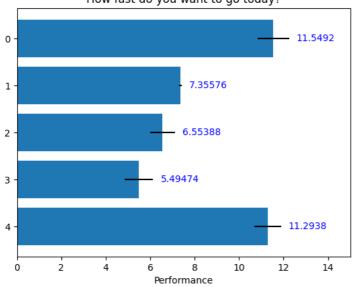
hbars = ax.barh(y_pos, performance, xerr=error, align='center')
ax.invert_yaxis()
ax.set_xlabel('Performance')
ax.set_title('How fast do you want to go today?')

Label with given captions, custom padding and annotate options
ax.bar_label(hbars, padding=8, color='b')
ax.set_xlim(right=15)

plt.show()

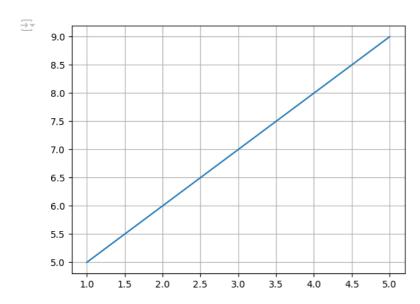
 $\overline{\Rightarrow}$

How fast do you want to go today?



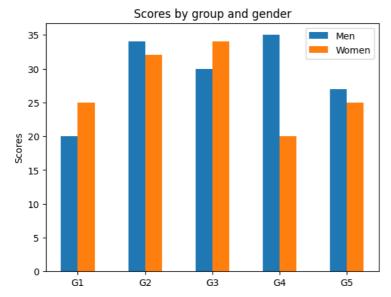
print(np.arange(10, 20, 2))

```
# pprint a axis plot with ax.grid()
import matplotlib.pyplot as plt
# Create a figure and an axes object
ax = plt.subplot()
# Plot some data
ax.plot([1, 2, 3, 4, 5], [5,6,7,8,9])
# Enable the grid
ax.grid(True)
# Show the plot
plt.show()
```



```
print(np.arange(10, 20, 2))
10 12 14 16 18]
# grouped bar charts example
import numpy as np
import matplotlib.pyplot as plt
labels = ['G1', 'G2', 'G3', 'G4', 'G5']
men_means = [20, 34, 30, 35, 27]
women_means = [25, 32, 34, 20, 25]
x = np.arange(len(labels))
\mbox{\tt\#} width of the individual component
width = 0.25
fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, men_means, width, label='Men')
rects2 = ax.bar(x + width/2, women_means, width, label='Women')
\mbox{\tt\#} Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.set xticks(x)
ax.set_xticklabels(labels)
ax.legend();
plt.show()
```





```
# adding labels to individual bars with their scores
```

```
fig, ax = plt.subplots()
ax.grid(linestyle='--', color='0.75', axis = 'y')
ax.set_axisbelow(True)

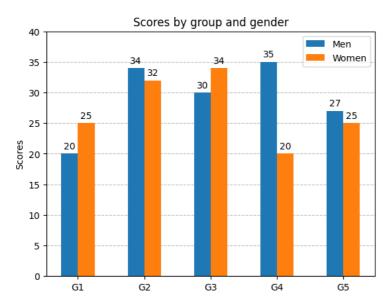
rects1 = ax.bar(x - width/2, men_means, width, label='Men')
rects2 = ax.bar(x + width/2, women_means, width, label='Women')

ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.set_xticks(x)
ax.set_xticklabels(labels)

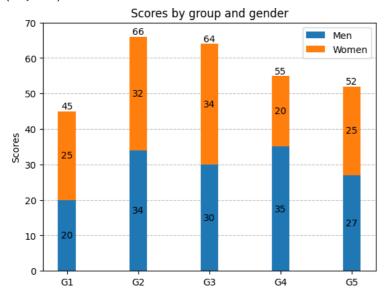
ax.legend()

# Adding the bar labels
ax.bar_label(rects1, padding=3)
ax.bar_label(rects2, padding=3)
ax.set_ylim(0,40);
```





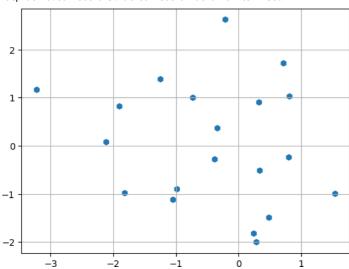
→ (0.0, 70.0)



```
# scatter plot
x = np.random.randn(20)
y = np.random.randn(20)

fig, ax = plt.subplots()
ax.grid(True)
ax.scatter(x, y, marker = 'h') # can change to any marker
```

<matplotlib.collections.PathCollection at 0x7b29651475e0>



```
fig, axs = plt.subplots(2, 3, sharex=True, sharey=True, figsize=(16,12));
# plt.style.use('seaborn-darkgrid')
# marker symbol
axs[0, 0].scatter(x, y, s=80, marker=">")
axs[0, 0].set_title("marker='>'")
# marker from TeX
axs[0, 1].scatter(x, y, s=80, marker=r'$\alpha$')
axs[0, 1].set_title("marker = " + r'$\alpha$')
# axs[0, 1].set_title(f"marker = {r'$\alpha$'}")
# marker from path
verts = [[-1, -1], [1, -1], [1, 1], [-1, -1]]
axs[0, 2].scatter(x, y, s=80, marker=verts)
axs[0, 2].set_title("marker=verts")
axs[1, 0].scatter(x, y, s=80, marker=(5, 0))
axs[1, 0].set_title("marker=(5, 0)")
# regular star marker
axs[1, 1].scatter(x, y, s=80, marker=(5, 1))
axs[1, 1].set_title("marker=(5, 1)")
# regular asterisk marker
axs[1, 2].scatter(x, y, s=80, marker=(5, 2))
axs[1, 2].set_title("marker=(5, 2)");
 ₹
                           marker='>'
                                                                                marker = \alpha
                                                                                                                                    marker=verts
                                                                                         α
                                                                                         α
                                                                                               α
                                                                                                                                    marker=(5, 2)
                          marker=(5, 0)
                                                                               marker=(5, 1)
```

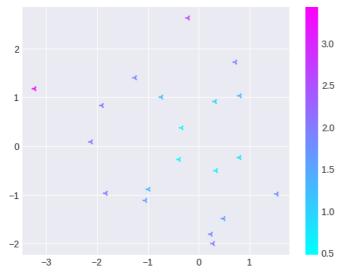
```
# setting the colors with matplotlib
plt.style.use('seaborn-darkgrid')

z1 = np.sqrt(x**2 + y**2)

fig, ax = plt.subplots()
pos = ax.scatter(x, y, c=z1, cmap='cool', marker='3')

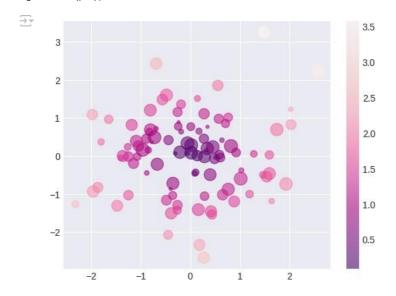
fig.colorbar(pos);
```

<ipython-input-51-3dd43bf91bb6>:2: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6,
plt.style.use('seaborn-darkgrid')

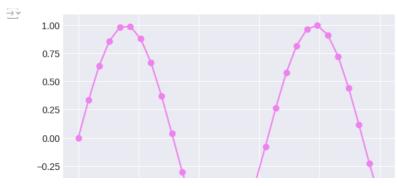


```
y = np.random.randn(100)
z1 = np.sqrt(x**2 + y**2)
z2 = np.random.randint(10, 200, size=len(x))
fig, ax = plt.subplots()
# pos = ax.scatter(x, y, c=z1, s=z2, alpha = 0.55, cmap='viridis')
pos = ax.scatter(x, y, c = z1, s = z2, alpha = 0.55, cmap='RdPu_r')
fig.colorbar(pos);
```

x = np.random.randn(100)



```
x = np.linspace(0, 10, 30)
y = np.sin(x)
plt.plot(x, y, 'o-', color='violet');
```

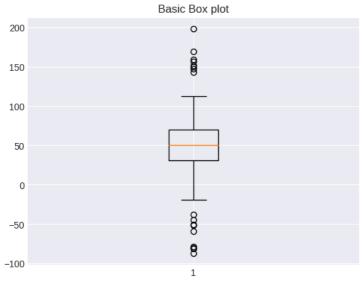


Box plots

```
# Generating the data
spread = np.random.rand(50) * 100
center = np.ones(25) * 50
flier_high = np.random.rand(10) * 100 + 100
flier_low = np.random.rand(10) * -100
data = np.concatenate((spread, center, flier_high, flier_low))

# Visualization of the data using box plot (basic)
fig, ax = plt.subplots()
ax.boxplot(data)
ax.set_title("Basic Box plot")
```

→ Text(0.5, 1.0, 'Basic Box plot')



Notched boxplot without outliers

0

0

112053 30.0000

111369 30.0000

23.4500

W./C. 6607

B42

NaN

C148

S

S

С

0

0

19.0

NaN

26.0

female

male

□ LABSHEET 7

import pandas as pd Code Text df = pd.read_csv('train.csv') df PassengerId Survived Pclass SibSp Name Sex Age Parch Ticket Fare Cabin **Embarked** 1 0 0 3 Braund, Mr. Owen Harris 22.0 1 A/5 21171 7.2500 NaN S male 0 Cumings, Mrs. John Bradley 2 1 38.0 0 PC 17599 71.2833 C85 С (Florence Briggs Th... STON/O2. 2 3 3 Heikkinen, Miss. Laina female 0 0 7.9250 S 1 26.0 NaN 3101282 Futrelle, Mrs. Jacques Heath (Lily 3 4 35.0 0 113803 53 1000 C123 S female May Peel) 0 Allen, Mr. William Henry 0 4 5 3 35.0 0 373450 8.0500 S male NaN 886 887 0 2 Montvila, Rev. Juozas male 27.0 0 0 211536 13.0000 NaN S

Graham, Miss. Margaret Edith female

Behr, Mr. Karl Howell

"Carrie"

Johnston, Miss. Catherine Helen

df.dtypes

887

888

889

888

889

890

1

1

PassengerId int64 Survived int64 Pclass. int64 Name object Sex object Age float64 SibSp int64 Parch int64 Ticket object Fare float64 Cabin object **Embarked** object dtype: object

df.describe()

$\overline{\rightarrow}$		PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
	count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
	mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
	std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
	min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
	25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
	50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
	75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
	max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

df.isna().sum()

PassengerId Survived 0 Pclass 0 0 Name Sex 0 Age 177 SibSp 0 Parch 0 Ticket 0 Fare Cabin 687 Embarked dtype: int64

age_mean_value=df['Age'].mean()
df['Age']=df['Age'].fillna(age_mean_value)

df.drop("Cabin",axis=1,inplace=True)

df.head()

\rightarrow	Pa	assengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
	0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
	1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th	female	38.0	1	0	PC 17599	71.2833	С
	2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	S
	3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S

filtered_age = df[df.Age>40]
filtered_age

\rightarrow		PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
-	6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	S
	11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	S
	15	16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55.0	0	0	248706	16.0000	S
	33	34	0	2	Wheadon, Mr. Edward H	male	66.0	0	0	C.A. 24579	10.5000	S
	35	36	0	1	Holverson, Mr. Alexander Oskar	male	42.0	1	0	113789	52.0000	S
	862	863	1	1	Swift, Mrs. Frederick Joel (Margaret Welles Ba	female	48.0	0	0	17466	25.9292	S
	865	866	1	2	Bystrom, Mrs. (Karolina)	female	42.0	0	0	236852	13.0000	S
	871	872	1	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47.0	1	1	11751	52.5542	S
	873	874	0	3	Vander Cruyssen, Mr. Victor	male	47.0	0	0	345765	9.0000	S
	879	880	1	1	Potter Mrs Thomas Jr (Lily Alexenia Wilson)	female	56 0	0	1	11767	83 1583	С

let's sort the column Name in ascending order
sorted_passengers = df.sort_values('Name',ascending=True,kind ='heapsort')

sorted_passengers.head(10)

\rightarrow												
		PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
	845	846	0	3	Abbing, Mr. Anthony	male	42.0	0	0	C.A. 5547	7.5500	S
	746	747	0	3	Abbott, Mr. Rossmore Edward	male	16.0	1	1	C.A. 2673	20.2500	S
	279	280	1	3	Abbott, Mrs. Stanton (Rosa Hunt)	female	35.0	1	1	C.A. 2673	20.2500	S
	308	309	0	2	Abelson, Mr. Samuel	male	30.0	1	0	P/PP 3381	24.0000	С
	874	875	1	2	Abelson, Mrs. Samuel (Hannah Wizosky)	female	28.0	1	0	P/PP 3381	24.0000	С
	365	366	0	3	Adahl, Mr. Mauritz Nils Martin	male	30.0	0	0	C 7076	7.2500	S
	401	402	0	3	Adams, Mr. John	male	26.0	0	0	341826	8.0500	S
	40	41	0	3	Ahlin, Mrs. Johan (Johanna Persdotter Larsson)	female	40.0	1	0	7546	9.4750	S
	855	856	1	3	Aks, Mrs. Sam (Leah Rosen)	female	18.0	0	1	392091	9.3500	S
	207	208	1	3	Albimona, Mr. Nassef Cassem	male	26.0	0	0	2699	18.7875	С

 $\label{eq:merged_df} $$ \mbox{merge(df.head(2),df.tail(2),how='outer',indicator=True)} $$ \mbox{merged_df} $$$

5/15/24, 4:16 PM				Data_wr	Data_wrangling.ipynb - Colab								
$\overline{\rightarrow}$	Pas	sengerId S	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	_merge
	0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5	7.2500	S	left only

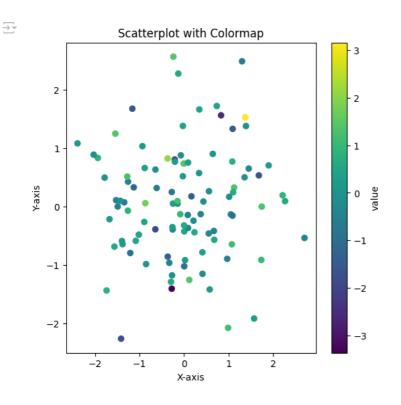
_	,	Passengeria	Survivea	PCTASS	Name	Sex	Age	510Sp	Parcn	пскет	Fare	Embarked	_merge
·	0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	s	left_only
	1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th	female	38.0	1	0	PC 17599	71.2833	С	left_only
	2	890	1	1	Behr. Mr. Karl Howell	male	26.0	0	0	111369	30.0000	С	riaht only

group_df = df.groupby('Name')

group_df

п LABSHEET 8

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# Sample dataframe with multiple columns
data = pd.DataFrame({
    "x": np.random.randn(100),
    "y": np.random.randn(100),
    "value": np.random.randn(100)
})
# Define the colormap and alpha values
cmap = "viridis"
alpha = 1
# Create the scatterplot
plt.figure(figsize=(6, 6))
plt.scatter(data["x"], data["y"], c=data["value"], cmap=cmap, alpha=alpha)
# Customize the plot (optional)
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Scatterplot with Colormap")
plt.colorbar(label="value")
# Show the plot
plt.show()
```



```
import pandas as pd
import numpy as np
print(np.random.randn(100))
```

```
7.25060198e-01 2.53900412e+00 1.26528031e+00 1.84136990e+00
      -2.60848832e+00 -5.59983281e-01 4.35035456e-01 -7.00367135e-02
      1.96931749e+00 1.04382097e+00 -5.23481680e-01 4.38611173e-01
     -6.03314609e-02 -1.62331938e+00 -1.75368806e-01 -1.45327854e-01
      7.11162067e - 01 \ -1.24752326e + 00 \ 1.10879435e + 00 \ 6.15797150e - 01
      3.22382085e-02 -4.94204444e-01 -1.56553377e+00 1.86476127e+00
     -1.53372917e+00 6.21845005e-01 1.08857491e+00 -1.69076421e+00
       -3.80722950e+00 4.70410313e-01 8.77562643e-01 -8.95285501e-01
      9.83561836e-01 9.32718991e-01 -6.78531171e-01 9.14953408e-05
      -2.21344622e+00 -6.15124358e-02 -9.18144802e-02 7.84013469e-01
      9.64181023e-01 -1.75737978e+00 1.19471319e+00 -1.02246958e-01
     -8.54821744e-01 -3.80648950e-01 -5.87306646e-01 5.54602769e-01
      1.40580004e+00 1.08580790e+00 -8.33862936e-01 7.08280769e-01
     -1.43281505e+00 -1.93642975e-01 6.86796860e-01 5.50748349e-01
     7.79495185e-01 -2.71795003e-01 -1.16407843e+00 1.38373041e+00
     -2.90569948e-01 1.27385062e+00 -4.24752220e-01 5.69263764e-01
      -1.45006382e+00 8.39335515e-01 -9.49539071e-01 -2.04611107e+00
```

.36

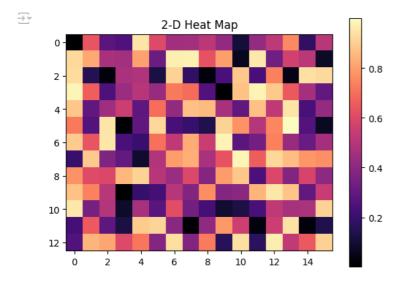
5/15/24,	, 3:48 PM 1.00680640e+00			Co	lormaps .ipynb	- Colab		
	1.00680640e+00 -9.72496062e-01	2.59974257e-01	-1.29858485e+00	9.67979863e-01				
	J.72430002C 01	1.723313030100	3.420301030 01	4.202304700 01				

LABSHEET 9

```
# Program to plot 2-D Heat map
# using matplotlib.pyplot.imshow() method
import numpy as np
import matplotlib.pyplot as plt
data = np.random.random(( 13 , 16 ))
plt.imshow( data,cmap="magma" )
```

plt.title("2-D Heat Map")

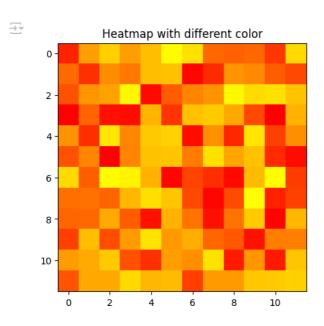
plt.colorbar()
plt.show()



Program to plot 2-D Heat map
using matplotlib.pyplot.imshow() method
import numpy as np
import matplotlib.pyplot as plt

data = np.random.random((12, 12))
plt.imshow(data, cmap='autumn')

plt.title("Heatmap with different color")
plt.show()



importing the modules
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

- # generating 2-D 10x10 matrix of random numbers
- # from 1 to 100

5/15/24, 3:50 PM data = np.random.randint(low=14,	Heatmap.ipynb - Colab
data = np.random.randint(low=14, high=100,	
0,	
	38

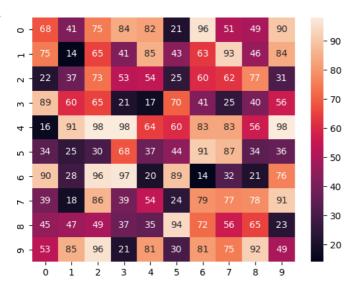
size=(10, 10))

plotting the heatmap

hm = sns.heatmap(data=data,annot=True)

displaying the plotted heatmap
plt.show()





All the IPython Notebooks in Python Scaborn Module lecture series by Dr. Milaan Parmar are available @ GitHub

П LABSHEET 10



☐ Seaborn Color Palettes

Color is an utmost important aspect of figure styling because it reveals pattern in the data if used effectively; or hide those patterns if used poorly. Even professionals often assume usage of color to portray data as a solved problem. They just pick a palette from a drop-down menu

(probably either a grayscale ramp or a rainbow), set start and end points & finally press apply. But it isn't that simple and thus many visualizations fail to represent the underlying data as appropriately as they could.

Primary objective with choice of color is to illuminate datapoints that are concealed in huge datasets. Quoting Robert Simmon:

Although the basics are straightforward, a number of issue complicate color choices in visualization. Among them: The relationship between the light we see and the colors we perceive is extremely complicated. There are multiple types of data, each suited to a different color scheme. A significant number of people (mostly men), are color blind. Arbitrary color choices can be confusing for viewers unfamiliar with a data set. Light colors on a dark field are

perceived differently

than dark colors on a bright field, which can complicate some visualization tasks, such as target detection.

One of the most fundamental and important aspects of color selection is the mapping of numbers to colors. This mapping allows us to pseudocolor an image or object based on varying numerical data. By far, the most common color map used in scientific visualization is the rainbow color map. Research paper on <u>Diverging</u>

Color Maps for Scientific Visualization by

Kenneth Moreland very well deals with the extended color concepts, if the topic interests you forfurther analysis.

With all that been said, let us now focus on what Seaborn has to offer BUT before doing that let me once again remind you that Seaborn runs on top of Matplotlib so any color that is supported by Matplotlib will be supported by Seaborn as well. So at first, let us understand what Matplotlib has to offer:

- * an RGB or RGBA tuple of float values in [0, 1] (e.g., (0.1, 0.2, 0.5) or
- (0.1, 0.2, 0.5, 0.3)) a hex RGB or RGBA string (e.g., '#0F0F0F' or '#0F0F0F')
- * a string representation of a float value in [0, 1] inclusive for gray
- level (e.g., '0.5') one of {'b', 'g', 'r', 'c', 'm', 'y', 'k', 'w'}
- a X11/CSS4 color name
- * a name from the xkcd color survey prefixed with 'xkcd:' (e.g.,
- 'xkcd:sky blue') one of {'CO', 'C1', 'C2', 'C3', 'C4', 'C5', 'C6', 'C7', 'C8', 'C9'}
- one of {'tab:blue', 'tab:orange', 'tab:green', 'tab:red', 'tab:purple', 'tab:brown', 'tab:pink',
 - 'tab:gray', 'tab:olive', 'tab:cyan'} which are the <u>Tableau</u> Colors from the 'T10' categorical palette (which is the default color cycle).

Note that all string specifications of color, other than "CN", are NOT case-sensitive. Let us briefly go through a couple of common supported colors here:

- RGB/RGBA tuples are 4-tuples where the respective tuple components represent Red,
 - Green, Blue, and Alpha (opacity) values for a color. Each value is a floating point number between 0.0 and 1.0. For example, the tuple (1, 0, 0, 1) represents an opaque red, while (0,
 - 1, 0, 0.5) represents a half transparent green.
- This is actually another way of representing RGBA codes and common Color Conversion Calculators can be used to translate values. Here is a **Hex to RGBA** and **RGB to Hex** Color converter for your future assistance.
- * Dictionary of values from {'CO', 'C1', 'C2', 'C3', 'C4', 'C5', 'C6', 'C7', 'C8', 'C9'} represent Color Quantization. I have attached a link in the provided notebook that shall guide you to an online book where on Page-29 you could find specifics.

My sole purpose of keeping you posted of Matplotlib background every now and then is only to

ensure that when you get to production-level and try to customize a plot as per your analysis, you should know what is ACTUALLY running in the background. This shall empower you to accordingly tweak parameters here and there. Let us now look into few Seaborn options for colors:

Importing required Libraries:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

Setting a figure size for all the plots we shall be drawing in this kernel:
sns.set(rc={"figure.figsize": (6, 6)})

☐ Building color palettes:

current_palette = sns.color_palette()
sns.palplot(current_palette)



The most important function for working with discrete color palettes is color_palette() . This function provides an interface to many (though not all) of the possible ways you can generate colors in seaborn, and it's used internally by any function that has a palette argument (and in some cases for a color argument when multiple colors are needed).

color_palette() will accept the name of any seaborn palette or matplotlib colormap (except jet, which you should never use). It can also take a list of colors specified in any valid matplotlib format (RGB tuples, hex color codes, or HTML color names). The return value is always a list of RGB tuples.

Finally, calling color_palette() with no arguments will return the current default color cycle.

sns.palplot(sns.color_palette("hls", 8))



sns.palplot(sns.color_palette("husl", 8))



Let me explain these Qualitative (or categorical) palettes. These are best when you want to distinguish discrete chunks of data that do not have an inherent ordering. Ideally, when importing Seaborn, the default color cycle is changed to a set of six colors that evoke the standard matplotlib color cycle. But when we have more than

6, say 8 categories in our data to distinguish, then the most common way is using hls color space, which is a simple transformation of RGB values.

Then there is also hls_palette() function that lets you control the lightness and saturation of colors.

All of it displayed above is just the basic Seaborn aesthetics. Let us now look at xkcd_rgb

dictionary that has 954 colors in it. Let us try to pull a few out of it:

sample_colors = ["windows blue", "amber", "greyish", "faded green", "dusty purple", "pale red",
sns.palplot(sns.xkcd_palette(sample_colors))



Other style is cubehelix color palette that makes sequential palettes with a linear increase or decrease in brightness and some variation in hue. Actually let us plot this color palette in a Density contour plot:

```
# Default Matplotlib Cubehelix version:
sns.palplot(sns.color_palette("cubehelix", 8))
```



Default Seaborn Cubehelix version:
sns.palplot(sns.cubehelix_palette(8))

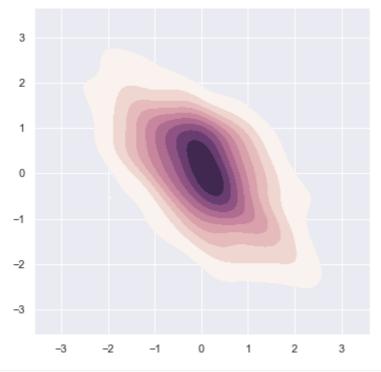


```
# Density Plot with Seaborn defaults:
x, y = np.random.multivariate_normal([0, 0], [[1, -.5], [-.5, 1]], size=300).T
sample_cmap = sns.cubehelix_palette(light=1, as_cmap=True)
sns.kdeplot(x, y, cmap=sample_cmap, shade=True)
```



C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: warnings.warn(

<AxesSubplot:>



□ Interactive widget to create a sequential cubehelix palette:

Let us now play with the parameters to have some fun and choose best parameters:

```
sns.choose_cubehelix_palette(as_cmap=True)
```

NameError Traceback (most recent call last) <ipython-input-1-230a1c9055e9> in <cell line: 1>() ----> 1 sns.choose_cubehelix_palette(as_cmap=True)

NameError: name 'sns' is not defined

Note that this app only works in this Jupyter Notebook as of now to help choose best parameters for our plot:

sns.palplot(sns.cubehelix_palette(n_colors=8, start=1.7, rot=0.2, dark=0, light=.95, reverse=Tru



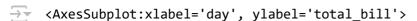
start is always between 0 and 3. rot an abbreviation for rotation is kept between -1 and 1. reverse

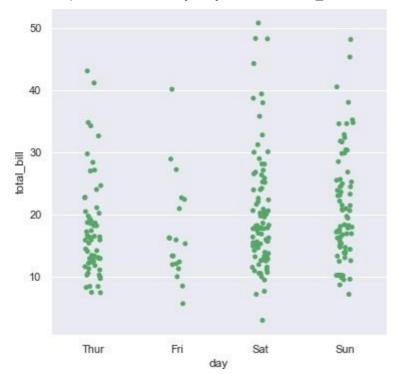
converses the color ordering and hue refers to plot appearance.

☐ Generic Seaborn Plots:

```
# Loading up built-in dataset:
tips = sns.load_dataset("tips")

# Creating Strip plot for day-wise revenue:
sns.stripplot(x="day", y="total_bill", data=tips, color="g")
```





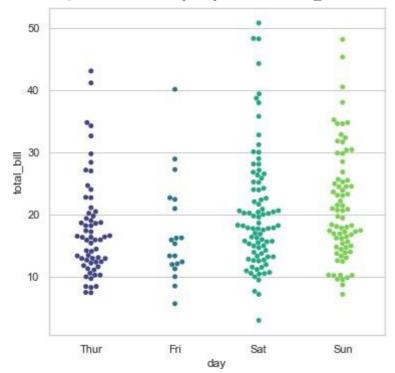
This does the job for us but let us try to get better results by plotting each day in different color instead of same color. For this, we shall replace color parameter with palette parameter:

```
# Set Theme:
sns.set_style('whitegrid')

# Creating Strip plot for day-wise revenue:
sns.swarmplot(x="day", y="total_bill", data=tips, palette="viridis")
```



<AxesSubplot:xlabel='day', ylabel='total_bill'>



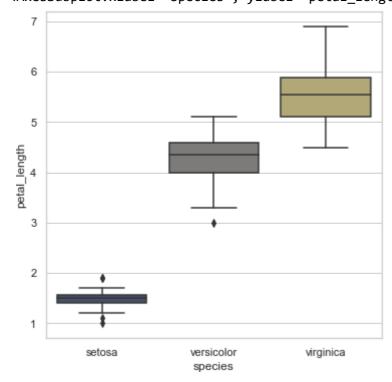
Similarly, let us plot one more and for a change, this time we shall plot a Violin plot:

```
iris = sns.load_dataset("iris")
```

sns.boxplot(x="species", y="petal_length", data=iris, palette="cividis")



<AxesSubplot:xlabel='species', ylabel='petal_length'>



There are multiple such palette available for us to play around with like magma,

5/15/24, 3:52 PM	Seaborn_Color_Palettes.ipynb - Colab
warm grey,	
gunmetal, dusky blue, cool blue, deep	o teal, viridian, twilight blue and many more. For
customized	

https://colab.research.google.com/drive/10QQQFqVCEV7_CCCg2Ywee-x8S3mr-Ulc#scrollTo=VHHcYJ3I0hHW&printMode=true

color brewing, we may also use color brewer that also offers interesting color palettes for working with Qualitative data. The cool thing about it is that you can use the an interactive Ipython widget function to make the selection of the palette. For this, you only need to use choose_colorbrewer_palette().

There are multiple such palette available for us to play around with like magma, warm grey,

gunmetal, dusky blue, cool blue, deep teal, viridian, twilight blue and many more. For customized color brewing, we may also use color brewer that also offers interesting color palettes for working with Qualitative data. A nice feature of the **Color Brewer website** is that it provides some guidance on which palettes are color blind safe.

The cool thing about it is that you can use the an interactive Ipython widget function to make the selection of the palette. For this, you only need to use choose_colorbrewer_palette(). To access this on your web browser, please access ColorBrewer link provided in the notebook.

I also found a nice representation of Color Schemes in Seaborn, that I found somewhere on web, so thought of sharing it in your Resource bucket to check out if you wish to. Let's have a look at it

LABSHEET 11

#Installation #pip install seaborn





Figure

It refers to the whole figure that you see. It is possible to have multiple sub-plots (Axes) in the same figure.

Axes

An Axes refers to the actual plot in the figure. A figure can have multiple Axes but a given Axes can be part of only one figure.

Axis

An Axis refers to an actual axis (x-axis/y-axis) in a specific plot.

Four sub-plots (Axes) in a single figure.



Seaborn

Seaborn - can create complicated plot types from Pandas data with relatively

simple commands Plotting in seaborn is either: Axes-level functions OR Figure-level

function

PLOT CATEGORIES IN SEABORN

- I. Relational plots: This plot is used to understand the relation between two variables.
- II. Categorical plots: This plot deals with categorical variables and how they can be visualized.
- III. Distribution plots: This plot is used for examining univariate and bivariate distributions
- IV. Matrix plots: A matrix plot is an array of scatterplots.

5/15/24, 3:58 PM Univariate, Bivariate Visualization.ipynb - Colab V. Regression plots: The regression plots in seaborn are primarily intended to add a visual guide that helps to emphasize patterns in a dataset during exploratory data analyses. Seaborn #Import necessary Packages import numpy as np import pandas as pd ${\tt import\ matplotlib.pyplot\ as\ plt}$ from matplotlib.pyplot import figure import seaborn as sns %matplotlib inline

5/15/24, 3:58 PM

```
#Simple Plotting with Seaborn
```

```
#Data
```

```
dates = ['1981-01-01', '1981-01-02', '1981-01-03', '1981-01-04', '1981-01-05', '1981-01-06', '1981-01-07', '1981-01-08', '1981-01-09', '1981-01-10']
```

min_temperature = [20.7, 17.9, 18.8, 14.6, 15.8, 15.8, 15.8, 17.4, 21.8, 20.0]

max_temperature = [34.7, 28.9, 31.8, 25.6, 28.8, 21.8, 22.8, 28.4, 30.8, 32.0]

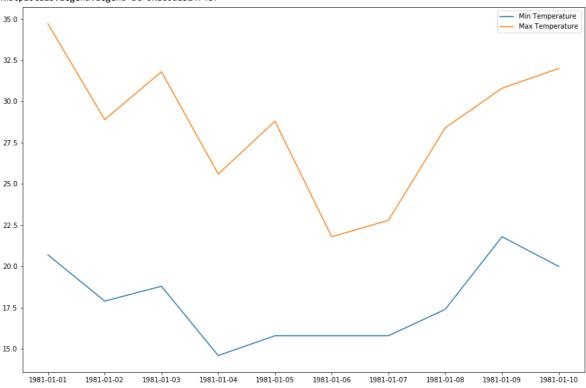
#Plotting

fig,axes = plt.subplots(nrows=1, ncols=1, figsize=(15,10))

axes.plot(dates, min_temperature, label='Min Temperature')
axes.plot(dates, max_temperature, label = 'Max Temperature')

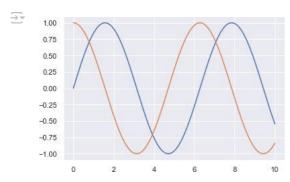
axes.legend()

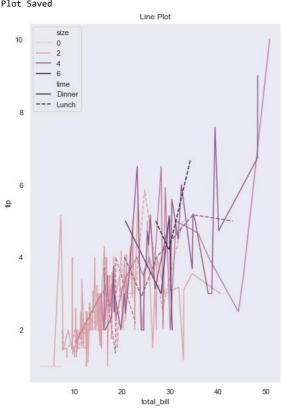
<matplotlib.legend.Legend at 0x1c0d2b24748>

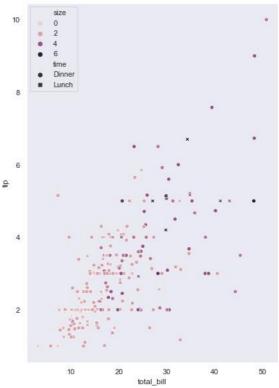


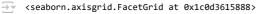
#seaborn style as the default matplotlib style
sns.set()

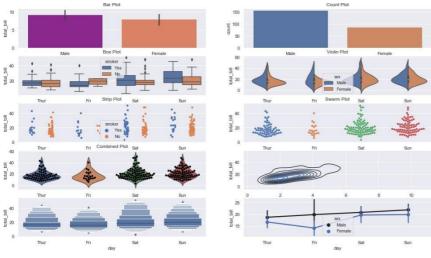
#Simple sine plot
x = np.linspace(0, 10, 1000)
plt.plot(x, np.sin(x), x, np.cos(x));

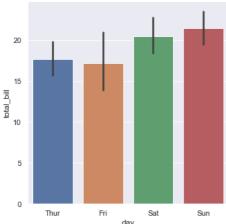












III. Distribution plots in seaborn is used for examining univariate and bivariate distributions. 4 main types of distribution plots:

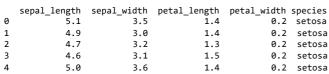
joinplot distplot pairplot rugplot

pairgrid = pairgrid.map_lower(sns.kdeplot)

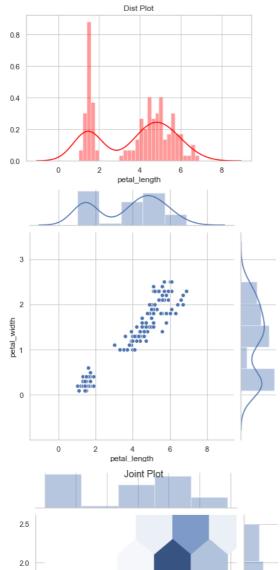
g = sns.PairGrid(df, diag_sharey=False, corner=True)

#Avoid Redundancy

g.map_lower(sns.scatterplot)
g.map_diag(sns.kdeplot)



<seaborn.axisgrid.PairGrid at 0x1c0d2b15888>



5/15/24, 3:58 PM	Univariate, Bivariate Visualization.ipynb - Colab	
		_
 https://colab.research.google.com/drive/14HjfqQhjXg7Agzhx8_FZ	Z_h0lBjtdzYt#scrollTo=pwW0AgpqR8mU&printMode=true 9,	49

□ LABSHEET 12

Load the Pacakges



To get started, open a Colab notebook and load the Pandas, Matplotlib, and Wordcloud packages.

import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from wordcloud import STOPWORDS

Mount the drive and read the CSV file from the drive.

Here we are going to use netflix_titles.csv dataset downloaded

from kaggle. Since it is text visualization we are going to consider

only one column.

from google.colab import drive

drive.mount('/content/drive/')

→ Mounted at /content/drive/

df=pd.read_csv('/content/drive/My Drive/Data/netflix_titles.csv', usecols=['cast'])
df.head()



- 1 Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...
- 2 Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...
- 3 Nai
- 4 Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...

Perform Prepeocessing to remove the records containing NaN

ndf=df.dropna()
ndf.head()

,

 \equiv

Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...

- 2 Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...
- 4 Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...
- 5 Kate Siegel, Zach Gilford, Hamish Linklater, H...
- 6 Vanessa Hudgens, Kimiko Glenn, James Marsden, ...

The wordcloud package requires single string instead of column.

Joining the all text data of the coloumn 'cast' to single string to make text visualization easy

text = " ".join(item for item in ndf['cast'])
print(text)

🚰 Ama Qamata, Khosi Ngema, Gail Mabalane, Thabang Molaba, Dillon Windvogel, Natasha Thahane, Arno Greeff, Xolile Tshabalala, Getmore

Sometimes, there will be words in your dataframe that are insignificant and don't add any insight, whe can take these out using the STOPWORDS module which is included in Wordcloud. Stopwards - set(\$1990845) Create a basic word cloud Treate a basic word cloud	5/15/24, 4:00 PM	TextVisualization.ipynb - Colab	
Create a basic word cloud	Sometimes, there will be words in your datafr	rame that are insignificant and don't add any insight. We can take these out	using
Create a basic word cloud	the STOPWORDS module which is included in	Wordcloud.	
Create a basic word cloud			
	stopwords = set(STOPWORDS)		
Note: the supposed of the supp	Create a basic word cloud		
The county respect to the county of the coun			
55			
The first decrease of the control of			
55			
55			
55			
5.5			
5.5			
25.5			
This first the page of the state of the stat			
titus vividalu sessari i monte contitussi ilimi (OCRITIVino) MASSOVAN-OE TUVOS Tita villa			
ultips // coloin respect to conde consideration (COR) ELWing MISSROVANOE ELWors Tils confidence to the condensate of the			
withs (figured) assessed to consider considerate (Anni 100 PPH Minn) MH 1880 Web 100 F. HWW ST Risk consists of the consists o			
withs //walata respect to consider consistency (Armid OCP PHY (for LMH 1980 Web 1905 - 11 Wors Tills con NT regions to the web the wide substantial Marketings - 25			
titus //adad respend montile condition/though though OPH Mont MHSBOWANGE HWost Have reflected the world world wind the last respendition of the condition of th			
attins //orbeit respect to unrothe consisting / though OOPs II Winst MH 1980 Webs OFE HWW set West of the right flow while service Market ser			
withins //mobality respected to more than 1000 Pet Print MH 1980 Webs OF HWW meet the constitution with the activity and the constitution of the c			
tillins // relativesseuri nonnote com/dispe/thmi/DOPHTV(m) MHSROVAtivOF HWwst#scon#Topin Howardscotksvinddorks tong			
attos // mbsh ressearch monter com/drive/HontOOBrt Minnt MHSROWstroof-HWmsTite-military New			
attos://odah.ussaan.h.monte.com/dive/tkuuloOPHTY/md.MHSROVAhOF HWmsTiscon/HTresind-Havddws-ut&winddods-hous			
ntins://milatureseanturondergon/thise/thoulooptity(mt MHSBOWAtroff HWmsTilsrontTresiontHoustMswitsprintModestone 995)			
ntins://orbidy-pssparch-normalis-com/chiop/fkmdOOPrt-Yfmal-MHSROWAhrOF-HWms-Tiss-rollTracjov-Househoks-ut-ReviewMeater-ma			
ntins://milab.pesparch.com/deige/tkontiOOPt11V/mil.Mt1SROWahrOF_thWmsT//scrudiff-eigent-flowybhs-wik-minthly-das-brus-			
75 Titres // anisals respersed to manufaction of the man of the ma			
attos //ocksh responde com/thive/fkmπΘοΡεΙΤΥίασΙ ΜΗSΒΟ∀ΑΙαΘΕ-ΗΥΜαςΣ#εγναθΕτροίασΗ ΘυννένΝευνθευνίαθανουσουσο			
1885 //milah respansh muutle convictiive/tkmulOOPtHYfnul MHSBOVAhnOE HWmsT#scmdfTn=jnul t0wyt-Newt8-vinithurba-tuna			
nthos //colado resseancto novorbe convictive/14kmil00Prt PYmnt MHSROVAhroF HWmsT#scrodfTre-innt Howelineswik novinthering			
ntins://collab.research.novode.com/chive/tkroutOOPd M/ford MHS8OVAhoOF HWmsT#scroutTre-ingl+towybNswt8roindMyda=hoa			
nttos://collab.research.google-com/duive/1kmul00Prt P/ford MHSR0VAhr0F-HWmsT#scrodtTr⊷ind t0wvbNsv/t8-vindMrds=trus			
ntios //colab.research.google.com/trive/1kmul00Ptt P/foot Mt ISBOVAbr0F_HWmsT#szynitTv=jrmt+0wvbNswt8zwjntMrvta=trug			
titos //collab research noonde com/drive/1kmπlOOPrt Yfnot MHSBOVAkoOF - HWmsT#srmtTn=innt towvbNswt8.vintMrvtα=trus			
titos //culab research noonde com/drive/1kmπlOOPrt Yfrot MHSBOVAkoOF - HWmsT#srmHTn=inrt How√bNswt&rvintMrvts=trus 956			
titos //culab research noonde com/drive/1kmπlOOPrt Yfrot MHSBOVAloOF -HWmsT#srmHTn=inrt How√bNswt8-wintMr√ls=trus 956			
ulius //collab research popule com/drive/1kmulOOPrHYfrod MHSBOVAInOF HWmsT#srrollTn=innHow+bNswt&nrintMrvts=hns			
ulius //collab research popule com/drive/1kmul00PrHYfrol MHS80VAlv0F-HWmsT#srrollTn=innH0wvhNswt&nrintMrvts=hns			
utios://collab.research.pppode.com/drive/1kmul00PrHYfrot MHS80VAIn0F_HWmsT#srrottTn=innHowshNewt&nriotMrvtg=trus			
attos://catab.research.gogode.com/drive/1kmul00Prt.lYfnot.MHSBOVAIn0F_HWmsT#srmftTn=innt-l0wvt-Niswt8-mintMnvts=trus			
CONTRACTOR OF THE SECOND CONTRACTOR OF THE CONTR	hitting (freeligh response) response response (the second	H MH ISBOWA IN OF 11W/ms-T//scrapHTr=innel forms I bloom 18 crimitA-mineral	,51

By instantiating WordCloud and then appending generate(text), we can pass in our big list of words and WordCloud will calculate the word frequencies, and determine the sizes, and colours of each of the words shown based on their frequencies within the text.

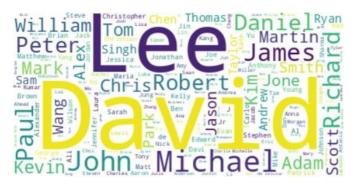
The other bits of Matplotlib code turn off the axes and ticks to make the word cloud look a bit neater.

```
wordcloud = WordCloud(background_color="white").generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.margins(x=0, y=0)
plt.show()
```



plt.axis("off")

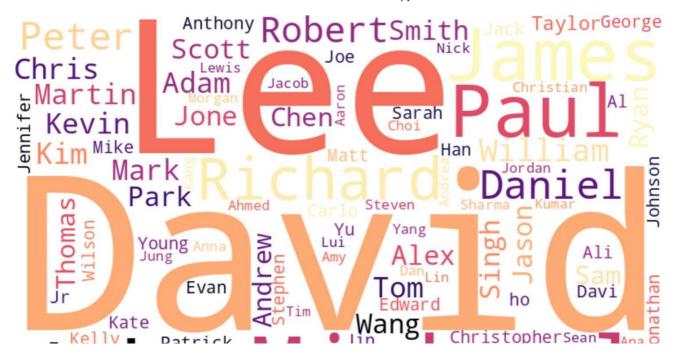
plt.show()



wordcloud = WordCloud(background color="white

```
max_words=100,
                      max font size=300,
                      width=800.
                      height=500,
                      colormap="magma'
                     ).generate(text)
plt.figure(figsize=(20,20))
plt.imshow(wordcloud, interpolation='bilinear')
plt.margins(x=0, y=0)
plt.savefig("cloud.jpg", format="jpg")
```





□ LABSHEET 13

A time series is the series of data points listed in time order.

A time series is a sequence of successive equal interval points in time.

A time-series analysis consists of methods for analyzing time series data in order to extract meaningful insights and other useful characteristics of data.

For performing time series analysis download stock_data.csv

import pandas as pd

import numpy as np

import matplotlib.pyplot as $\overline{\text{plt}}$

reading the dataset using read_csv
df = pd.read_csv(r"stock_data.csv")
displaying the first five rows of dataset
df.head()

$\supseteq_{}^{\downarrow}$		Date	Open	High	Low	Close	Volume	Name	
	0	1/3/2006	39.69	41.22	38.79	40.91	24232729	AABA	
	1	1/4/2006	41.22	41.90	40.77	40.97	20553479	AABA	
	2	1/5/2006	40.93	41.73	40.85	41.53	12829610	AABA	
	3	1/6/2006	42.88	43.57	42.80	43.21	29422828	AABA	
	4	1/9/2006	43 10	43.66	42 82	43 42	16268338	AARA	

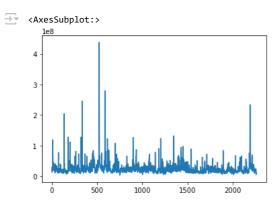
We have used the 'parse_dates' parameter in the read_csv function to convert the 'Date' column to the DatetimeIndex format. By default, Dates are stored in string format which is not the right format for time series data analysis.

Now, removing the unwanted columns from dataframe i.e.

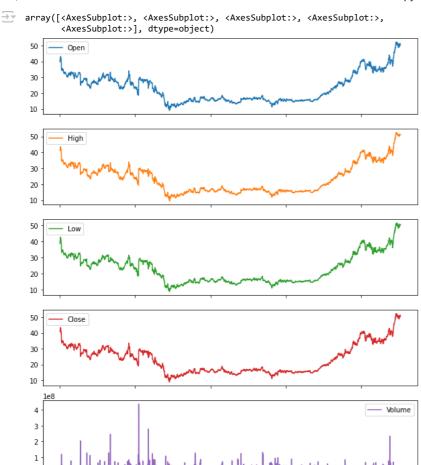
'Unnamed: O'. Example 1: Plotting a simple line plot for time

series data.

df['Volume'].plot()



Example 2: Now let's plot all other columns using subplot.



1000

Resampling: Resampling is a methodology of economically using a data sample to improve the accuracy and quantify the uncertainty of a population parameter. Resampling for months or weeks and making bar plots is another very simple and widely used method of finding seasonality. Here we are going to make a bar plot of month data for 2016 and 2017.

2000

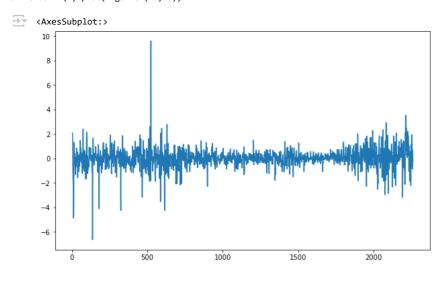
1500

Example 3:

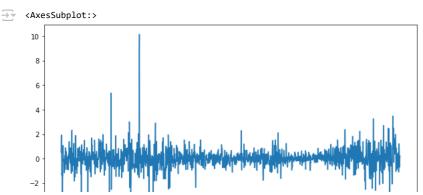
Differencing: Differencing is used to make the difference in values of a specified interval. By default, it's one, we can specify different values for plots. It is the most popular method to remove trends in the data.

df.Low.diff(2).plot(figsize=(10, 6))

500



df.High.diff(2).plot(figsize=(10, 6))

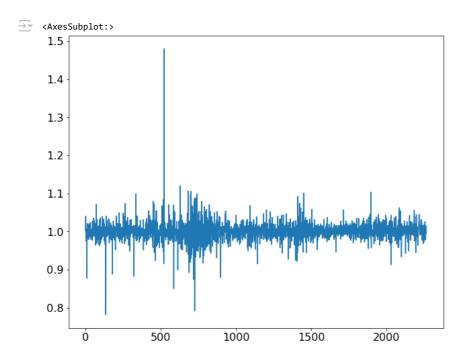


Plotting the Changes in Data

We can also plot the changes that occurred in data over time. There are a few ways to plot changes in data.

Shift: The shift function can be used to shift the data before or after the specified time interval. We can specify the time, and it will shift the data by one day by default. That means we will get the previous day's data. It is helpful to see previous day

df['Change'] = df.Close.div(df.Close.shift())
df['Change'].plot(figsize=(10, 8), fontsize=16)



.div() function helps to fill up the missing data

values. Actually, div() means division.

If we take df. div(6) it will divide each element in df by 6.

We do this to avoid the null or missing values that are created by the

'shift()' operation. Double-click (or enter) to edit