**Síi Jayachamaíajendía (Govt.) Polytechnic**

# Seshadíi Road, Bengaluíu-560001

**Depaítment of Computeí Science**



LAB MANUAL



**Aítificial Intelligence & Machine Leaíning**

**Course Code: 20CS51**



**INDEX**

|  |  |  |
| --- | --- | --- |
| **SL**  **no:** | **Question Name** | **Page**  **no:** |
| **1.** | **GitHub** |  |
|  | Account creation & Repository creation | 5 |
|  | Collaboration | 6-7 |
|  | Migration | 8-11 |
| **2.** | **Git** |  |
|  | Installation | 12-13 |
|  | Basic local Git operations | 14-18 |
|  | Branching and merging | 19 |
| **3.** | Array Aggregation Functions [NUMPY] | 20 |
| **4.** | **Vectorized Operations using NUMPY :**Vectorized Sum and Multiplication | 21 |
| **5.** | Use Map, Filter, Reduce and Lambda Functions on List using NumPy | 22 |
| **6.** | Using aggregation functions on a Data Frame | 22 |
| **7.** | Grouping using Pandas on a Data Frame | 23 |
| **8.** | Pivot and melt functions using Pandas. | 24 |
| **9.** | Use Map, Filter and Reduce, Lambda functions using Pandas df | 25 |
| **10.** | Time series using Pandas (resample, shift operations) | 26 |
| **11.** | Data visualization using Matplotlib (Bar ,pie, line ,histogram, scatter) | 27-28 |
| **12.** | Visualization of time series data using temperature on different days | 29 |
| **13.** | Visualization of Iris-dataset using Scatter Plot | 30 |
| **14.** | Visualization of Iris-dataset using Pie Chart | 31 |
| **15.** | Visualization of Titanic-dataset using Histogram | 32 |
| **16.** | Visualization of Titanic-dataset using bar chart | 33 |
| **17.** | Visualize Employee dataset using Line graph [Represent Salary &  Experience] | 34 |
| **18.** | Visualize Iris dataset using Box-Plot | 34 |
| **19.** | Pivot Table & Operations on it | 35-36 |
| **20.** | Operations on Pandas Series | 37 |
| **21.** | Perform the given operations on Car Manufacturing dataset | 38 |
| **22.** | Perform the given operations on performance of 32 automobiles dataset | 39-40 |
| **23.** | Perform the given operations on Ramesh’s Fitness dataset | 41 |
| **24.** | Perform the given operations on Stationaries dataset | 42-43 |
| **25.** | Linear Algebra [Vector , Scalar, Tensors, Matrix, Gradient, Eigen Values  and Vectors] | 44 |
| **26.** | Co-variance and Co-relation. | 45 |

|  |  |  |
| --- | --- | --- |
| **27.** | Univariate & Multivariate Distribution Plot | 46-47 |
| **28.** | Univariate & Multivariate Comparison Plots | 48-49 |
| **29.** | Univariate & Multivariate Composition Plot | 50-51 |
| **30.** | Multivariate Relationship Plot | 52 |
| **31.** | Detect missing values | 53 |
| **32.** | Replace | 54 |
| **33.** | Remove data objects [Rows] with missing values | 55 |
| **34.** | Remove the attributes [Columns] with missing values | 56 |
| **35.** | Estimate and impute missing values using arbitrary value | 57 |
| **36.** | Estimate and impute missing values using Mean value | 58 |
| **37.** | Estimate and impute missing values using Median value | 59 |
| **38.** | Estimate and impute missing values using Mode value | 60 |
| **39.** | Univariate Outliers | 61 |
| **40.** | Multivariate Outliers | 62 |
| **41.** | Time series outlier detection | 63 |
| **42.** | Perform the given operations on Titanic dataset | 64-66 |
| **43.** | Perform the given operations on Credit dataset | 67-70 |
| **44.** | Salary Prediction according to Experience. | 71 |
| **45.** | Marks Prediction according to Study Hours. | 72-73 |
| **46.** | Perform Data Preprocessing, Exploration, Splitting on Boston Housing  price dataset | 74-76 |
| **47.** | Perform Data Preprocessing, Exploration, Splitting on Cricket match  result dataset | 77-80 |
| **48.** | Perform Data Preprocessing, Exploration, Splitting on Performance of a  cricket player dataset | 81-83 |
| **49.** | Perform Data Preprocessing, Exploration, Splitting on Crop yield dataset | 84-86 |
| **50.** | Regression Algorithms [ Decision Tree, Random Forest, Support vector] | 87-88 |
| **51.** | Build decision tree-based model for Breast Cancer Wisconsin dataset | 89-90 |
| **52.** | Logistic regression [ using Fish prediction dataset ] | 91-92 |
| **53.** | SVM Model [Using Fish prediction dataset] | 93-94 |
| **54.** | Random Forest Classifier model [Using Breast Cancer dataset] | 95 |
| **55.** | K-means [ using Mall Customers Data] | 96 |
| **56.** | Dimensionality Reduction [PCA] using iris Dataset | 97 |
| **57.** | Shallow Neural Network | 98-99 |
| **58.** | Deep Neural Network | 100-101 |
| **59.** | Tokenization [Sentence & Word] | 102 |
| **60.** | N-Grams | 102-103 |
| **61.** | Frequency Distribution of Words | 103 |
| **62.** | Removing Stop-words | 104 |
| **63.** | Word-cloud | 105 |
| **64.** | Stemming & Lemmatization | 106 |

**Week-01**

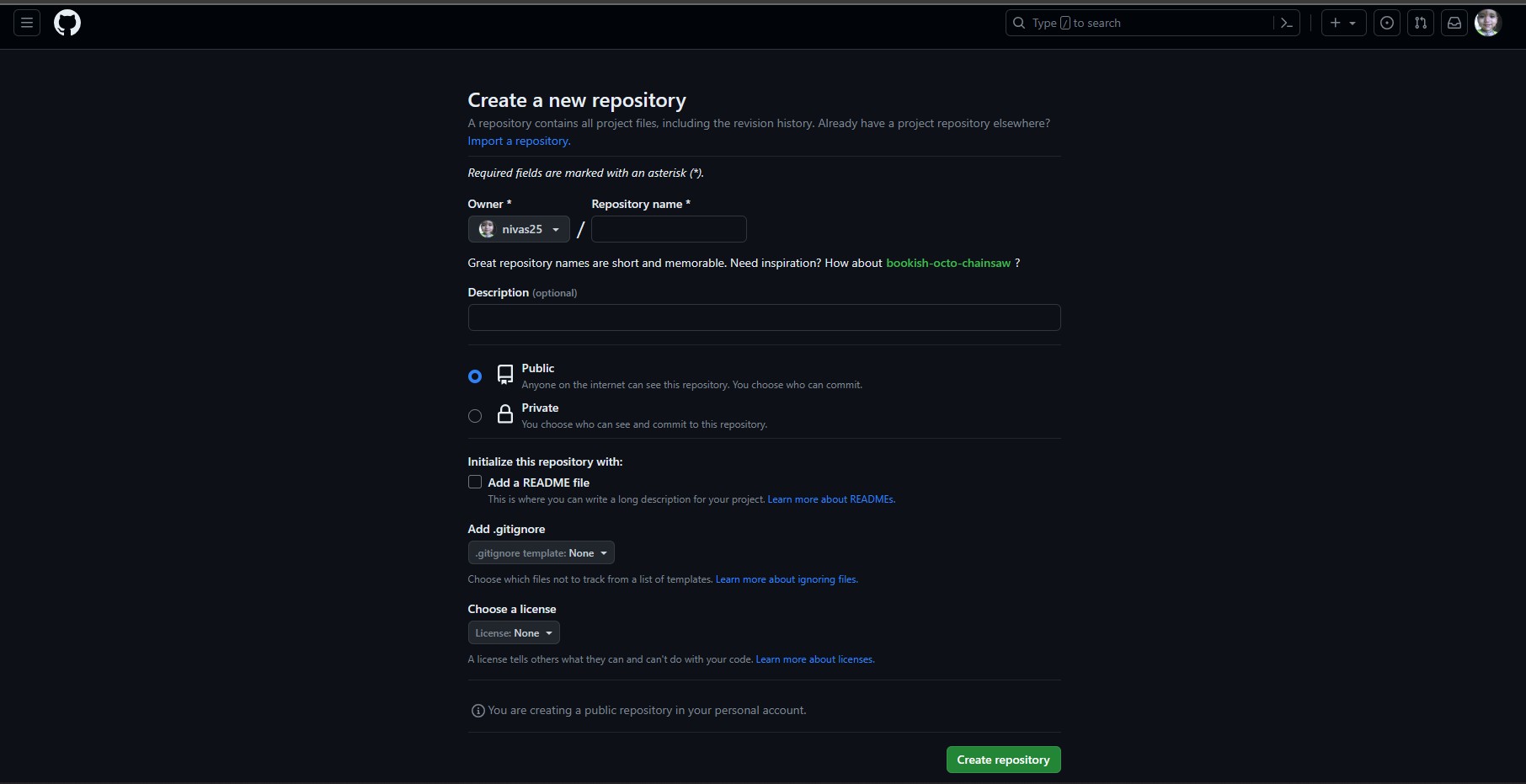
1. **GitHub**

**Account creation and configuration:**

* + Go to the GitHub website: [GitHub: Let’s build from here · GitHub](https://github.com/)
  + Click "Sign Up" in the upper-right corner.
  + Choose a plan (usually "Free") and provide a unique username, email, and password.
  + Verify your email address by clicking the link sent to your inbox.
  + Complete your account setup by adding a profile picture and configuring account settings.

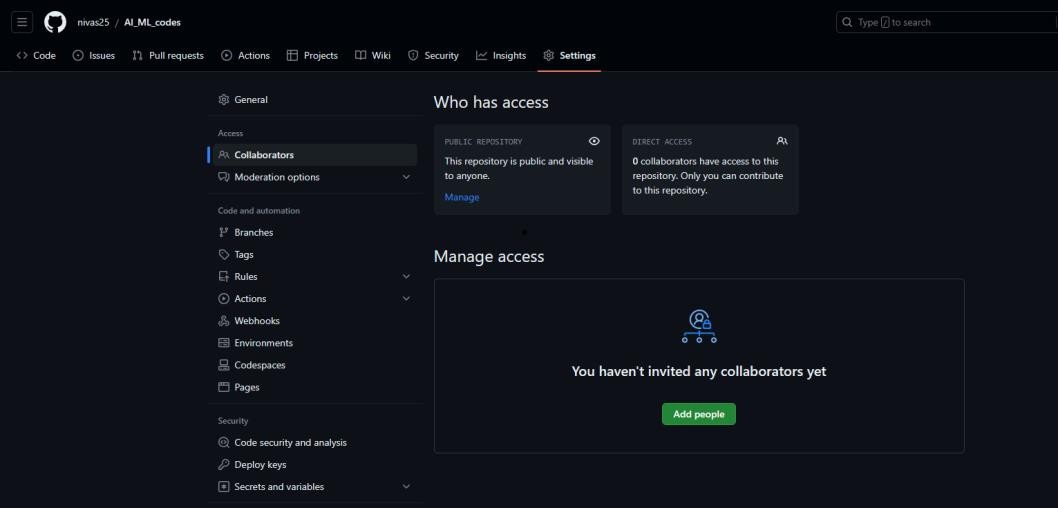
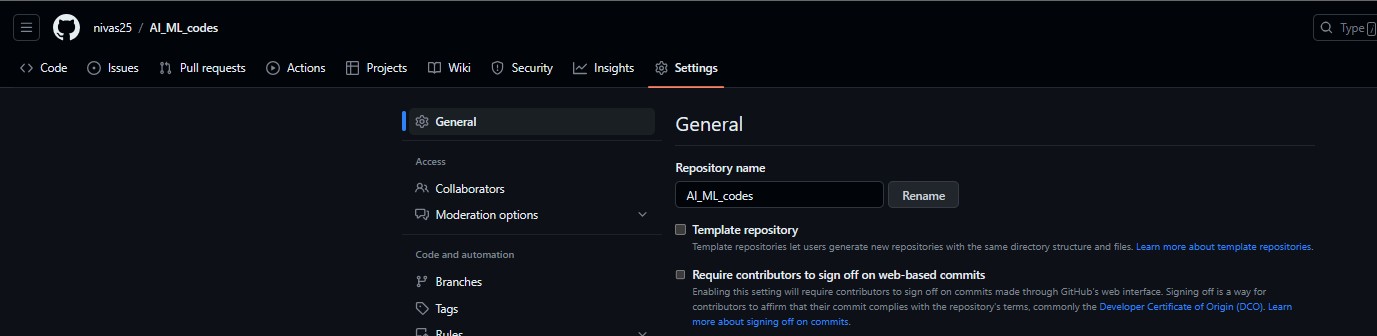
**Create and push to repositories:**

* Sign in to your GitHub account.
* Click on the "+" icon in the top-right corner and select "New repository."
* Enter a name for your repository, add an optional description, and choose the repository's visibility (public or private).

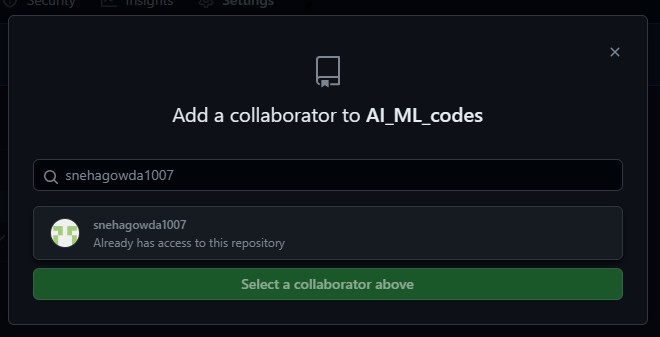


**Collaboration:**

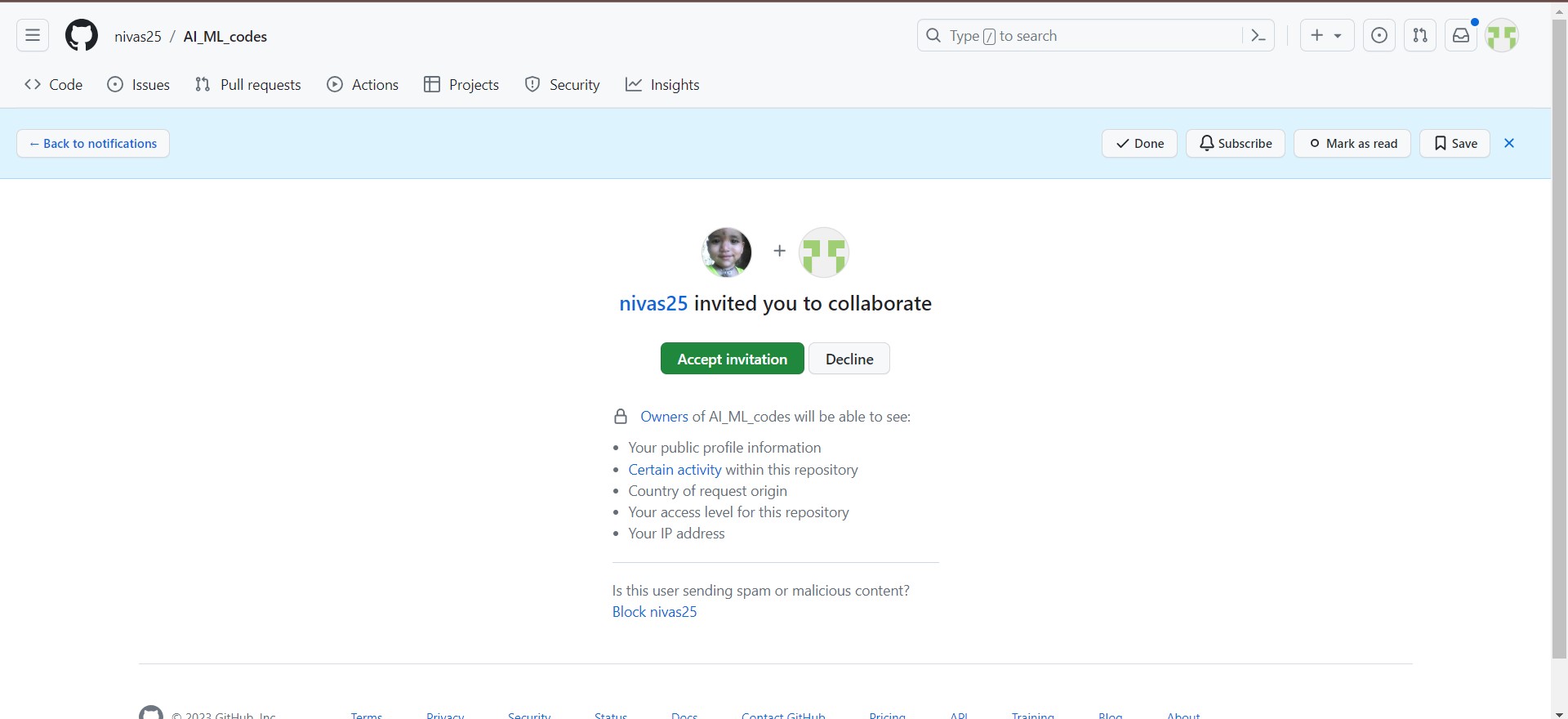
* Open your GitHub repository.
* Go to "Settings."
* Select "Collaborators" from the left sidebar.



* Add the collaborator's username or email.
* Click "Add" to send the invite.



* Collaborator accepts the invite.



* That’s all collaborator is added.



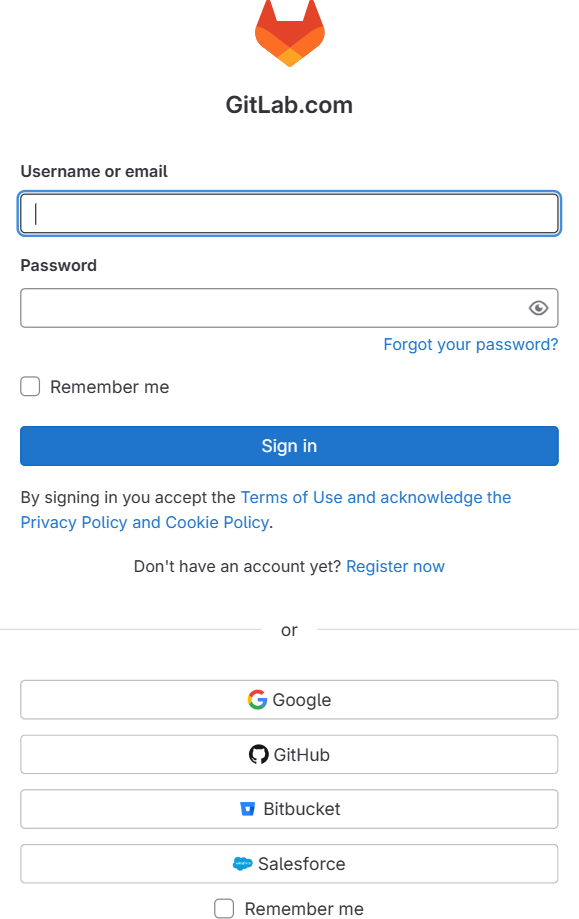
**Migration**

Migration, in a software development context, refers to the process of moving data, applications, or other resources from one environment, platform, system, or version to another.

* In this experiment we are moving an existing GitHub repository to new platform known as Gitlab

**Creation of Gitlab account:**

* Go to this website: [Sign in · GitLab](https://gitlab.com/users/sign_in)



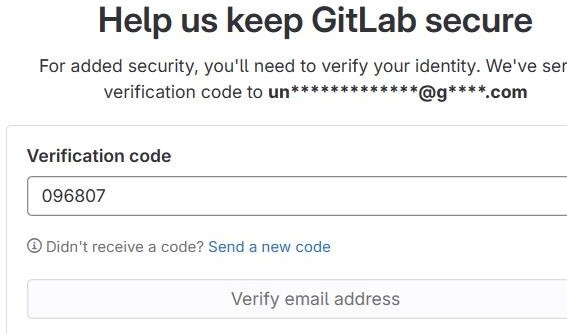
* Login using google account [Use the same account which is used in GitHub]

## Agree to the Terms:

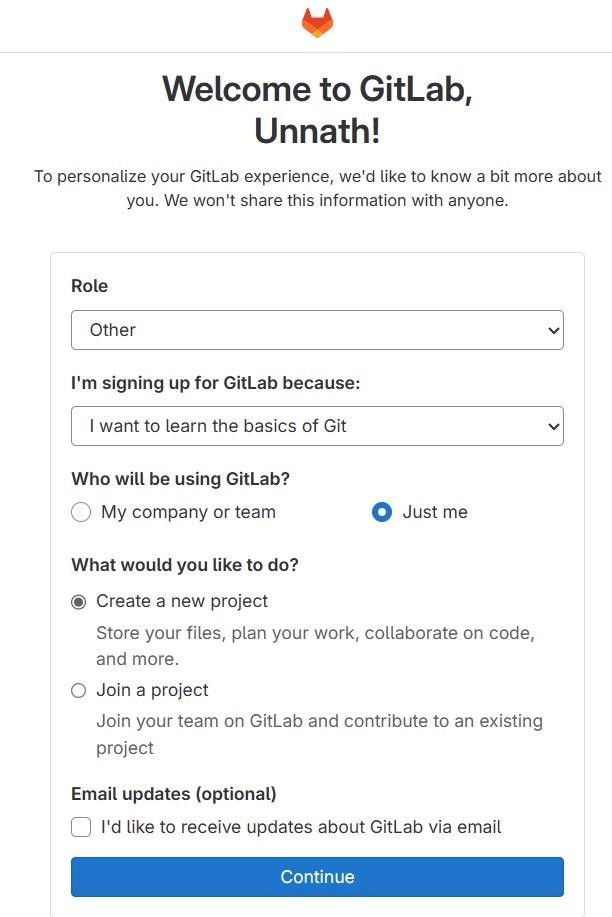
Read through GitLab's Terms of Service and Privacy Policy, and if you agree, check the box indicating your acceptance.

## Verify Your Email:

GitLab will send a verification email to the address you provided during registration. Open your email inbox and look for the verification email. Click the link in the email to verify your email address. This step is important to ensure that you receive notifications and can reset your password if needed.

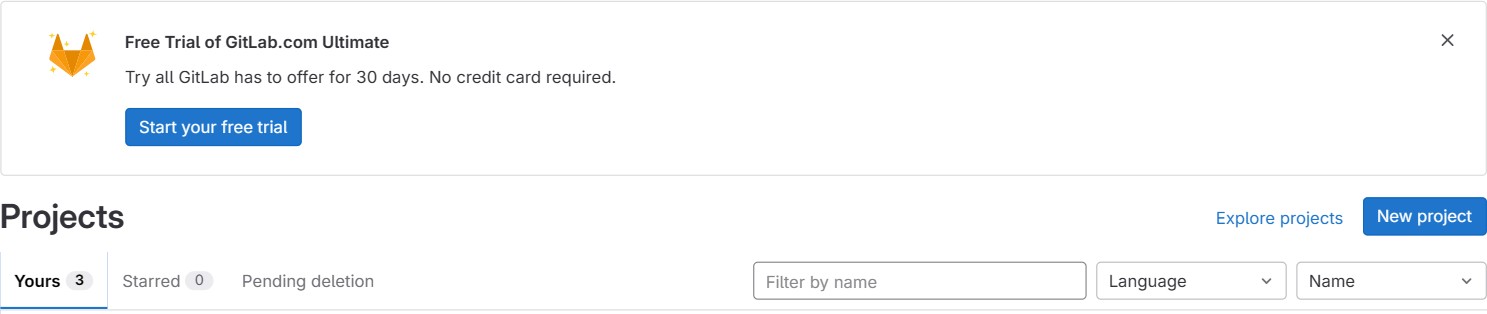


## Set Up Your Profile:

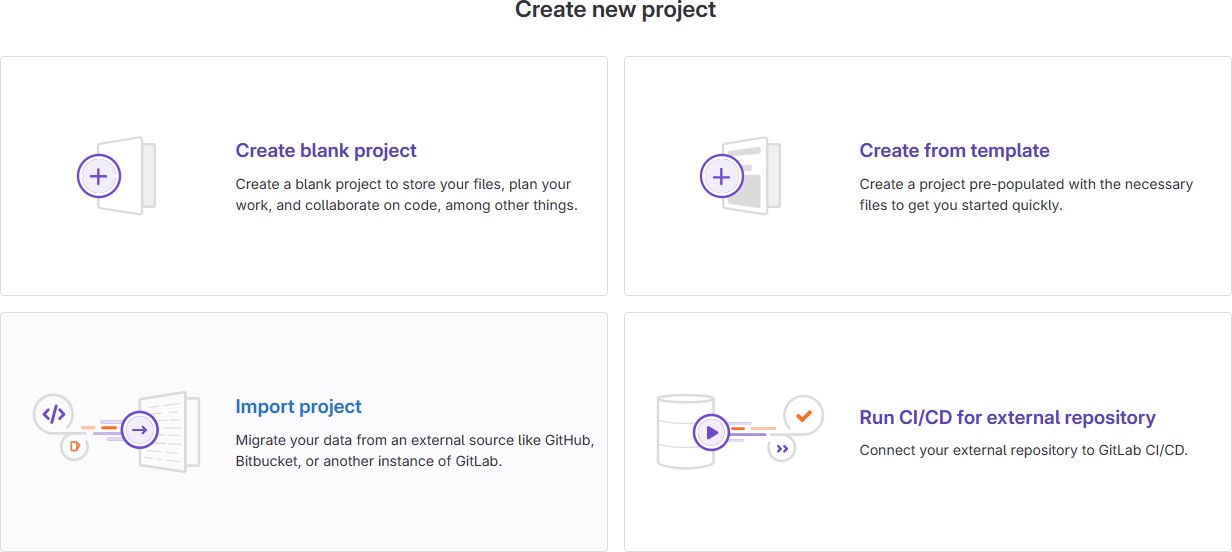


**Importing Repository from GitHub to Gitlab**

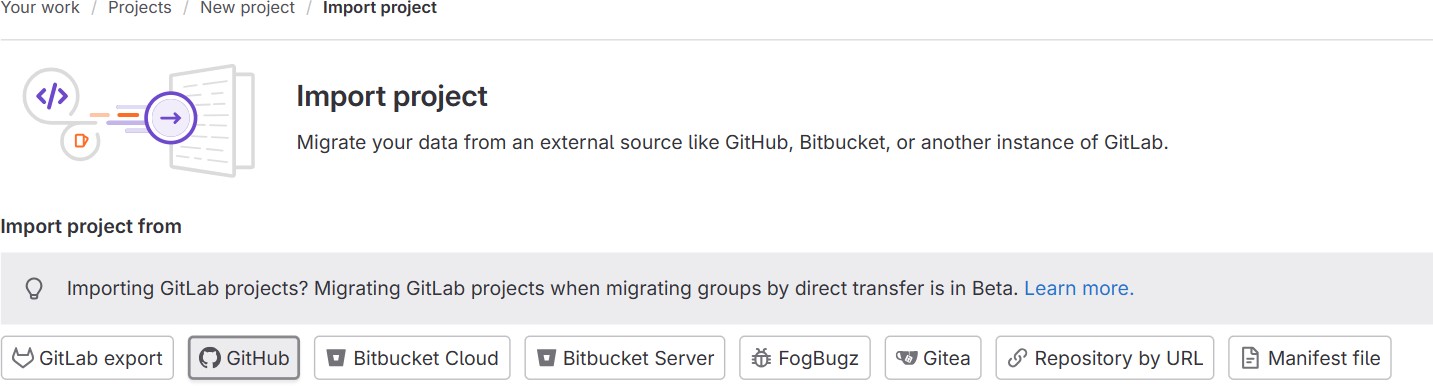
* Restart the browser and go to [Projects · GitLab](https://gitlab.com/)
* Click on new project



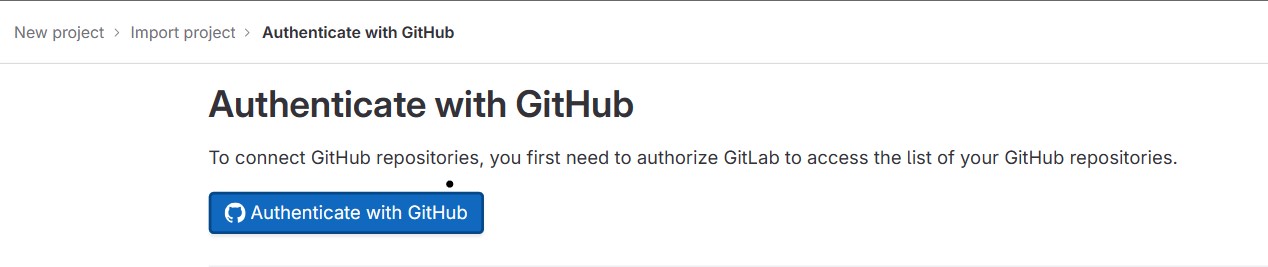
* Click on Import projects to import a repository from GitHub to Gitlab



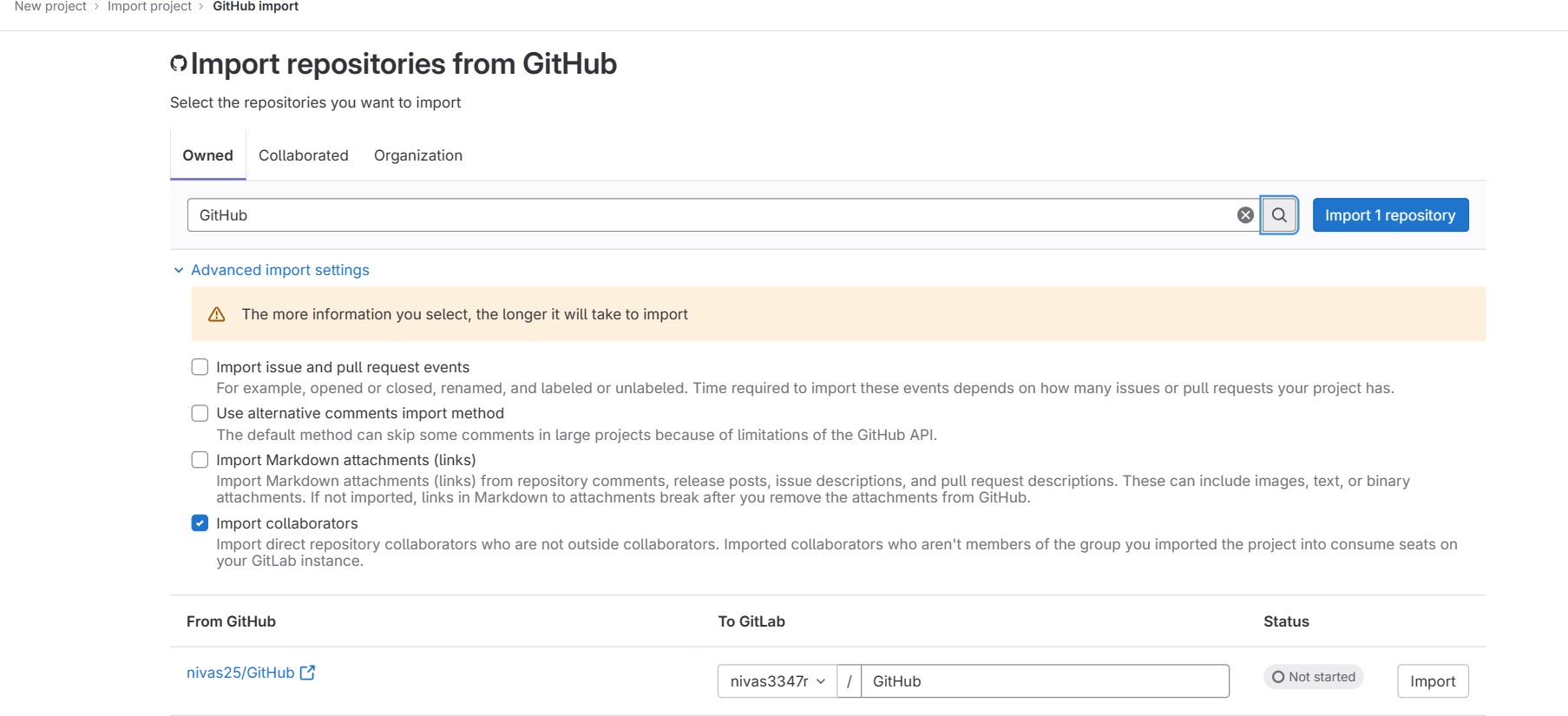
* Click on GitHub to connect your GitHub account to Gitlab



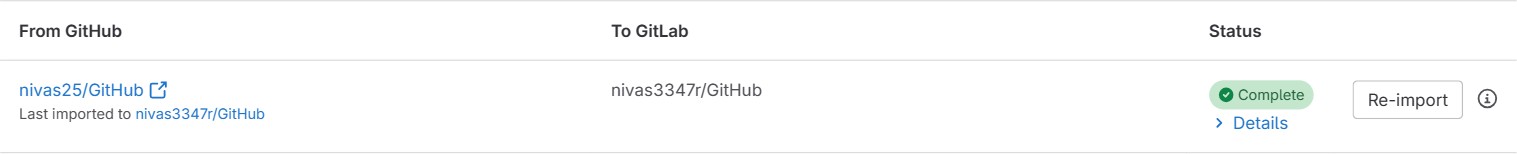
* Authenticate with GitHub



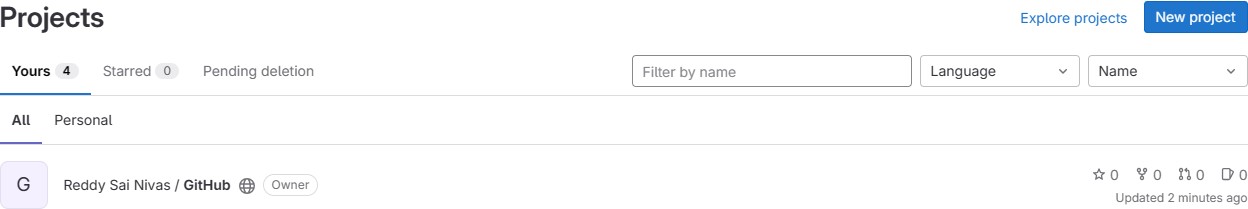
* If you’re GitHub account email and Gitlab email is same it connects without any problem
* Type the repository name which u need to import and click on search icon



* Click on import repository.
* Wait for few seconds to complete the import process after that it shows completed if any errors don’t occur.



* Check in projects to view the imported repository.



**Video Sources:**

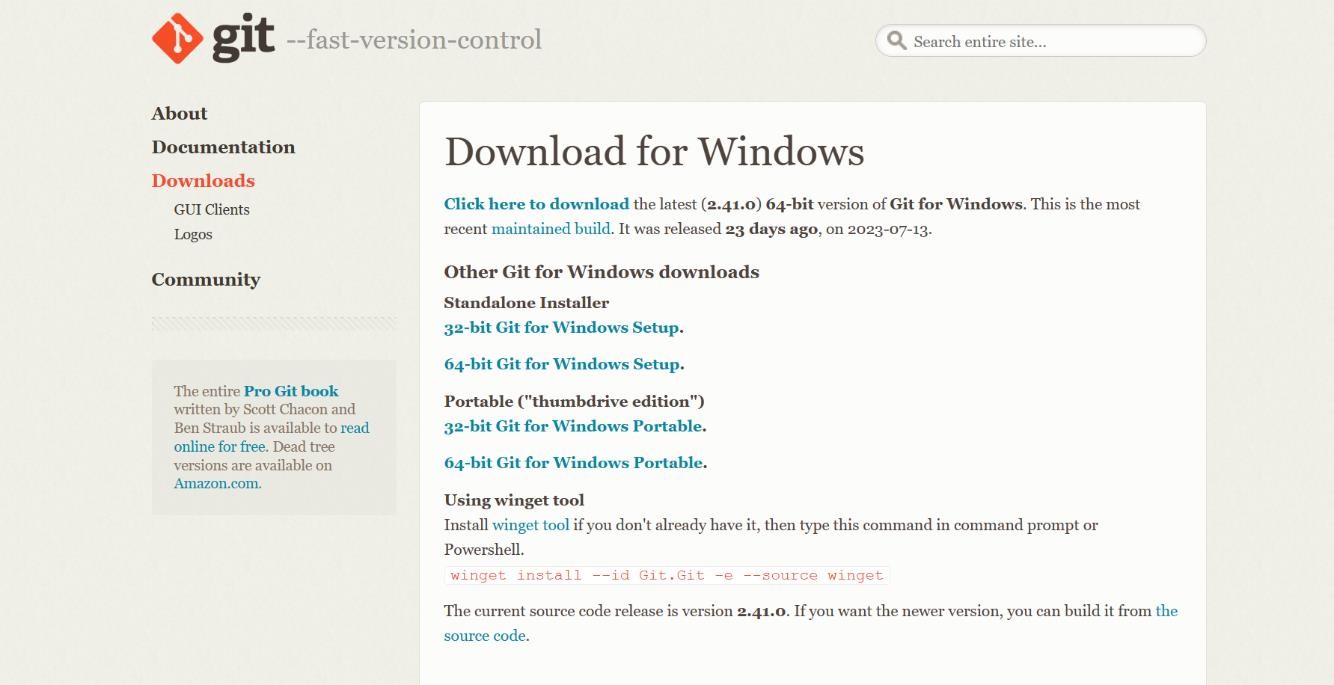
[nivas25/Git\_GitHub\_Videos](https://github.com/nivas25/Git_GitHub_Videos)

## Code Links:

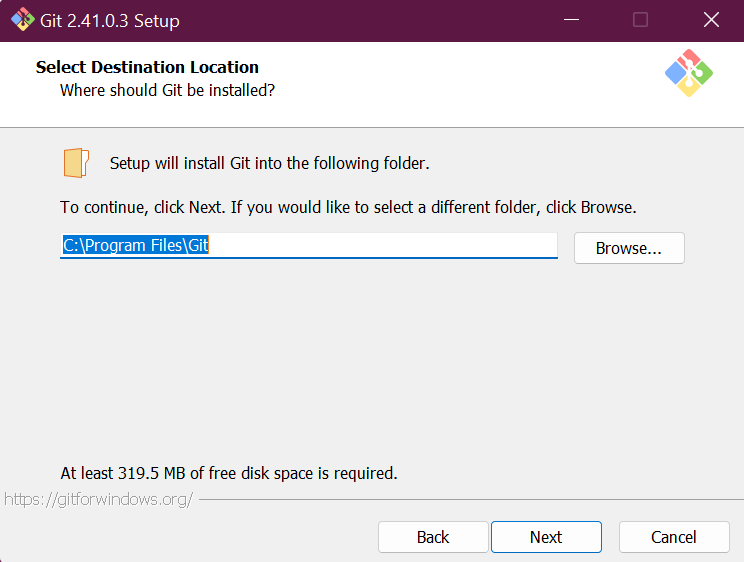
[github.com](https://github.com/nivas25/AI_ML_codes)

## Git installation and setup Installation:

* + Download the Git installer for Windows from [Git - Downloads (git-scm.com)](https://git-scm.com/downloads) .
  + Download the 64bit Windows Setup.



* + Open the git installer .exe file and start installation
  + A git installer window opens and click on next and select installation destination if it doesn’t show default installation location.



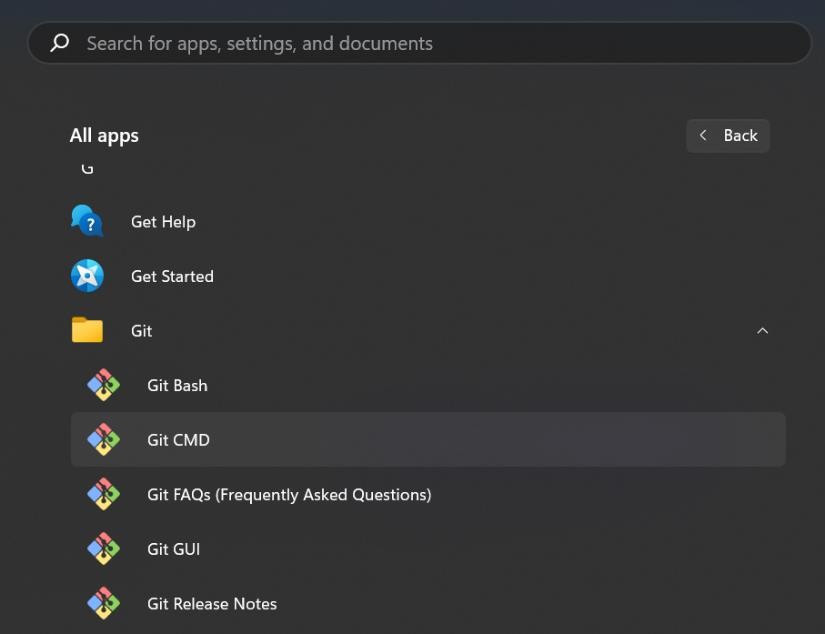
* + Be clicking on next button until install button appears and click on “Install” to complete the installation.
  + That’s all GIT is ready for working!!

**Visual Studio Code Installation:**

* + Download the VS code installer from [Visual Studio Code - Code Editing. Redefined](https://code.visualstudio.com/) .
  + Open the vs code installer .exe file and start installation
  + Run the installer and follow the on-screen instructions with default settings.

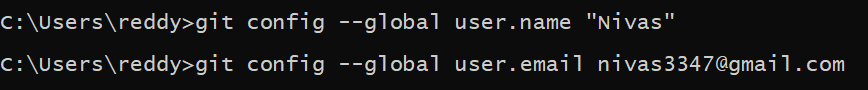
## Git Configuration:

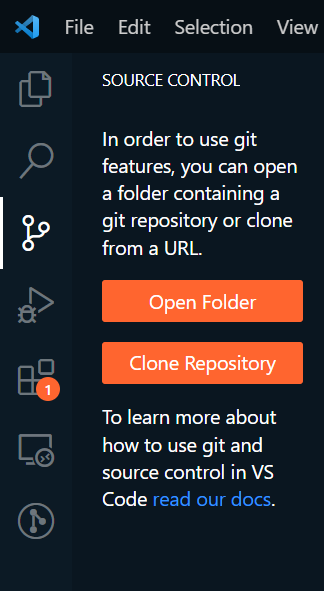
* + Open the Git CMD which will be there in git folder and it appears in Start



* + Run the commands to config the git, enter a username and email id git config --global user.name “yourname”

git config --global email “your\_email\_id”



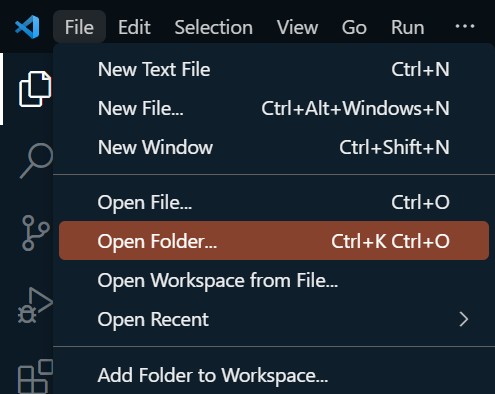
* + If successfully configed the git , u can check it by opening VS code and by clicking on source control which is the left side of panel

**Basic local Git operations**

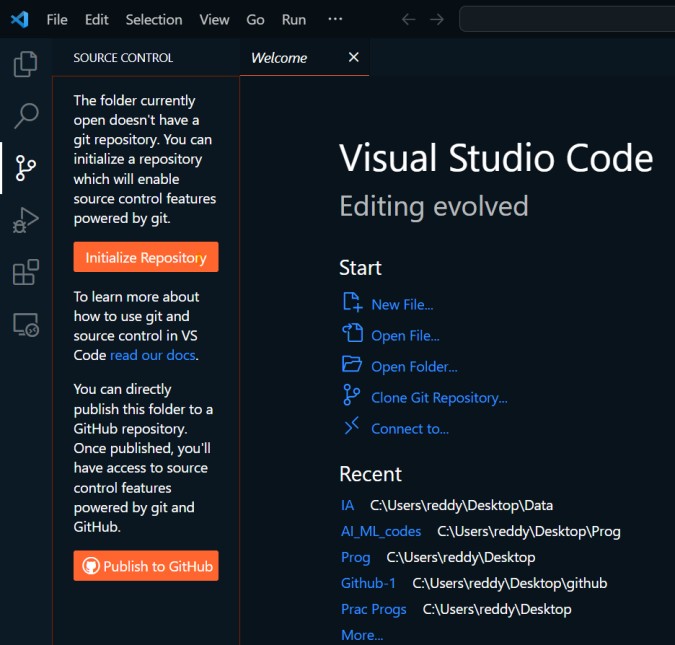
* + Local Git refers to the use of Git on your local machine, where you work directly with the Git version control system without the need for a remote repository or internet connection.

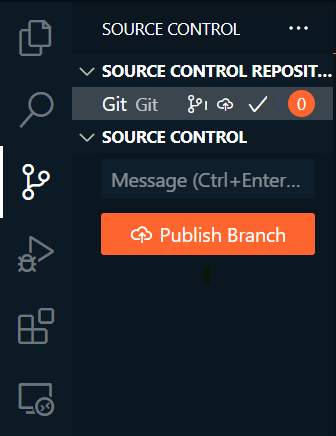
**Creating a repository:**

* + A repository, often abbreviated as "repo," is a central place where version control systems like Git store all the files, data, and information related to a project.
  + Create a Folder in desktop or any other location and name it. Open the VS code.
  + Open the Folder in VS code



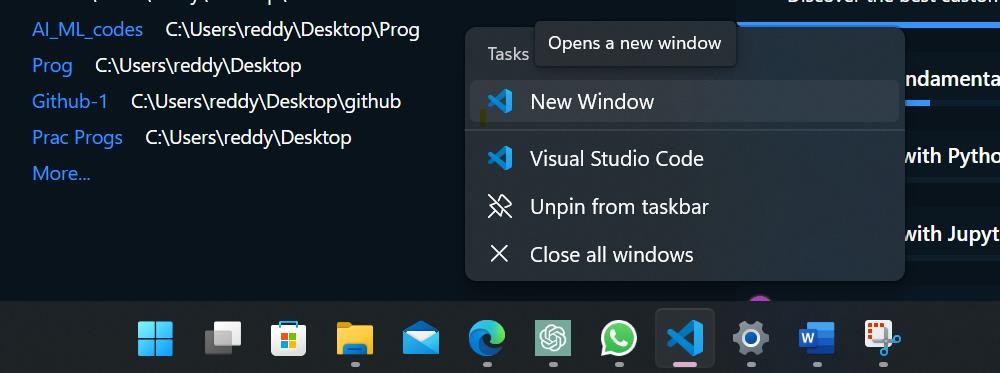
* + Go to Source  Control which is on left side of the panel.
  + Click on Initialize Repository for Initialize that folder to Git.



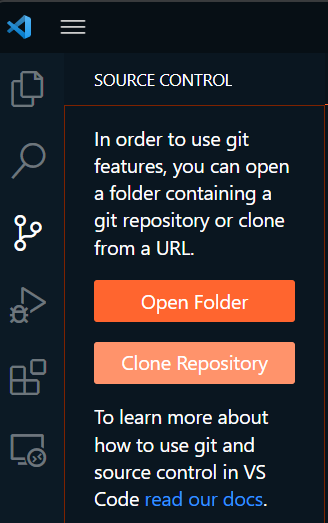
After successfully Initial of folder Source Control will appear like this--->

**Cloning a repository.**

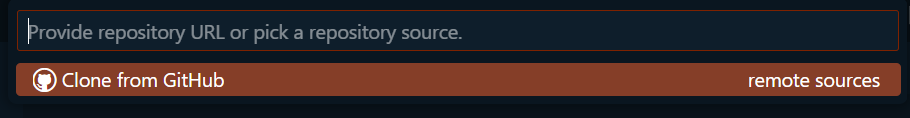
* + Cloning is the process of creating an exact copy of a Git repository from a remote source (such as a Git hosting service like GitHub, GitLab, or Bitbucket) to your local machine. When you clone a repository, you download all the files, commit history, branches, and metadata associated with the original repository.
  + Every time when u do a new thing in VS code u need to open a new Window.



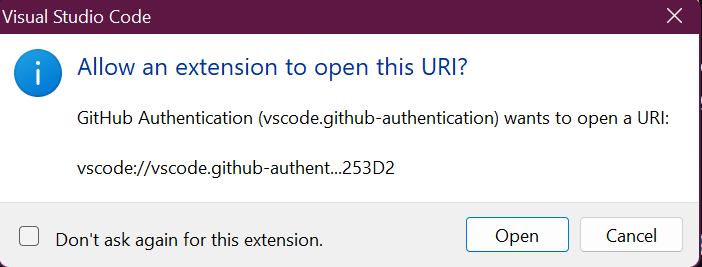
* + Click on Clone Repository in Source Control.



* + Click on remote sources.

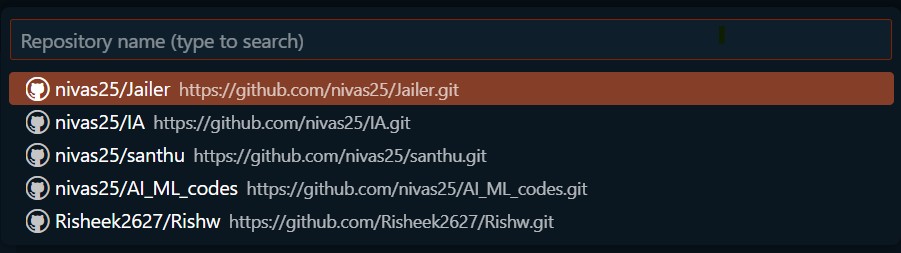


* + An Authentication Allow message appears in browser click allow again click on open in VS code Authentication.



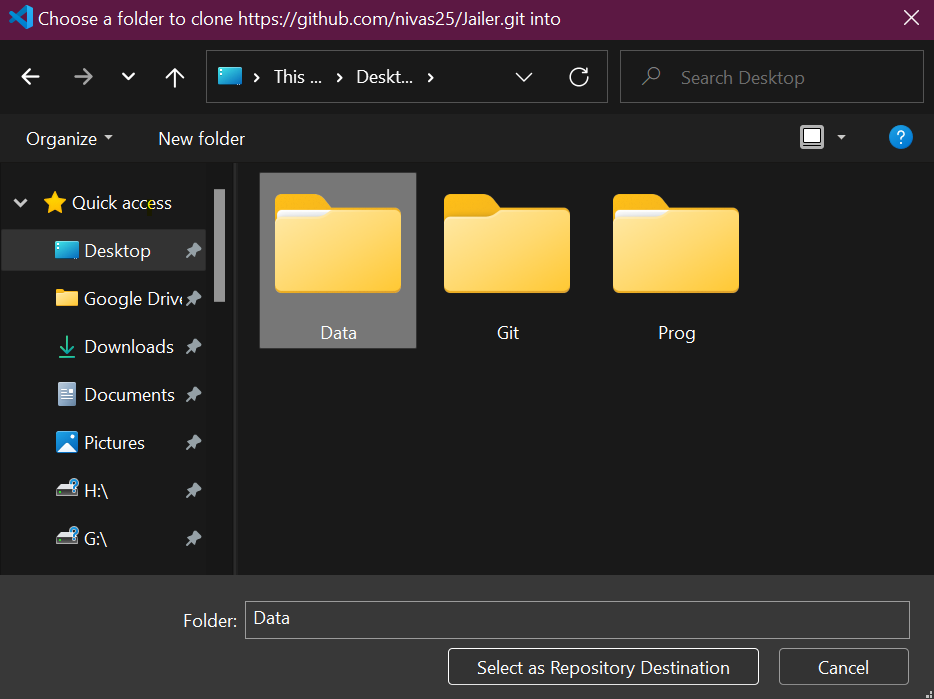
NOTE: u need to have GitHub account created using email id which we u used in git cmd configuration…

* + After successful authentication in browser by entering password and username of



GitHub account, in VS code this appears.

* + Click on the required repository need to cloned.
  + Select the Cloning Destination [USE a FOLDER for this process].



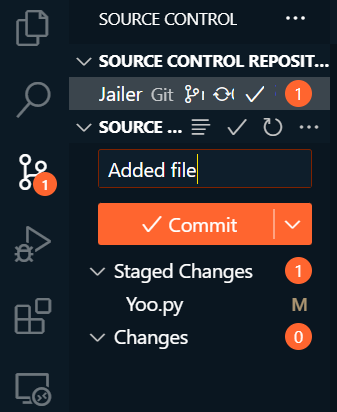
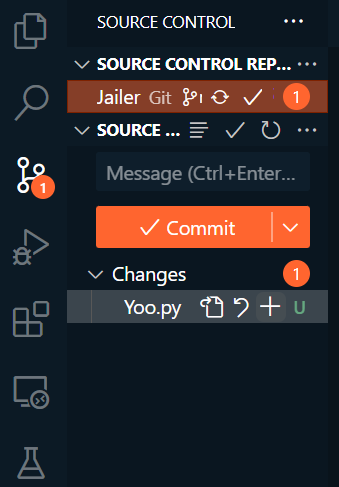
Cloning in Process ---

**Making and Recording changes:**

* + Here I am creating a new file named Yoo.py and saving it.
  + U can continue this experiment in previous window or new Window, it’s you’re wish.
  + For creating a new file click on create a new file in Explorer [left side of window]



* + In Source Control we can see in Changes tab a change is notified click on + to add it to Stagging place and give a commit message and click on “COMMIT”.

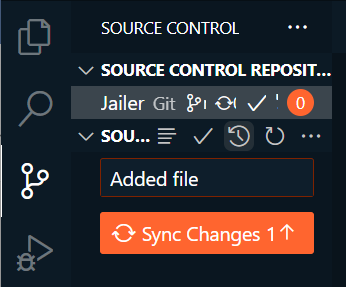
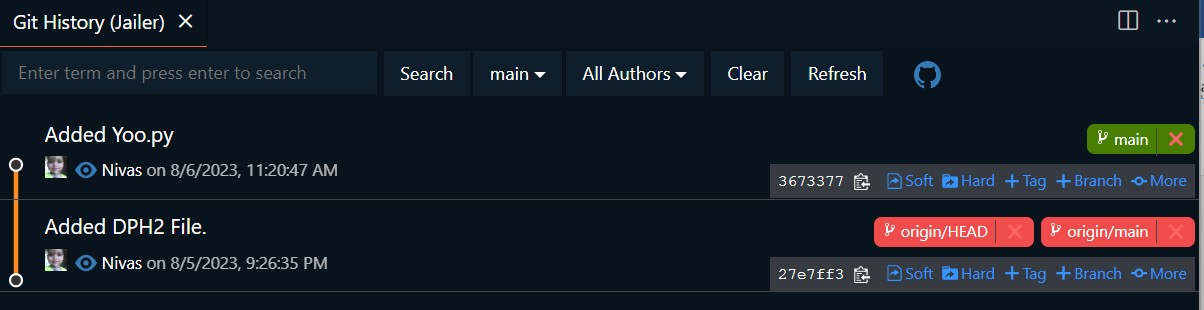


**Viewing the history of all the Changes:**

* + Go to Extensions and search for “GIT HISTORY” and click on install



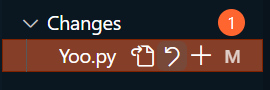
* + In source control click on clock like symbol to view the commit history.

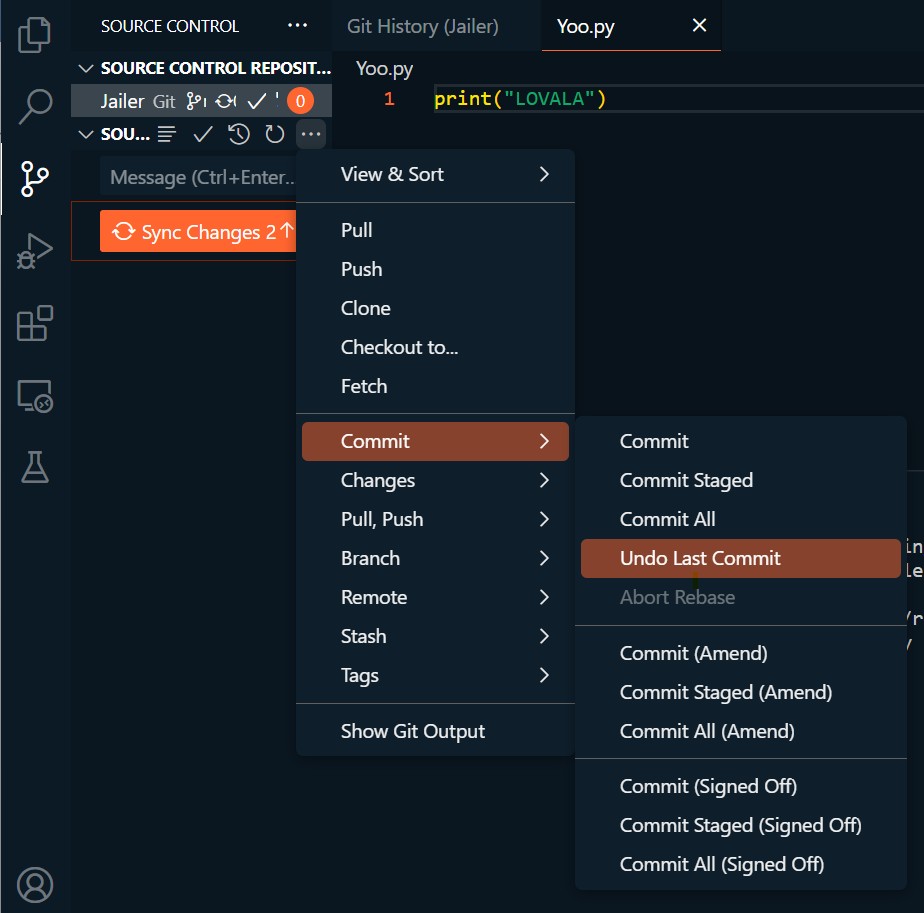
**Undoing Changes:**

* + **Undoing a change when it’s in changed stage**  In source control--- In Changed ----

click on discard Changes…



* + **Undoing the Commits**  In source control --- click on 3dots --- commit undo last

commit.

* + - By clicking remove from Stage and

discard changes it completely goes to previous commit stage.

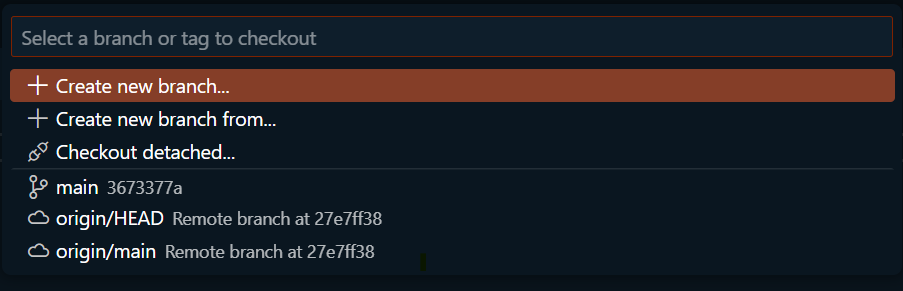
**Git Branching and merging.**

**Creating and switching to new branches**

* + Click on branch name [main] which is visible on left-bottom of window.

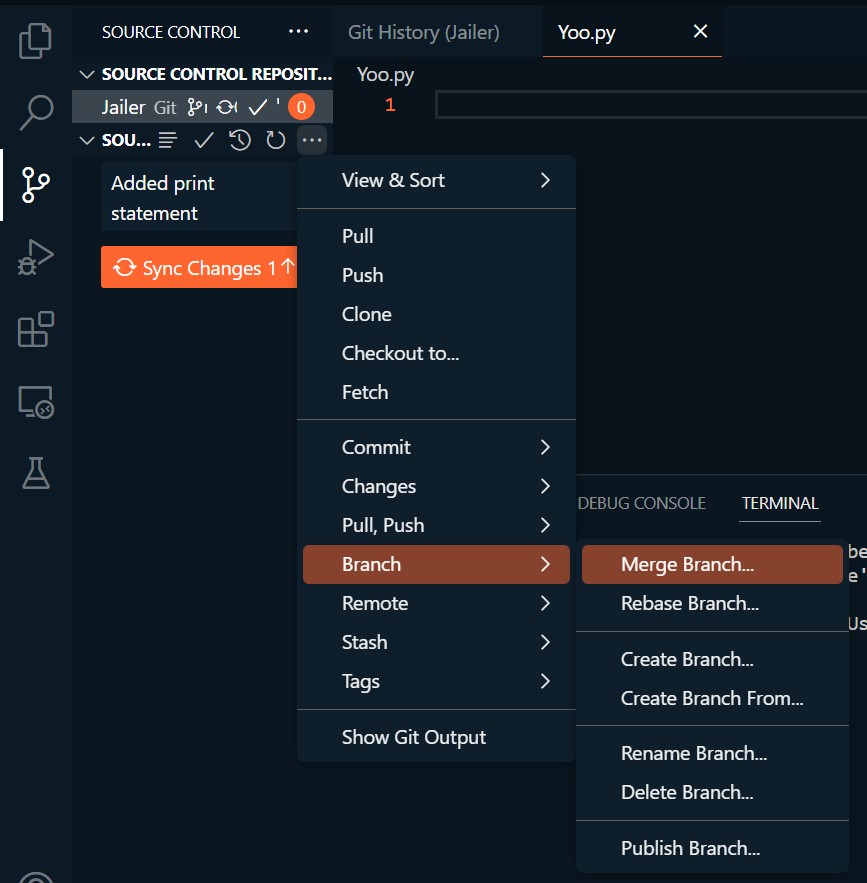


* + To create a new branch, click on create new branch and to switch between them click on their names.



**Merging local branches together:**

* + Select the branch you want to merge into the current branch
  + In source control --- 3 dots --- Branch --- Merge Branch
  + Resolve conflicts any and stage and commit the changes to complete the merge.



**Week 3**

1. **Array Aggregation Functions [NUMPY]**

**Code:**

import numpy as n a=n.array([54,78,32,46,89,76])

print("\nAn Array:",a) print("\nSum:",n.sum(a))

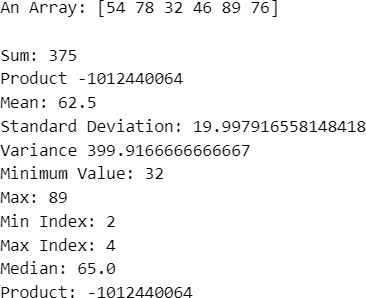
print("Product",n.prod(a))

print("Mean:",n.mean(a)) print("Standard Deviation:",n.std(a)) print("Variance",n.var(a)) print("Minimum Value:",n.min(a)) print("Max:",n.max(a))

print("Min Index:",n.argmin(a)) print("Max Index:",n.argmax(a)) print("Median:",n.median(a))

print("Product:",n.prod(a))

## Output:



1. **Vectorized Operations using NUMPY Vectorized Sum and Multiplication**

## Code:

**[Product]**

import numpy as np import timeit

np.a=[4,5,1]

print(np.prod(np.a))

print("Time taken by vectorized product : ",end= "")

%timeit np.prod(np.a)

total = 1

for item in np.a:

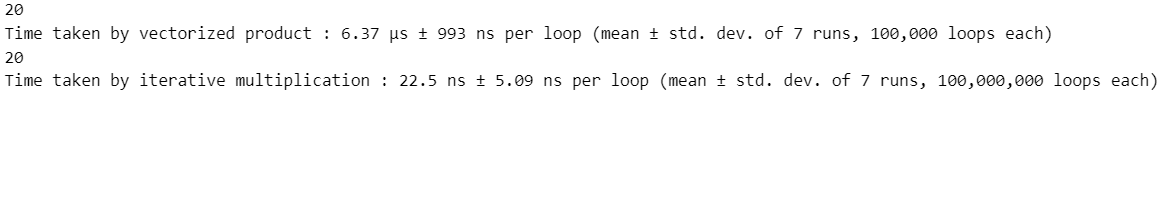
total =total\*item t = total

print(t)

print("Time taken by iterative multiplication : ",end= "")

%timeit t

**Output:**



**Code:**

## [Sum]

import numpy as n import timeit

print(n.sum(n.arange(4)))

print("Time taken to vectorized sum:")

%timeit n.sum(n.arange(4))

t=0

for i in range(0,4): t+=i

a=t print("\n"+str(a))

print("Time Taken by iterative sum:",end="")

%timeit a

## Output:

1. **Use Map, Filter, Reduce and Lambda Functions on List using Numpy Code:**

import numpy as n da=[60,8,7,5,34,78]

d=n.array(da)

from functools import reduce as r print(list(map(lambda num:num\*\*2,d))) print(list(filter(lambda num:num>2,d))) print(r(lambda x,y:x+y,d))

**Output:**



## Using aggregation functions on a Data Frame Code:

import pandas as p

d=p.DataFrame([[2,5,6],

[4,6,3],

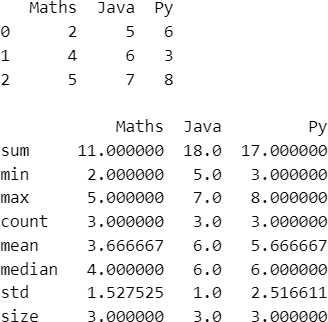
[5,7,8]],

columns=["Maths","Java","Py"])

print(d) c=d.agg(['sum','min','max','count','mean','median','std','size',]) print()

print(c)

## Output:



1. **Grouping using Pandas on a Dataframe**

**Code:**

import pandas as p

t={

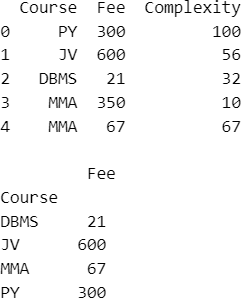
}

'Course':["PY","JV","DBMS","MMA","MMA"], 'Fee':[300,600,21,350,67], 'Complexity':[100,56,32,10,67]

d=p.DataFrame(t) print(d)

c=d.groupby('Course').agg({'Fee':'min'}) print("\n",c)

## Output:



1. **Pivot and melt functions using Pandas.**

## Code:

import pandas as p

t={

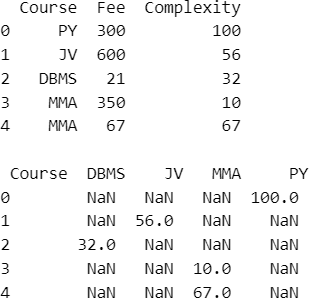
}

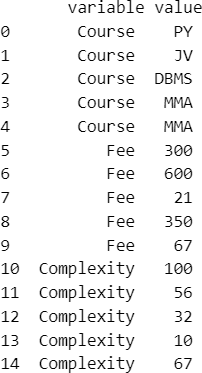
'Course':["PY","JV","DBMS","MMA","MMA"], 'Fee':[300,600,21,350,67], 'Complexity':[100,56,32,10,67]

d=p.DataFrame(t)

print(d) print("\n",d.pivot(columns='Course',values='Complexity')) print("\n",d.melt())

## Output:





1. **Use Map, Filter and Reduce, Lambda functions using Pandas [Data Frame] Code:**

import pandas as pd

from functools import reduce

data = {

'Numbers': [1, 2, 3, 4, 5], 'Letters': ['A', 'B', 'C', 'D', 'E']

}

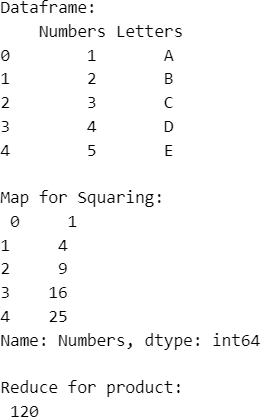
df = pd.DataFrame(data) sq=df['Numbers'].map(lambda x: x\*\*2)

ev=list(filter(lambda x: x % 2 == 0, df['Numbers']))

po = reduce(lambda x, y: x \* y, df['Numbers']) print("Dataframe:\n",df)

print("\nMap for Squaring:\n",sq) print("\nReduce for product:\n", po)

## Output:



1. **Time series using Pandas (resample, shift operations)**

## Code:

import numpy as n import pandas as p

d=p.DataFrame(

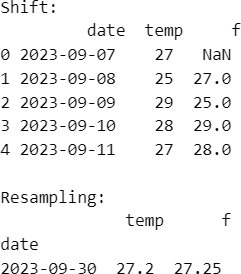
{"date":p.date\_range(start="2023-09- 07",periods=5,freq="D"),"temp":n.random.randint(18,30,size=5)}

)

d["f"]=d["temp"].shift(1) print("Shift:\n",d)

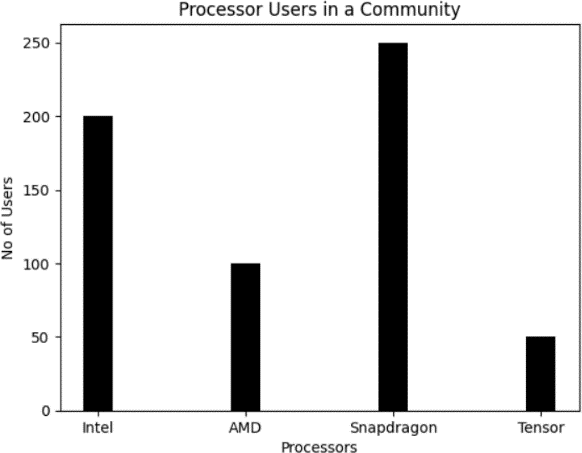
dfw=d.resample("M",on="date").mean() print("\nResampling:\n",dfw)

## Output:



1. **Data visualization using Matplotlib (Bar chart,pie,line ,histogram,scatter)**

## Code: Output:

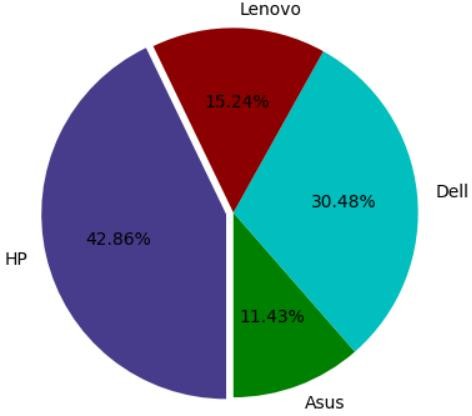
**Bar Chart:**

from matplotlib import pyplot as p

pro\_na=["Intel","AMD","Snapdragon","Tensor"] use=[200,100,250,50]

p.bar(pro\_na,use,color='black',width=0.2) p.xlabel("Processors"),p.ylabel("No of Users") p.title("Processor Users in a Community") p.show()

## Pie chart: Output:

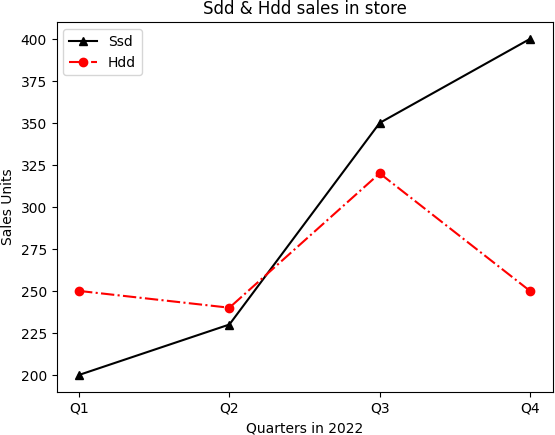
from matplotlib import pyplot as pi us=[12,32,16,45]

la=["Asus","Dell","Lenovo","HP"] e=[0,0,0,0.04] c=["g","c","#8B0000","#473C8B"]

pi.pie(us,labels=la,startangle=270, explode=e,colors=c,autopct='%1.2f%%')

pi.show()

## Line graph: Output:

from matplotlib import pyplot as p Q=["Q1","Q2","Q3","Q4"] ssd=[200,230,350,400] hdd=[250,240,320,250]

p.plot(Q,ssd,'^-',color='black')

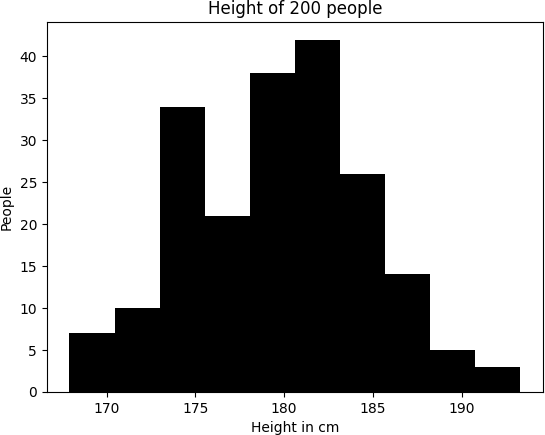
p.plot(Q,hdd,'o-.r')

p.xlabel("Quarters in 2022"),p.ylabel("Sales Units")

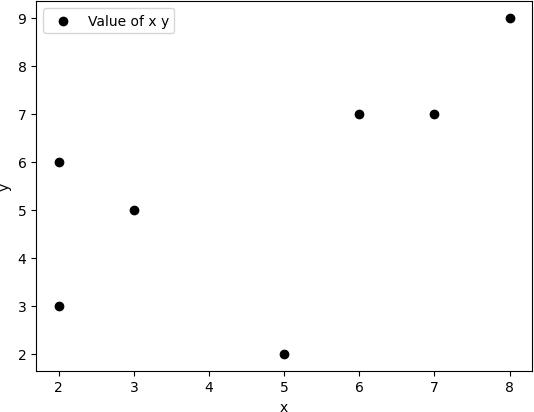
p.title("Sdd & Hdd sales in store") p.legend(['Ssd','Hdd'])

p.show()

## Histogram: Output:

from matplotlib import pyplot as p import numpy as n x=n.random.normal(180,5,200) p.hist(x,color='k') p.xlabel("Height in cm"),p.ylabel("People") p.title("Height of 200 people") p.show()

## Scatter Plot: Output:

from matplotlib import pyplot as p x=[2,6,8,7,3,2,5]

y=[6,7,9,7,5,3,2]

c=['k','b']

p.scatter(x,y,label='Value of x y',color='k')

p.xlabel('x')

p.ylabel('y') p.legend() p.show()

## Visualization of time series data using temperature on different days

**Code:**

import pandas as pd

import matplotlib.pyplot as plt

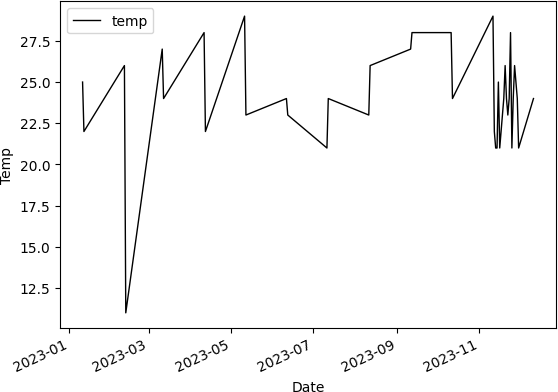
%matplotlib inline

t = pd.read\_csv('C:\\Users\\Desktop\\Data\\te.csv', parse\_dates=['day'], index\_col='day')

a = t.plot(color='k', linewidth=1) plt.xticks(rotation=25) a.set\_ylabel('Temp') plt.xlabel('Date')

plt.show()

## Output:



1. **Visualization of Iris-dataset using Scatter Plot**

## Code:

import pandas as pd

from matplotlib import pyplot as plt

t = pd.read\_csv("C:\\Users\\Desktop\\Data\\i.csv")

species\_colors = {

'Iris-setosa': 'k','Iris-versicolor': 'g','Iris-virginica': 'r'

}

for species, color in species\_colors.items():

sl = t[t['species'] == species]['sepal\_length'] sw = t[t['species'] == species]['sepal\_width'] plt.scatter(sl, sw, color=color, label=species)

plt.legend() plt.xlabel('Sepal Length') plt.ylabel('Sepal Width')

plt.title('Sepal Width and Length for Iris Species') plt.show()

## Output:



1. **Visualization of Iris-dataset using Pie Chart**

## Code:

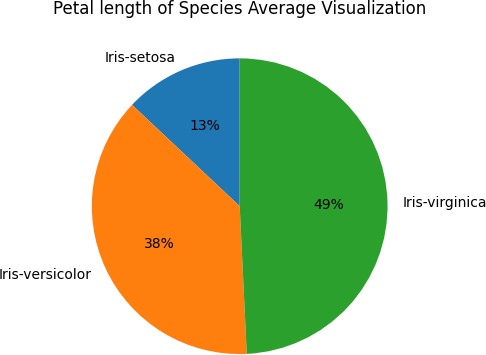
import pandas as pd

from matplotlib import pyplot as plt

t = pd.read\_csv("C:\\Users\\Desktop\\Data\\i.csv") sv=t.groupby("species")["petal\_length"].mean()

plt.pie(sv,labels=sv.index,startangle=90,autopct="%1.0f%%") plt.title("Petal length of Species Average Visualization") plt.show

## Output:



1. **Visualization of Titanic-dataset using Histogram**

## Code:

import matplotlib.pyplot as plt import pandas as pd

data = pd.read\_csv('C:\\Users\\Desktop\\Progs\\Titanic-Dataset.csv') age\_survived = data[data['Survived'] == 1]['Age']

age\_not\_survived = data[data['Survived'] == 0]['Age']

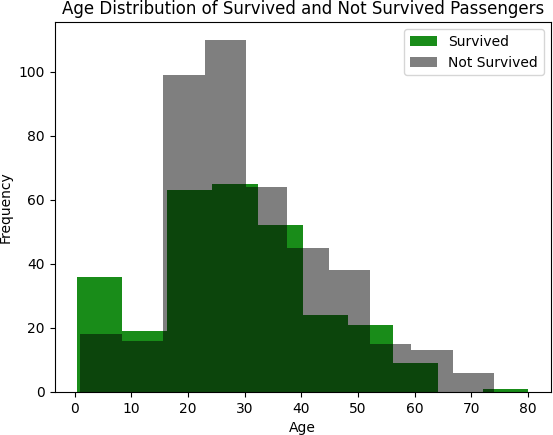
plt.hist(age\_survived, color='g', alpha=0.9, label='Survived') plt.hist(age\_not\_survived, color='k', alpha=0.5,label='Not Survived')

plt.xlabel('Age') plt.ylabel('Frequency')

plt.title('Age Distribution of Survived and Not Survived Passengers') plt.legend()

plt.show()

## Output:



1. **Visualization of Titanic-dataset using bar chart**

## Code:

import pandas as p

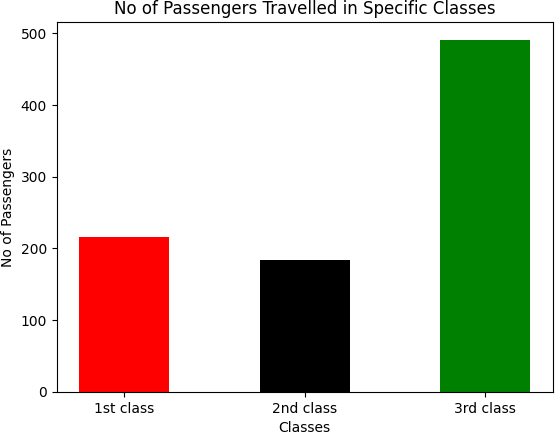
import matplotlib.pyplot as m d=p.read\_csv("C:\\Users\\Desktop\\Progs\\Titanic-Dataset.csv")

c=d["Pclass"].value\_counts() co=['g','r','k'] m.bar(c.index,c.values,color=co,width=0.5)

m.xticks([1,2,3],["1st class","2nd class","3rd class"]) m.xlabel("Classes");m.ylabel("No of Passengers");m.title("No of Passengers Travelled in Specific Classes")

m.show()

## Output:



1. **Visualize Employee dataset using Line graph [Represent Salary and Experience] Code:**

import pandas as p

import matplotlib.pyplot as m d={"Ex":[1,1.3,1.5,2,2.2,2.9,3,3.2,3.2],

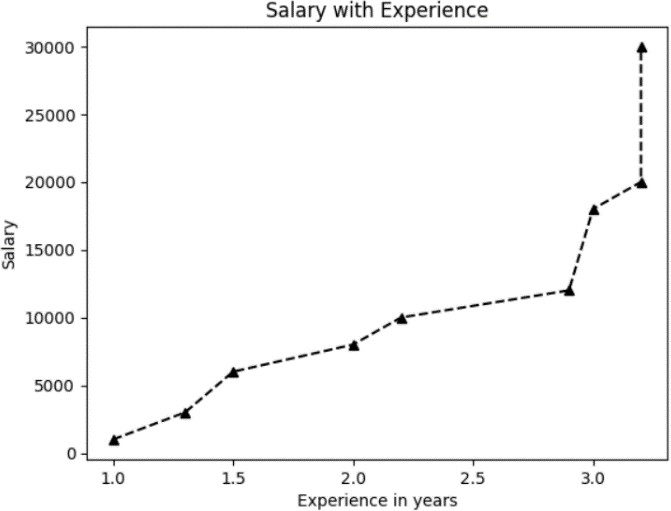
"Salary":[1000,3000,6000,8000,10000,12000,18000,20000,30000]}

df=p.DataFrame(d) m.plot(df["Ex"],df["Salary"],'^--',color='k')

m.xlabel("Experience in years");m.ylabel("Salary");m.title("Salary with Experience")

m.show()

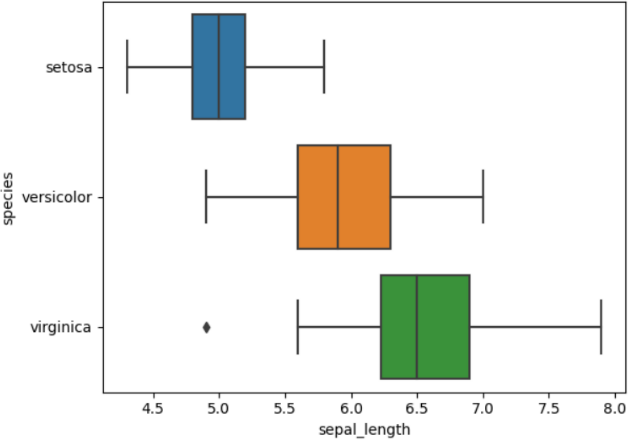
## Output:

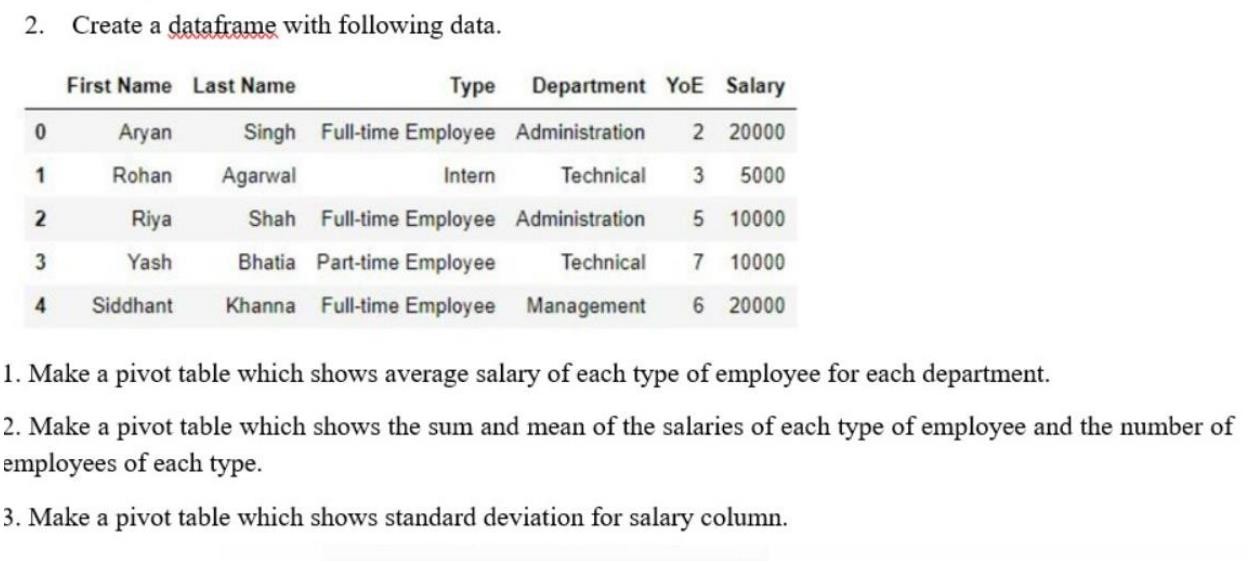


1. **Visualize Iris dataset using Box-Plot Code:**

import seaborn as s

import matplotlib.pyplot as p d=s.load\_dataset('iris') s.boxplot(x=d['sepal\_length'],y=d['species']) p.show()

**Output:**



**Code:**

import pandas as p

import matplotlib.pyplot as m

d={

"First\_name":["Aryan","Rohan","Riya","Yash","Siddhant"],

"Last\_name":["Singh","Agarwal","Shah","Bhatia","Khanna"],

"Type":["Full-Time","Itern","Full-Time","Part-Time","Full-Time"], "Dept":["Administration","Technical","Administration","Technical","Manageme

nt"],

'YoE':[2,3,5,7,6],"Salary":[20000,5000,10000,10000,20000]

}

df=p.DataFrame(d)

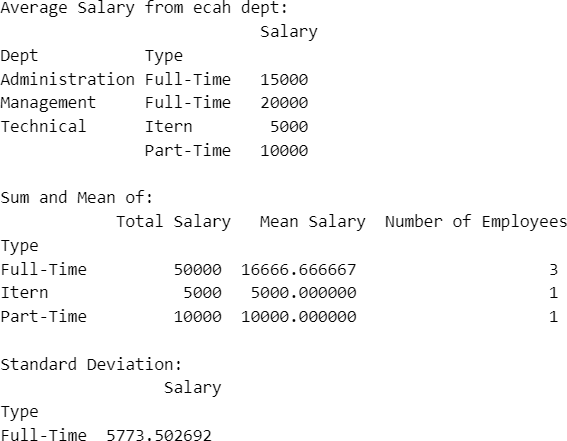
av=df.pivot\_table(index=['Dept', 'Type'], values='Salary', aggfunc='mean') print("Average Salary from ecah dept:\n",av)

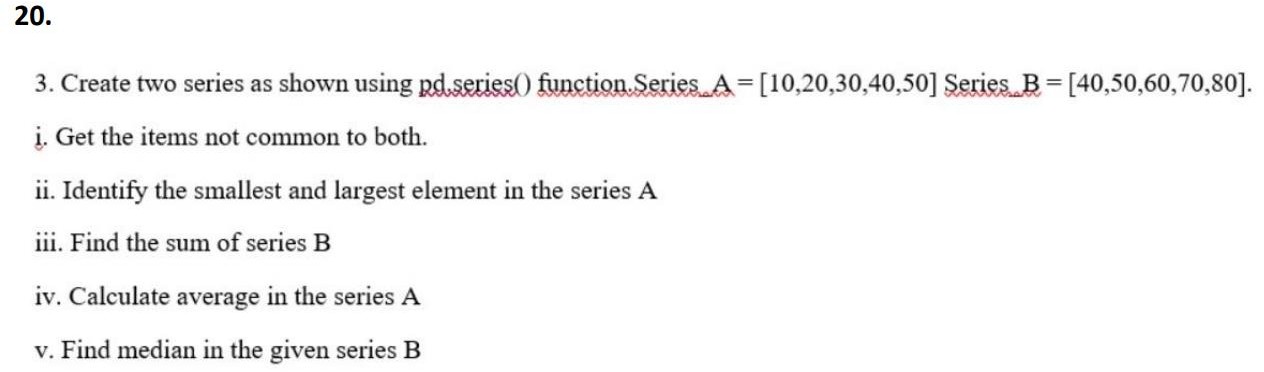
sm=df.pivot\_table(index=['Type'], values='Salary', aggfunc=['sum', 'mean', 'count'])

sm.columns=['Total Salary', 'Mean Salary', 'Number of Employees'] print("\nSum and Mean of:\n",sm)

st=df.pivot\_table(values='Salary', index='Type',aggfunc='std') print("\nStandard Deviation:\n",st)

## Output:



1. 

**Code:**

import pandas as pd

a = pd.Series([10, 20, 30, 40, 50])

b = pd.Series([40, 50, 60, 70, 80])

print("Series A:") print(a) print("\nSeries B:") print(b)

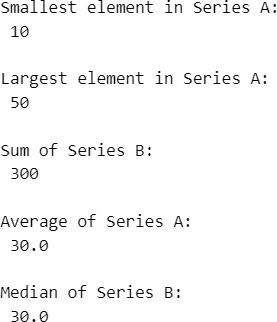
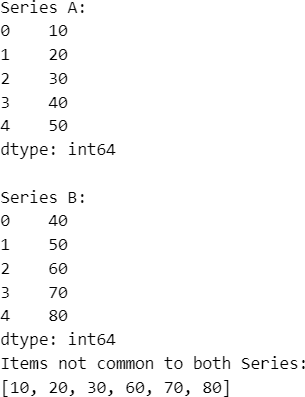
non\_com = a[~a.isin(b)].tolist() + b[~b.isin(a)].tolist() print("Items not common to both Series:")

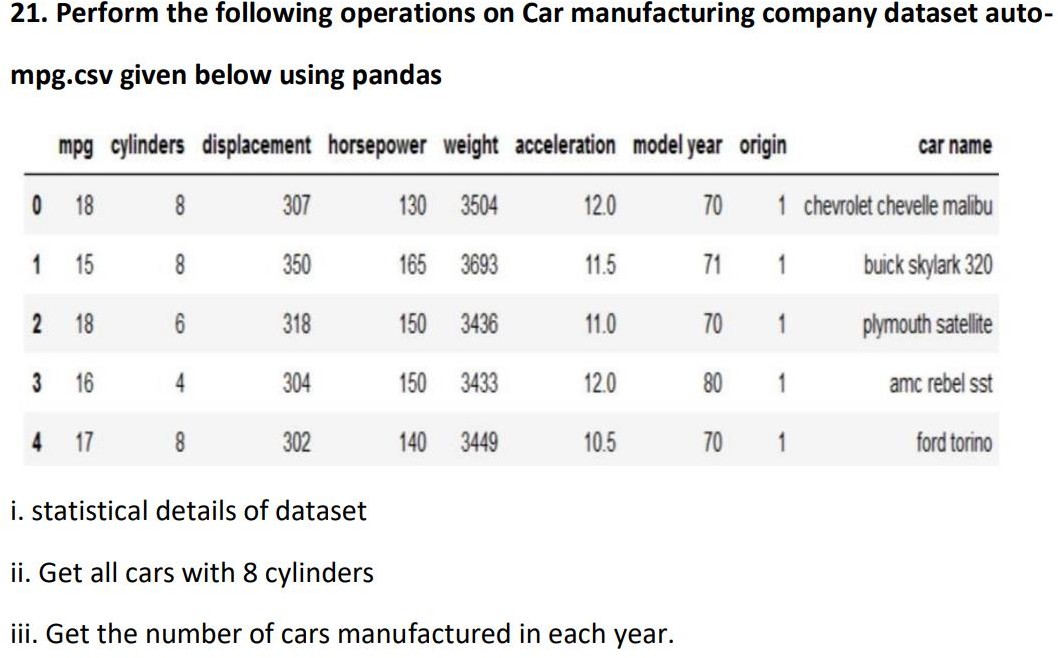
print(non\_com)

print("\nSmallest element in Series A:\n", a.min()) print("\nLargest element in Series A:\n",a.max())

print("\nSum of Series B:\n", b.sum()) print("\nAverage of Series A:\n",a.mean()) print("\nMedian of Series B:\n", a.median())

## Output:





**Code:**

import pandas as pd da={

"mpg":[18,15,18,16,17],"cylinders":[8,8,6,4,8],"displacement":[307,350,318, 304,302],

"horsepower":[130,165,150,150,140],"weigth":[3504,3693,3436,3433,3449], "acceleration":[12.0,11.5,11.0,12.0,10.5],"model year":[70,71,70,80,70],

"origin":[1,1,1,1,1],"car name":["cheverlot","buick","plymoth","amc","ford"]

}

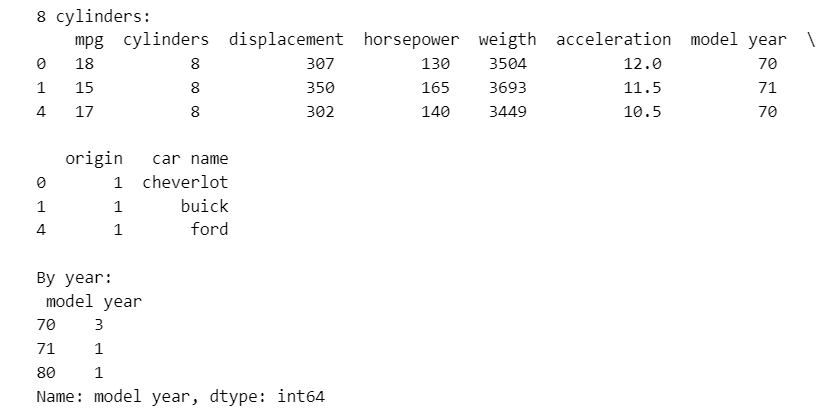
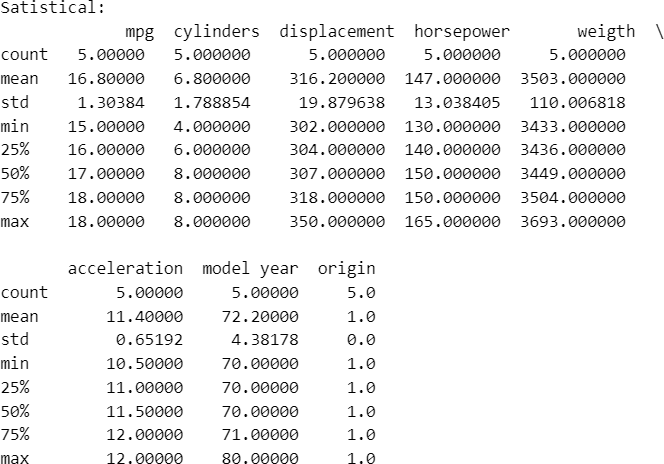
df=pd.DataFrame(da)

sa=df.describe() ei=df[df["cylinders"]==8]

ye = df.groupby('model year')["model year"].count()

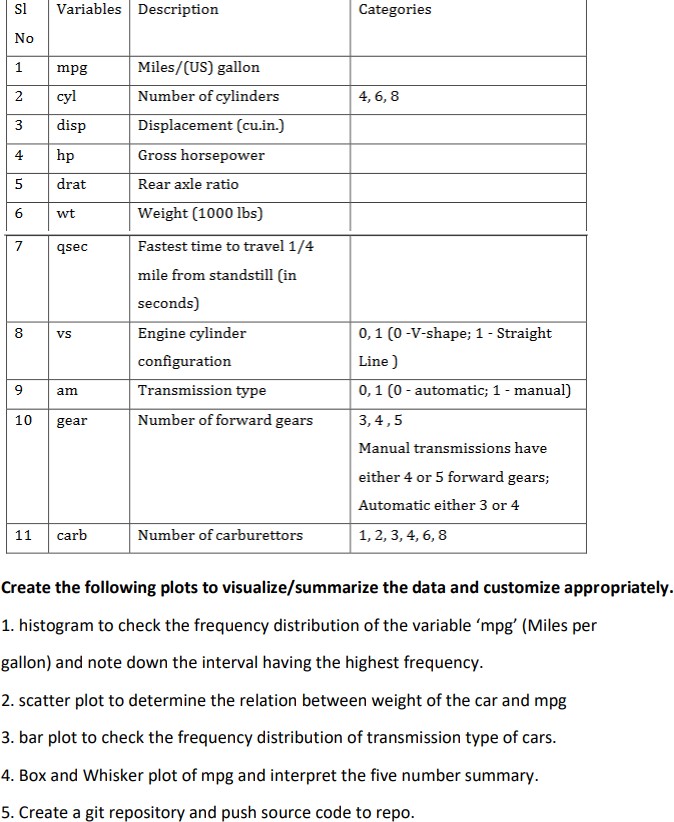
print("Satistical:\n",sa) print("\n8 cylinders:\n",ei) print("\nBy year:\n",ye)

## Output:



1. **Data from an online platform has been collected. This data contains fuel consumption and 11 aspects of automobile design and performance for 32 automobiles. Variable**

## description is given below.Dataset - ‘mtcars.csv



**Code:**

import pandas as p

import matplotlib.pyplot as m import seaborn as s

# data as 32 Elements data=p.read\_csv("C:\\Users\\reddy\\Desktop\\Data\\mtcars.csv")

# HISTOGRAM

mpg=data['mpg'] m.hist(mpg,bins='auto',color='k',edgecolor='c') m.xlabel('Miles per gallon (mpg)');m.ylabel('Frequency') m.title('Frequency Distribution of mpg')

m.show()

# SCATTER

wt=data['wt'] iv=range(len(data))

m.scatter(iv,mpg,color='k',label='mpg') m.scatter(iv,wt,color='g',label='wt') m.title("Relationship b/w Weigth and MPG") m.legend()

m.show()

# BAR PLOT

c=data['am'].value\_counts() co=['k','g']

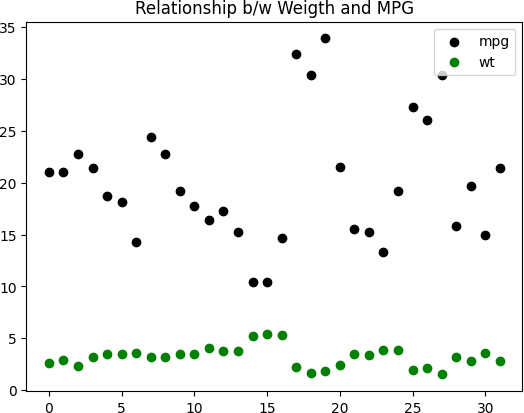
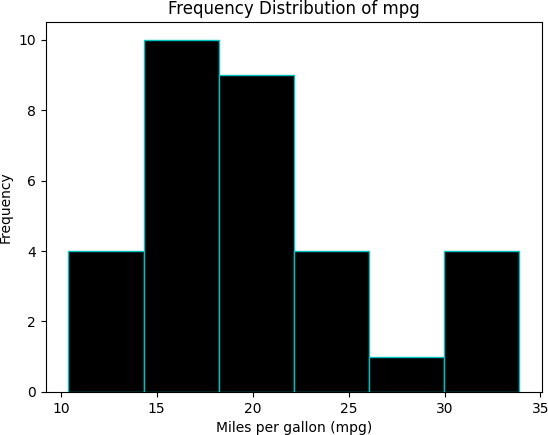
m.bar(c.index,c.values,color=co,width=0.3) m.xticks([0,1],['0-Automatic','1-Manual']) m.xlabel("Tranmisson Type");m.ylabel("No of Cars")

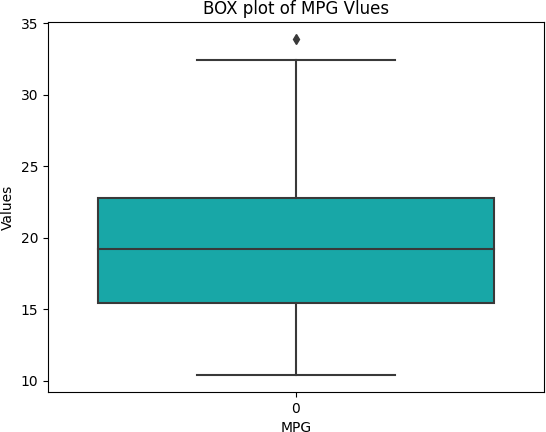
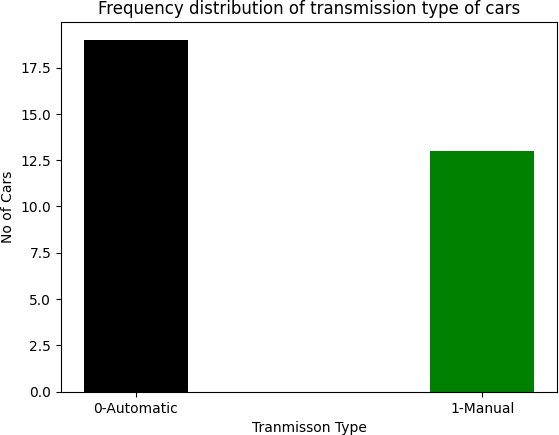
m.title("Frequency distribution of transmission type of cars") m.show()

# BOX PLOT

s.boxplot(mpg,color='c') m.xlabel("MPG");m.ylabel("Values") m.title("BOX plot of MPG Vlues")

## Output:





1. **Ramesh decides to walk 10000 steps every day to combat the effect that lockdown has had on his body’s agility, mobility, flexibility and strength. Consider the following data from fitness tracker over a period of 10 days**

## Code to add 1000 steps to all the observations

1. **Code to find out the days on which Ramesh walked more than 7000 steps**

## Code:

import pandas as p import numpy as n

d={"Day":[1,2,3,4,5,6,7,8,9,10], "Steps":[4335,9552,7332,4504,5335,7552,8332,6504,8965,7689]}

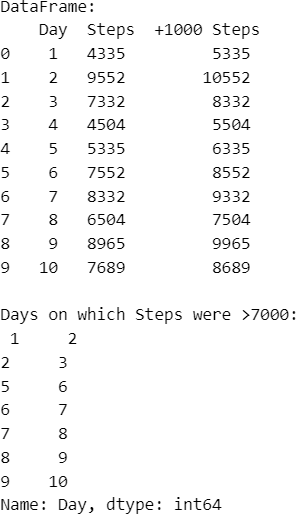
dp=p.DataFrame(d)

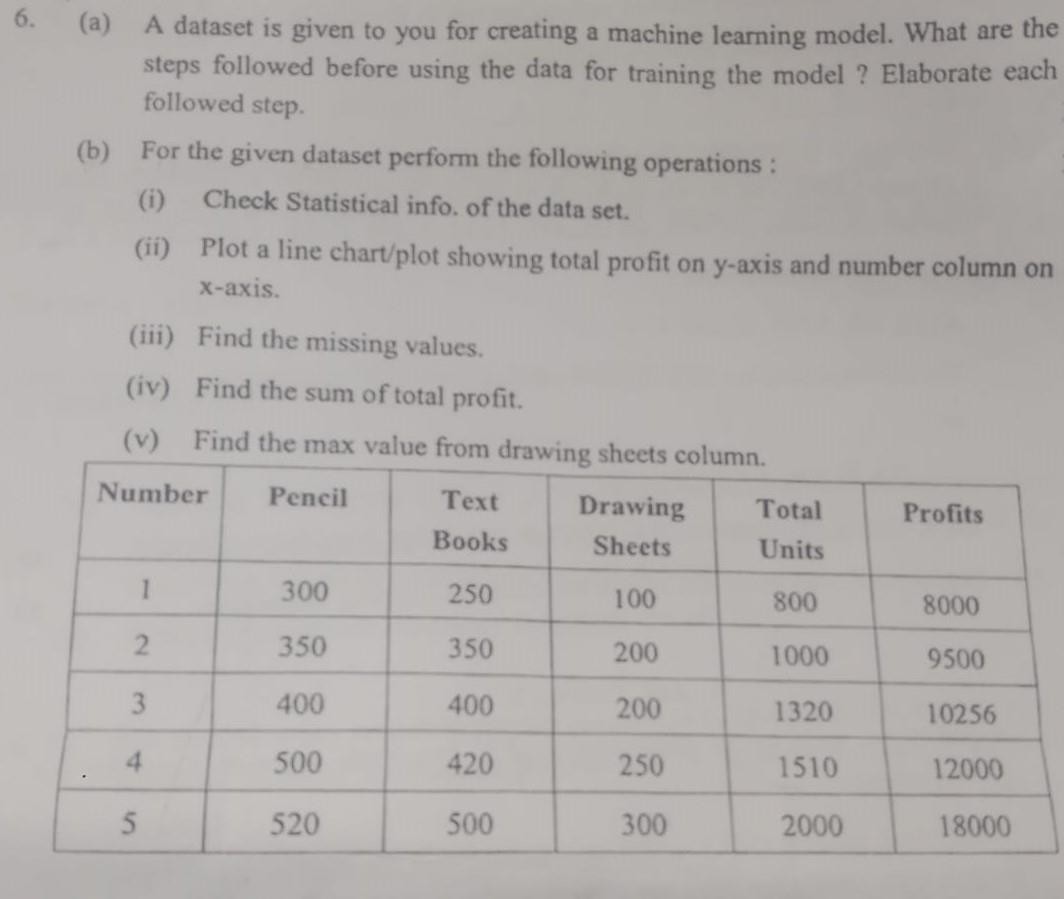
dp["+1000 Steps"]=dp["Steps"]+1000

fi=dp[dp["+1000 Steps"]>7000]["Day"] print("DataFrame:\n",dp)

print("\nDays on which Steps were >7000:\n",fi)

## Output:



**24.**

**Code:**

import numpy as n import pandas as p

import matplotlib.pyplot as m da={

'n':[1,2,3,4,5],'Pencil':[300,350,400,500,520],'TextBooks':[250,350,400,420

,500],

'Draw':[100,200,200,250,300],'Total':[800,1000,1320,1510,2000],"Profits":[8 000,9500,10256,12000,18000]

}

df=p.DataFrame(da)

sta=df.describe() print("Statistics:\n",sta)

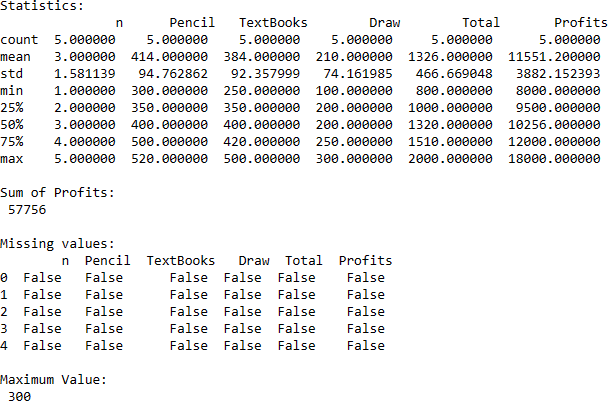
su=df['Profits'].sum() print("\nSum of Profits:\n",su)

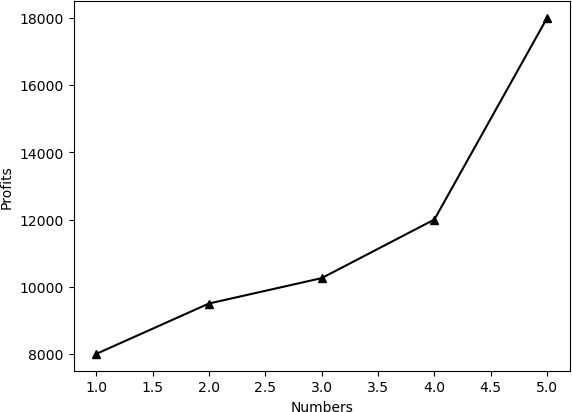
mi=df.isna()

print("\nMissing values:\n",mi) print("\nMaximum Value:\n",df['Draw'].max())

m.plot(df['n'],df['Profits'],'^-',color='k') m.xlabel("Numbers");m.ylabel("Profits") m.show()

## Output:





**Week -04**

## Linear Algebra [Vector , Scalar, Tensors, Matrix, Gradiant, Eigen Values and Vectors] Code:

import numpy as n import tensorflow as t

s=50

print("Scalar : ",s)

m=n.array([[0, 2],[2, 3]])

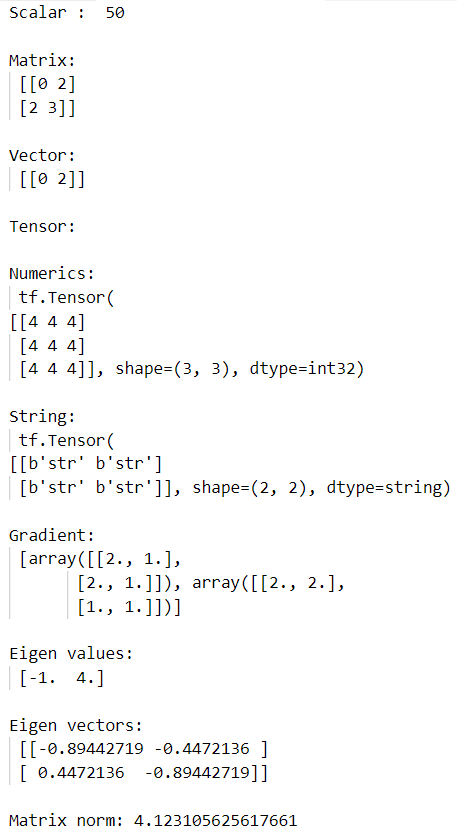
v=n.array([[0, 2]]) print("\nMatrix:\n",m) print("\nVector:\n",v)

print("\nTensor:")

fill\_2d = t.fill([3, 3],4, '2d')

fill\_string = t.fill([2, 2], "str", 'fill\_tensor\_string') print("\nNumerics:\n",fill\_2d) print("\nString:\n",fill\_string)

g=n.gradient(m) print("\nGradient:\n",g)

w,v = n.linalg.eig(m) mat\_norm = n.linalg.norm(m) print("\nEigen values:\n",w) print("\nEigen vectors:\n",v)

print("\nMatrix norm:", mat\_norm)

## Output:

1. **Covariance and Co-relation Code:**

import pandas as pd

from sklearn import datasets iris = datasets.load\_iris()

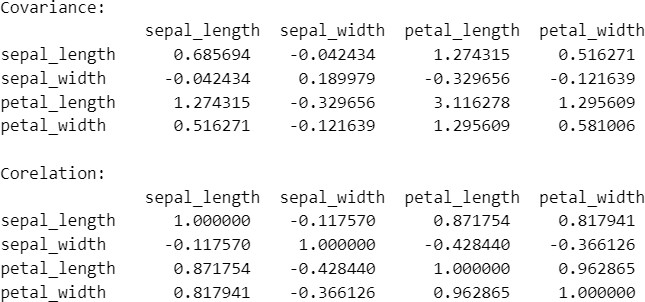
df = pd.DataFrame(iris.data, columns=["sepal\_length", "sepal\_width", "petal\_length", "petal\_width"])

df["class"] = iris.target cov=df.iloc[:, 0:4].cov()

cor=df.iloc[:, 0:4].corr()

print("Covariance:\n",cov) print("\nCorelation:\n",cor)

## Output:



1. **Univariate & Multivariate Distrubution Plot**

## Code:

import seaborn as sns

import matplotlib.pyplot as plt iris = sns.load\_dataset("iris")

#Univariate

sepal\_length\_data = iris["sepal\_length"] plt.figure(figsize=(8, 6)) sns.histplot(sepal\_length\_data, color="skyblue")

plt.xlabel("Sepal Length (cm)") plt.ylabel("Frequency")

plt.title("Univariate Distribution of Sepal Length") plt.show()

#Multivariate

# Create subplots for each numeric variable fig, axes = plt.subplots(2, 2, figsize=(12, 8))

# Plot sepal length distribution by species

sns.boxplot(x="species", y="sepal\_length", data=iris, ax=axes[0, 0]) axes[0, 0].set\_title("Distribution of Sepal Length by Species")

# Plot sepal width distribution by species

sns.boxplot(x="species", y="sepal\_width", data=iris, ax=axes[0, 1]) axes[0, 1].set\_title("Distribution of Sepal Width by Species")

# Plot petal length distribution by species

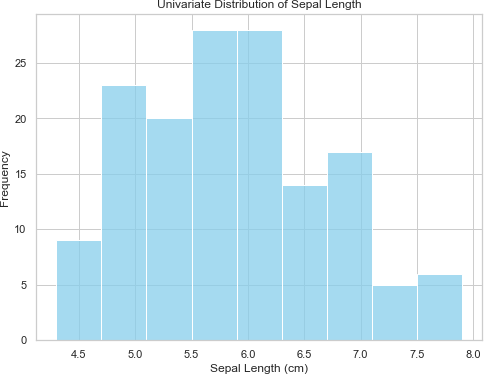
sns.boxplot(x="species", y="petal\_length", data=iris, ax=axes[1, 0]) axes[1, 0].set\_title("Distribution of Petal Length by Species")

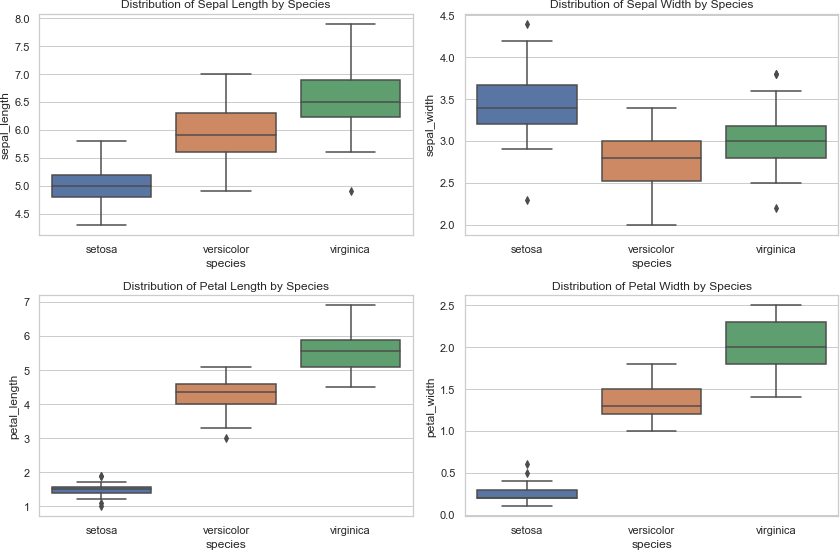
# Plot petal width distribution by species

sns.boxplot(x="species", y="petal\_width", data=iris, ax=axes[1, 1]) axes[1, 1].set\_title("Distribution of Petal Width by Species")

plt.tight\_layout() plt.show()

## Output:





1. **Univariate & Multivariate Comparision Plots Code:**

import seaborn as sns

import matplotlib.pyplot as plt import pandas as p

iris = p.read\_csv("C:\\Users\\reddy\\Desktop\\Data\\i.csv") # Set the figure size

plt.figure(figsize=(18, 10))

# Create grouped bar plots for sepal length, sepal width, petal length, and petal width by species

plt.subplot(2, 2, 1)

sns.barplot(x="species", y="sepal\_length", data=iris, palette="Set3") plt.title("Comparison of Sepal Length by Species")

plt.subplot(2, 2, 2)

sns.barplot(x="species", y="sepal\_width", data=iris, palette="Set3") plt.title("Comparison of Sepal Width by Species")

plt.subplot(2, 2, 3)

sns.barplot(x="species", y="petal\_length", data=iris, palette="Set3") plt.title("Comparison of Petal Length by Species")

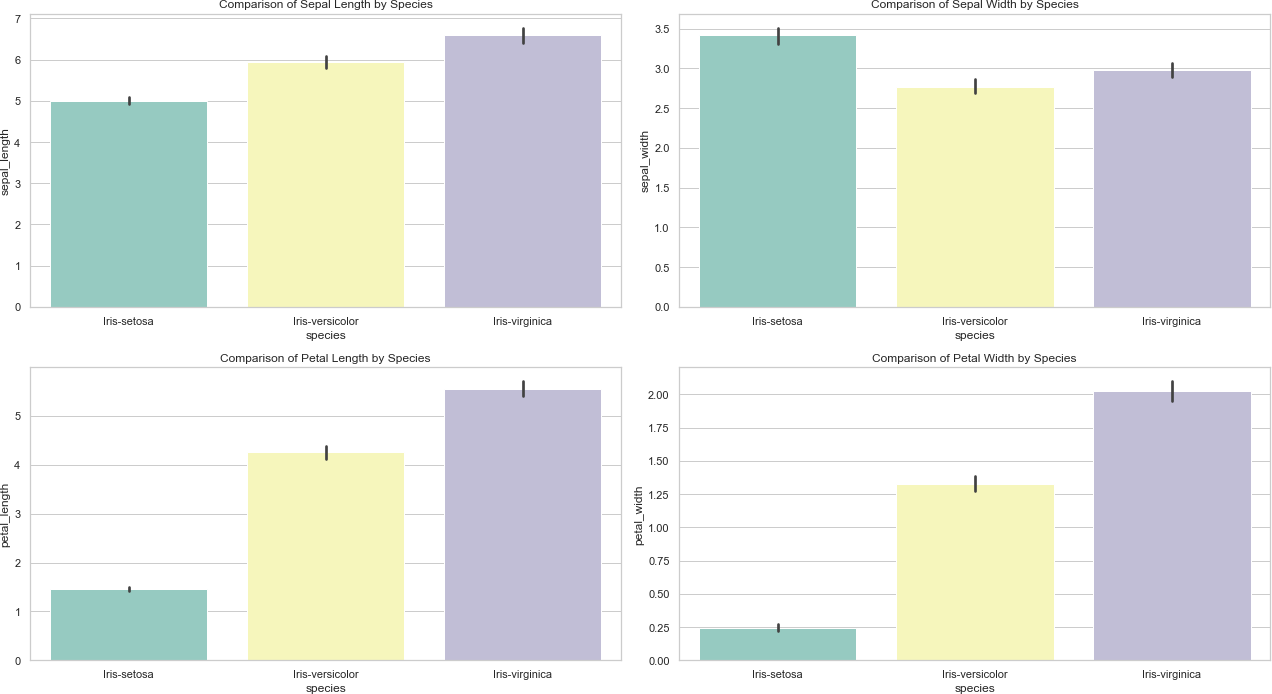
plt.subplot(2, 2, 4)

sns.barplot(x="species", y="petal\_width", data=iris, palette="Set3") plt.title("Comparison of Petal Width by Species")

# Adjust layout plt.tight\_layout()

# Show the plot plt.show()

## Output:



1. **Univariate & Multivariate Composition Plot Code:**

import seaborn as sns

import matplotlib.pyplot as plt # Load the Iris dataset

iris = sns.load\_dataset("iris")

#univariate

sl= iris.groupby("species")["sepal\_length"].mean() plt.pie(sl, labels=sl.index, autopct='%1.1f%%', colors=sns.color\_palette("Set3"))

plt.title("Composition of Mean Sepal Length by Species") plt.show()

#Multivariate

#Reducing the dataset to count of 40 iris = sns.load\_dataset("iris").head(40)

# Group data by species and calculate the mean of numeric variables species\_data = iris.groupby("species")[["sepal\_length", "sepal\_width", "petal\_length", "petal\_width"]]

# Create a stacked bar chart

species\_data.plot(kind="bar", stacked=True, colormap="Set3", figsize=(18, 10))

# Set labels and title

plt.title("Multivariate Composition of Iris Species") plt.xlabel("Species")

plt.ylabel("Mean Value")

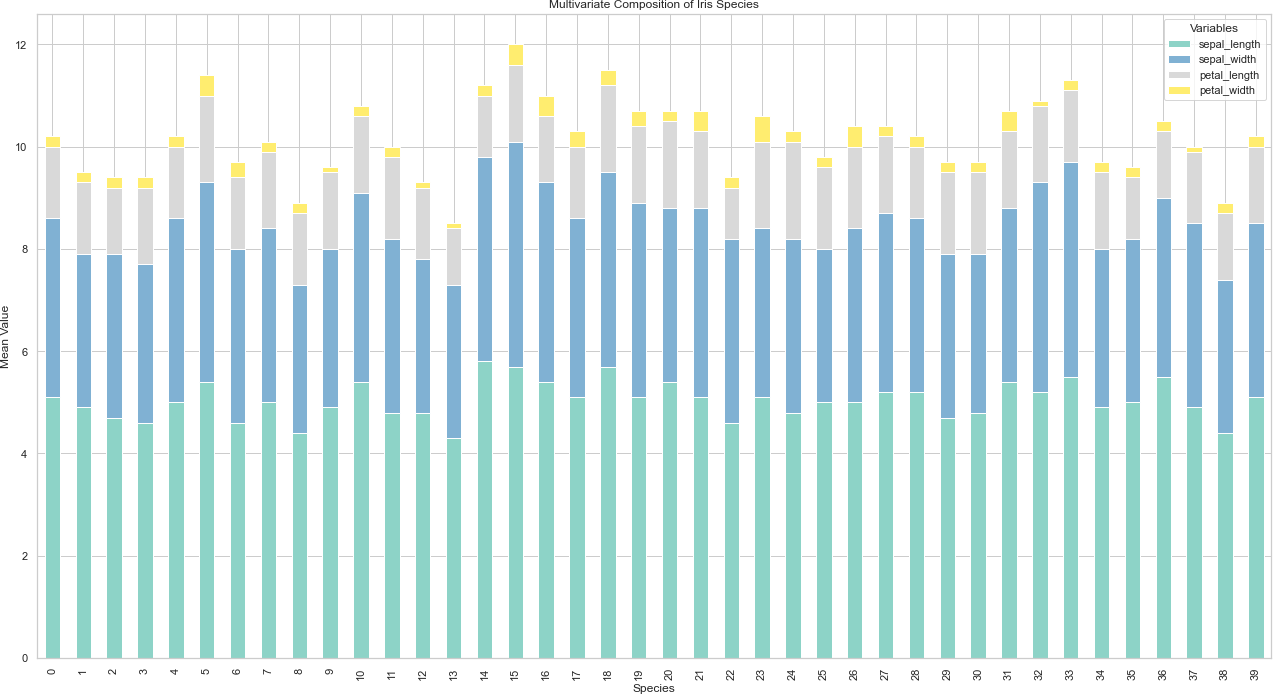
# Show the plot

plt.legend(title="Variables", loc="upper right") plt.tight\_layout()

plt.show()

## Output:





1. **Multivariate Relationship Plot**

## Code:

import seaborn as sns import pandas as pd

import matplotlib.pyplot as plt df = sns.load\_dataset('iris')

# plt.figure(figsize=(24,20)) sns.pairplot(df, hue="species") plt.suptitle('Multivariate plot',y=1.02) plt.show()

## Output:

**Week -05**

## Detect missing values with pandas dataframe. functions: .info() and .isna()

**Code:**

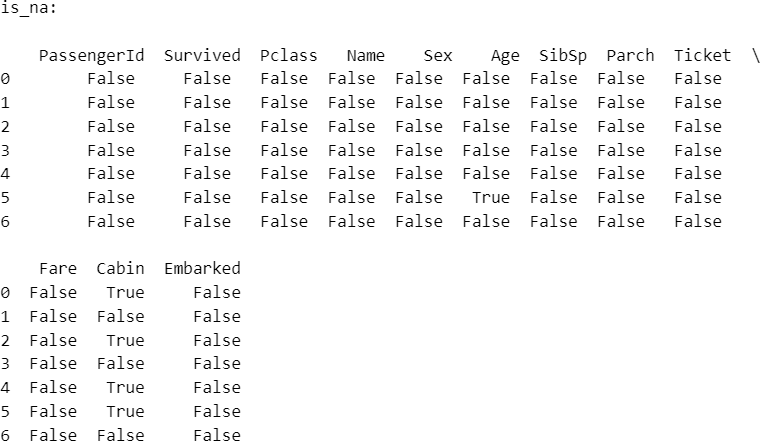
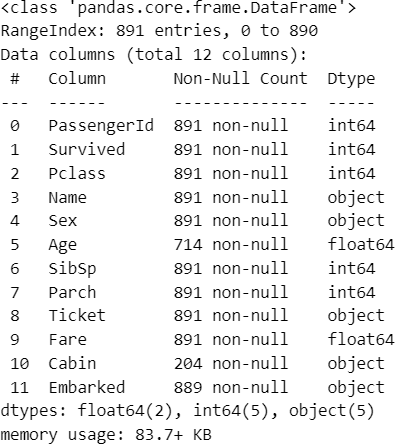
import pandas as p df=p.read\_csv("C:\\Users\\Desktop\\Data\\Titanic-Dataset.csv")

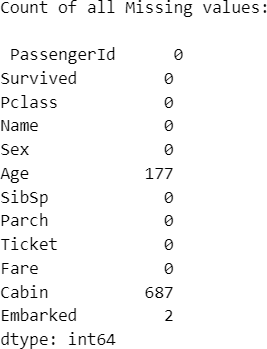
info=df.info()

print("\n\n\nis\_na:\n\n",df.isna().head(7)) is\_null\_su=df.isna().sum()

print("\n\n\nCount of all Missing values:\n\n",df.isna().sum())

## Output:





1. **Replace**

## Code:

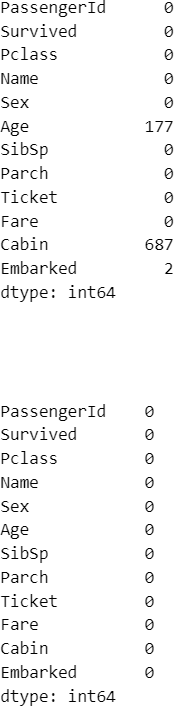
import numpy as n df=p.read\_csv("C:\\Users\\Desktop\\Data\\Titanic-Dataset.csv")

print(df.isna().sum())

#Replacing all NaN values with -1 df=df.replace({n.nan:-1})

print("\n\n\n") print(df.isna().sum())

## Output:



1. **Remove data objects with missing values**

## Code:

df=p.read\_csv("C:\\Users\\Desktop\\Data\\Titanic-Dataset.csv") print("\nBefore Droping:\n")

df.info()

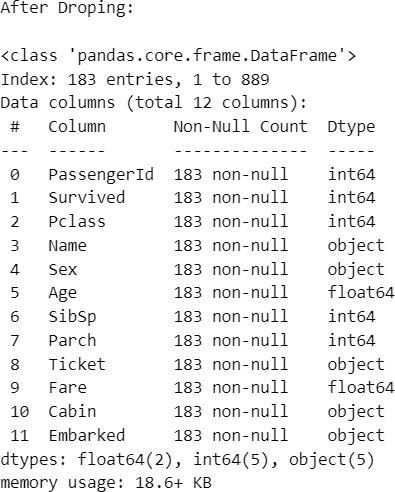
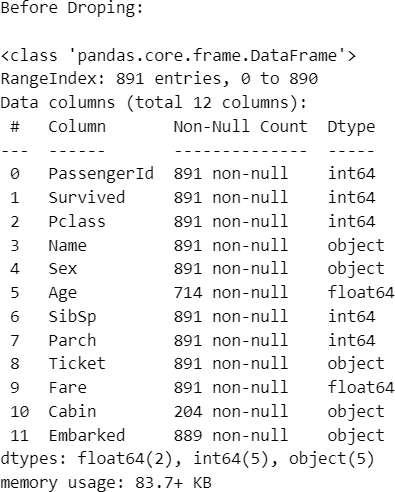
#drops Entier row data if as nan values in any coloumn df=df.dropna()

#OR

#dp=dp.dropna(axis=0)

print('\n\nAfter Droping:\n') df.info()

## Output:



1. **Remove the attributes with missing values**

## Code:

df=p.read\_csv("C:\\Users\\Desktop\\Data\\Titanic-Dataset.csv")

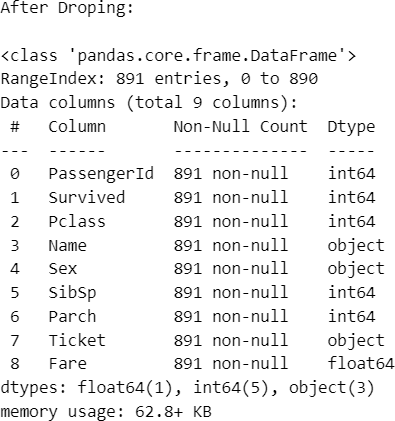
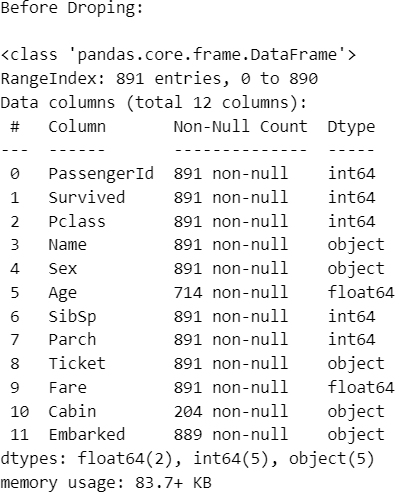
print("\nBefore Droping:\n") df.info()

df=df.dropna(axis=1) #OR

#df=df.drop(columns=df.columns[df.isnull().any()])

print('\n\nAfter Droping:\n') df.info()

## Output:



1. **Estimate and impute missing values**

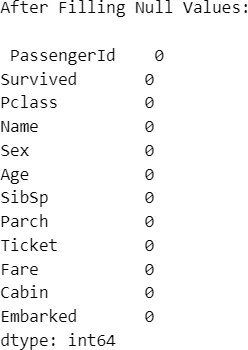
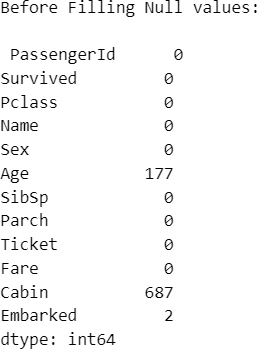
## Filling it with some Arbitrary value here it is 0

**Code:**

df=p.read\_csv("C:\\Users\\Desktop\\Data\\Titanic-Dataset.csv") print("Before Filling Null values:\n\n",df.isna().sum()) df=df.fillna(0)

print("\n\nAfter Filling Null Values:\n\n",df.isna().sum())

## Output:



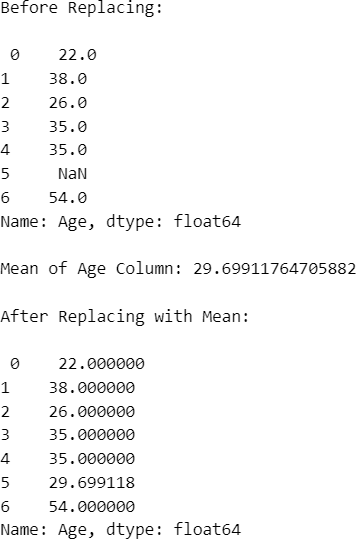
1. **Replacing with Mean Value**

## Code:

df=p.read\_csv("C:\\Users\\Desktop\\Data\\Titanic-Dataset.csv") print("Before Replacing:\n\n",df['Age'].head(7)) print("\nMean of Age Column:",df['Age'].mean())

dp=df['Age'].fillna(df['Age'].mean()) print("\nAfter Replacing with Mean:\n\n",dp.head(7))

## Output:

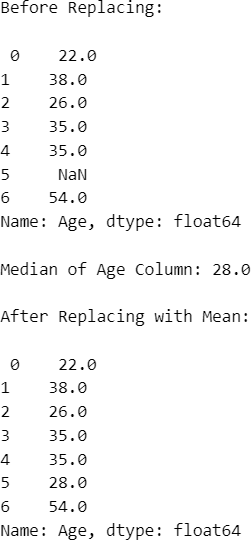


1. **Replacing with Median Value Code:**

df=p.read\_csv("C:\\Users\\Desktop\\Data\\Titanic-Dataset.csv") print("Before Replacing:\n\n",df['Age'].head(7)) print("\nMedian of Age Column:",df['Age'].median())

dp=df['Age'].fillna(df['Age'].median()) print("\nAfter Replacing with Mean:\n\n",dp.head(7))

## Output:



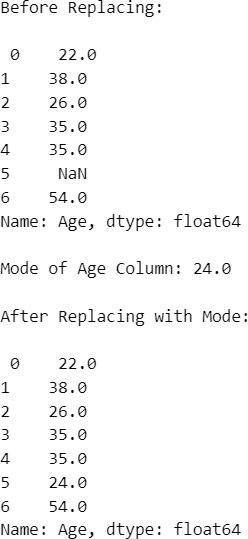
1. **Replacing with Mode value**

## Code:

df=p.read\_csv("C:\\Users\\Desktop\\Data\\Titanic-Dataset.csv") print("Before Replacing:\n\n",df['Age'].head(7)) print("\nMode of Age Column:",df['Age'].mode()[0])

dp=df['Age'].fillna(df['Age'].mode()[0]) print("\nAfter Replacing with Mode:\n\n",dp.head(7))

## Output:



1. **Univariate Outliers Code:**

from sklearn.datasets import load\_diabetes import matplotlib.pyplot as m

import seaborn as s

dp=load\_diabetes() col\_n =dp.feature\_names

df= p.DataFrame(dp.data);df.columns = col\_n

#Visualizing of Outliers s.boxplot(df['bmi'])

m.ylabel('Values');m.xlabel('bmi');m.title('Distrubution of bmi') m.show()

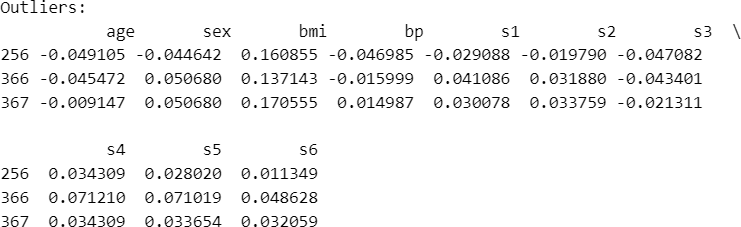
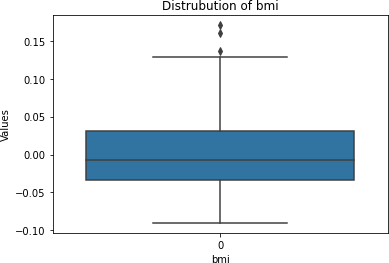
#IQR

q1=df['bmi'].quantile(0.25) q3=df['bmi'].quantile(0.75) iqr=q3-q1

#Floor and Capping flo=q1-1.5\*iqr cap=q3+1.5\*iqr

out=df[(df.bmi<=flo)|(df.bmi>=cap)] print("Outliers:\n",out)

## Output:



1. **Multivariate Outliers Code:**

from sklearn.datasets import load\_diabetes import matplotlib.pyplot as m

import seaborn as s

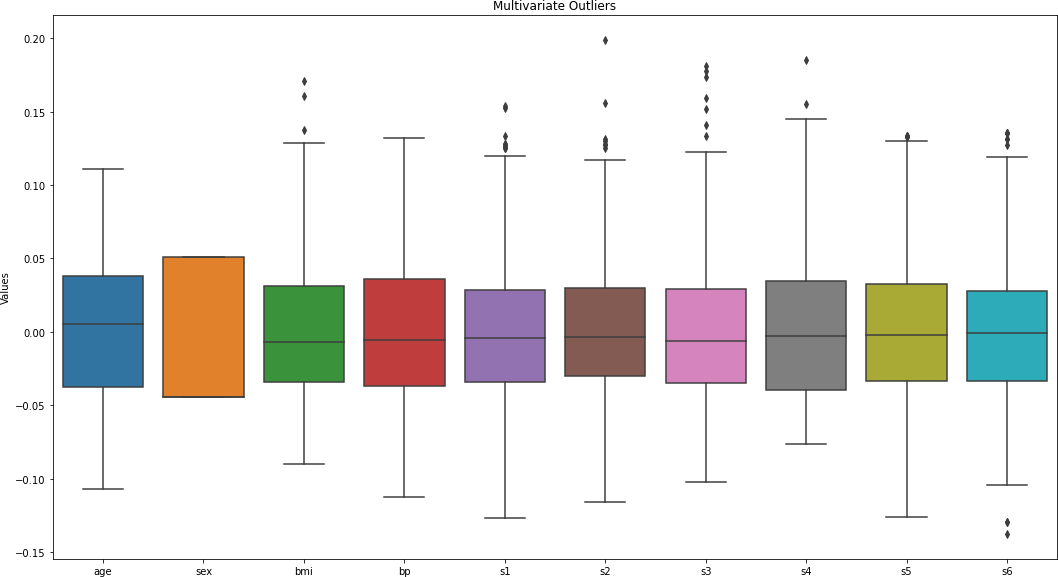
dp=load\_diabetes() col\_n =dp.feature\_names

df= p.DataFrame(dp.data) df.columns = col\_n

m.figure(figsize=(18,10)) s.boxplot(data=df) m.title('Multivariate Outliers') m.ylabel('Values')

m.show()

## Output:



1. **Time series outlier detection**

## Code:

import numpy as n import pandas as p

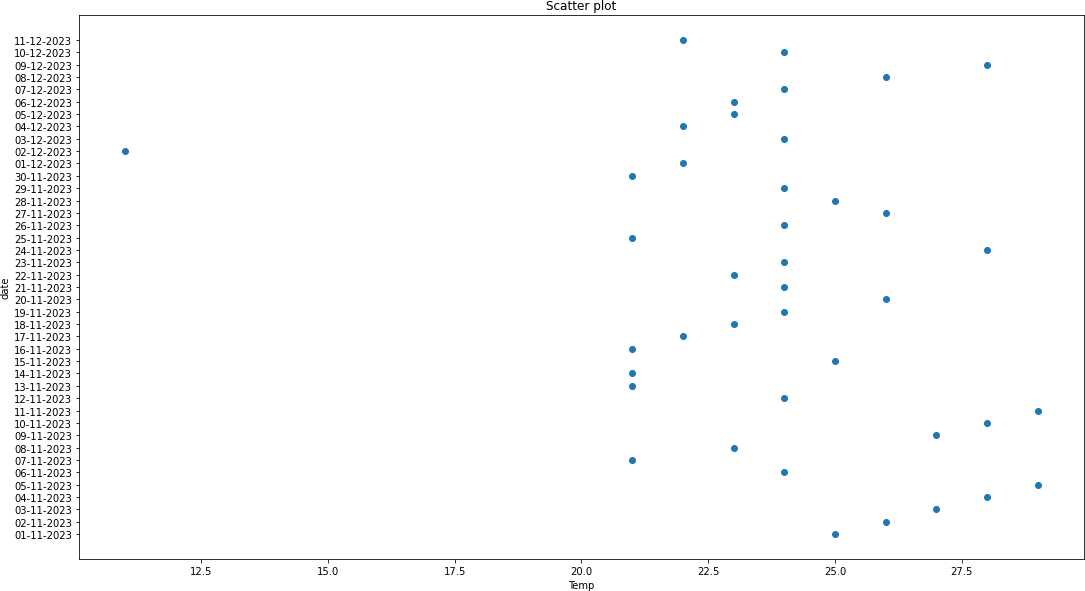
import matplotlib.pyplot as m df=p.read\_csv("C:\\Users\\reddy\\Desktop\\Data\\te.csv") x=df.temp

y=df.day m.figure(figsize=(18,10))

m.scatter(x,y,label="values of x & y") m.xlabel("Temp")

m.ylabel("date") m.title("Scatter plot") m.show()

## Output:



1. **Titanic Dataset Perform:**

## Visualize missing values as bar plot and matrix plot

* + **Handle Missing values by deleting data objects and attributes**

## Impute the missing values

**Code:**

import missingno as ms ti\_da=p.read\_csv("C:\\Users\\Desktop\\Data\\Titanic-Dataset.csv")

#Box Plot ms.bar(ti\_da)

m.title("Missing values in Dataset") m.show()

#Matrix Plot ms.matrix(ti\_da)

m.title('Missing Values Matrix Plot') m.show()

#Removing Null Objects print("Before Droping Objects:\n") ti\_da.info() ti\_d=ti\_da.dropna(axis=0)

print("\n\nAfter Droping objects:\n") ti\_d.info()

#Removing Null Attributes print("\nBefore Droping Attributes:\n") ti\_da.info()

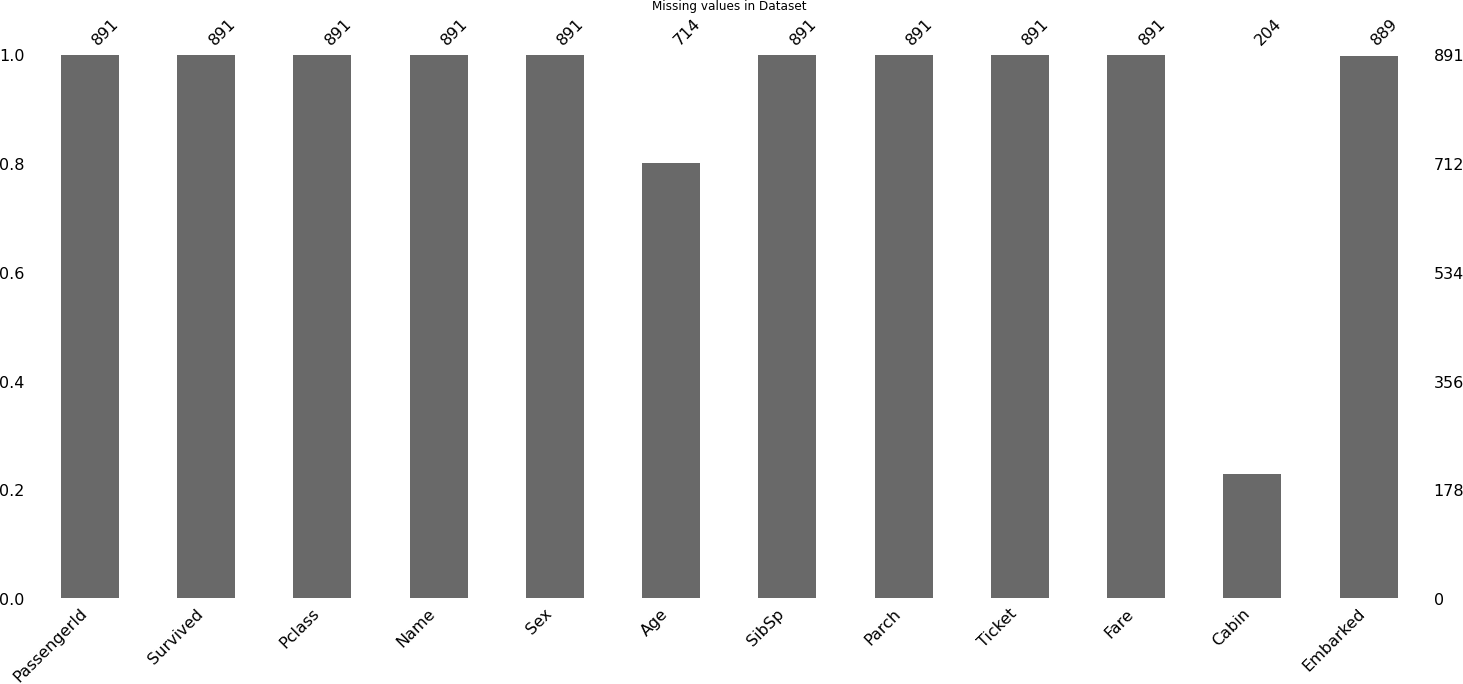
ti=ti\_da.dropna(axis=1)

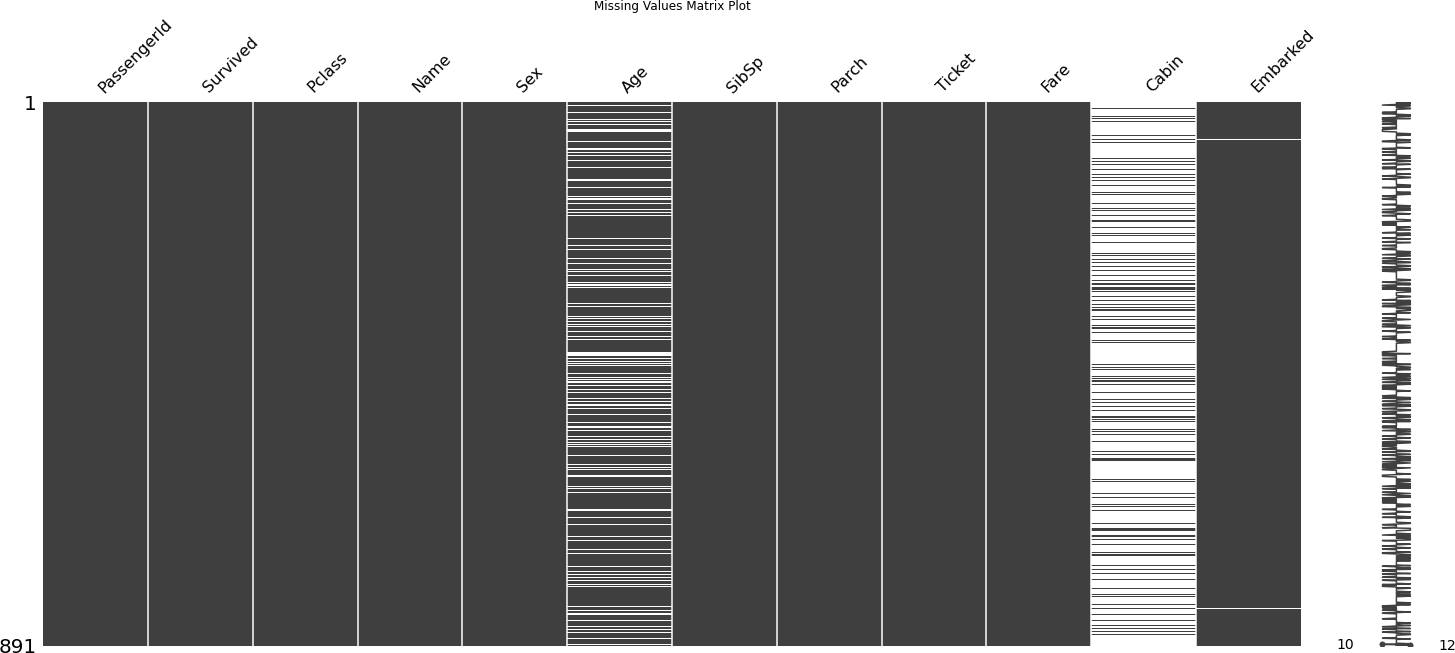
print("\n\nAfter Droping Attributes:\n") ti.info()

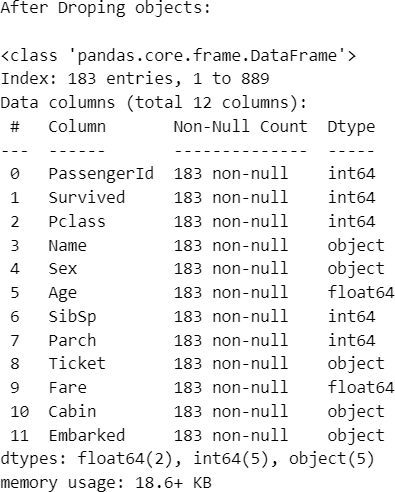
#Imputing Missing value of Age column through Mean print("\n\nAge column before imputing:\n") ti\_da['Age'].info() ti\_ag=ti\_da['Age'].fillna(ti\_da['Age'].mean()) print("\n\nAfter Imputing:\n")

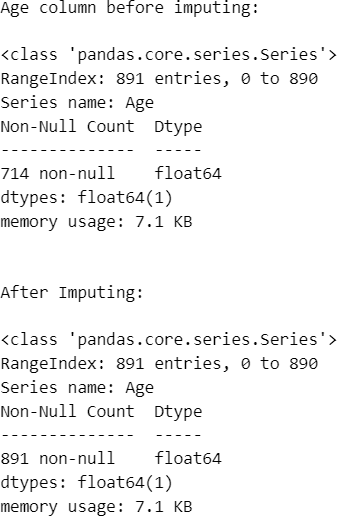
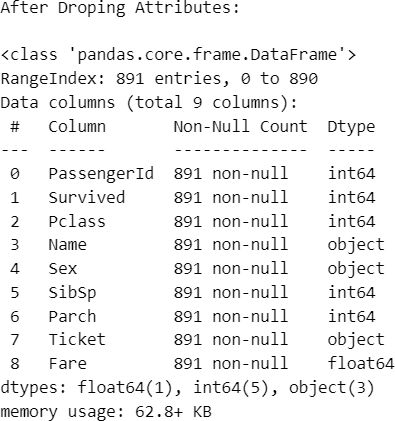
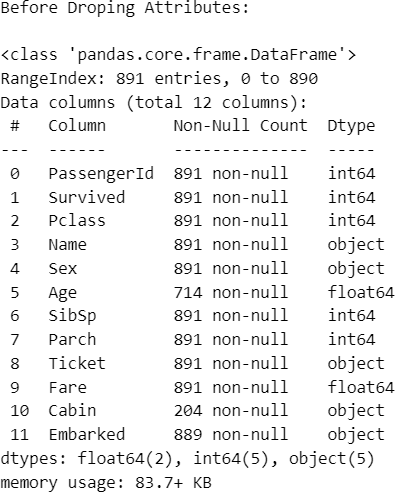
ti\_ag.info()

## Output:







## For Credit dataset

* + **Spot outliers in income using bivariate plot**

## Spot outliers in any feature using boxplot

* + **Detect outliers in any one feature using IQR method**

## Treat outliers using Imputation [Mean, Median, Zero]

**Code:**

import pandas as p

import matplotlib.pyplot as m import seaborn as s

da=p.read\_csv("C:\\Users\\Desktop\\Data\\credit\_risk\_dataset.csv").head(5 0)

#Bivariate Plot m.figure(figsize=(18,10))

s.scatterplot(x=da['person\_age'],y=da['person\_income'],data=da) m.title("Bivariate Plot")

m.show()

#Box Plot m.figure(figsize=(18,10)) s.boxplot(da['person\_income'])

m.xlabel('Income');m.ylabel('Values') m.title("Box Plot of Income column") m.show()

#Detect Outliers using IQR Method inc=da['person\_income'] q1=inc.quantile(0.25) q3=inc.quantile(0.75)

iqr=q3-q1 low=q1-1.5\*iqr hig=q3+1.5\*iqr

out=(inc <= low) | (inc>= hig) print("Outliers:\n",out.sum())

#Impute Outliers using Mean

mean\_in=inc[(inc >= low) & (inc <= hig)].mean() da.loc[out, 'person\_income'] = mean\_in

m.figure(figsize=(18,10)) m.boxplot(da['person\_income']) m.title("After Imputing with mean") m.show()

# #Impute with Median da=p.read\_csv("C:\\Users\\Desktop\\Data\\credit\_risk\_dataset.csv").head(5 0)

inc=da['person\_income']

out1= (inc <= low) | (inc >= hig) print("Outliers:\n",out1.sum())

medi=inc[(inc>= low) & (inc<= hig)].median() da.loc[out1, 'person\_income'] = medi

m.figure(figsize=(18,10)) m.boxplot(da['person\_income']) m.title("After Imputing with median") m.show()

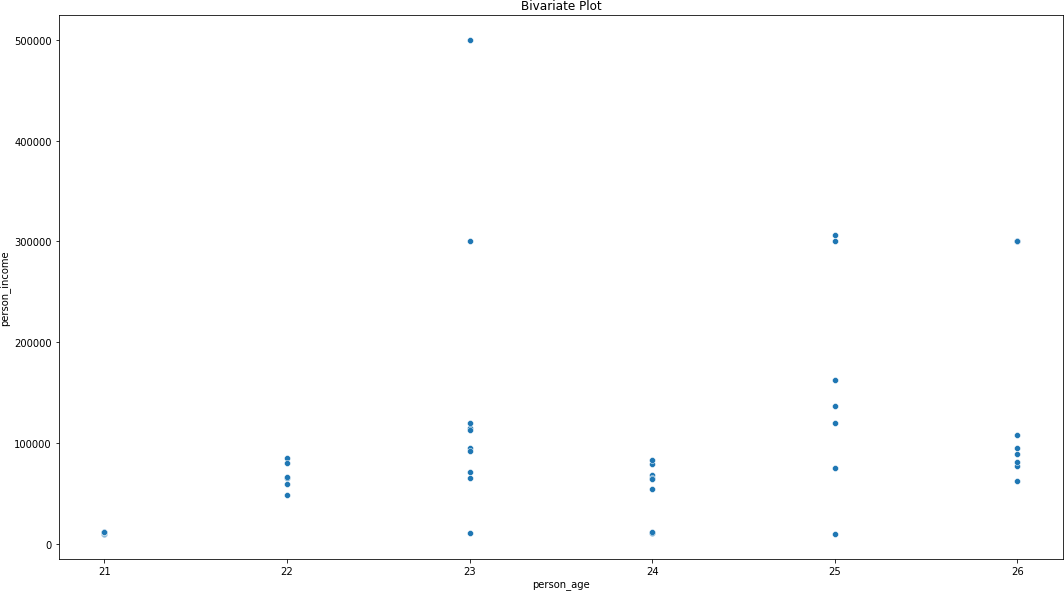
#Impute with Zero da=p.read\_csv("C:\\Users\\Desktop\\Data\\credit\_risk\_dataset.csv").head(5 0)

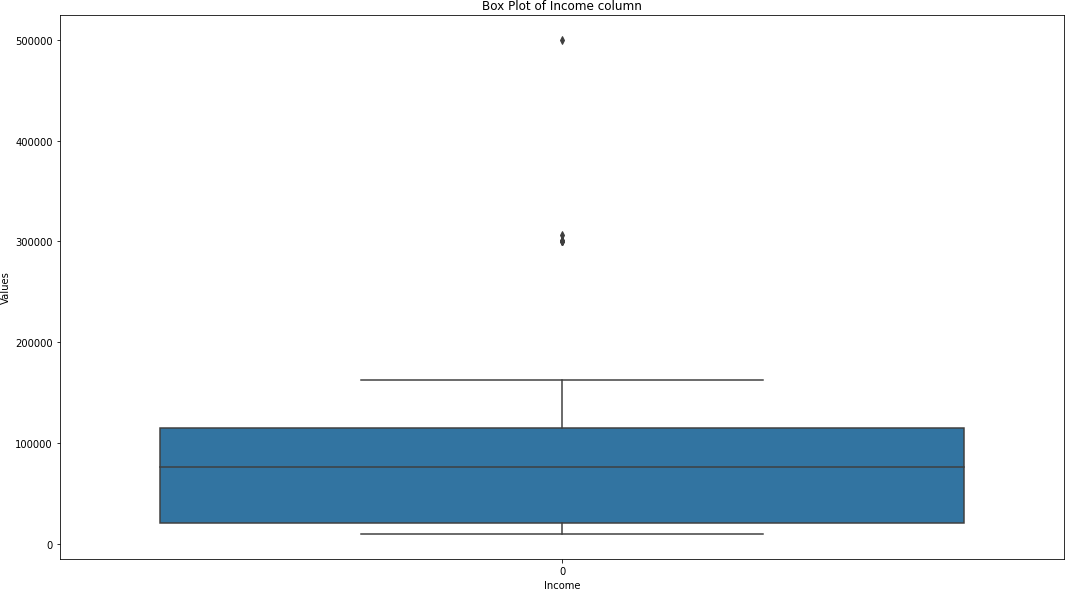
inc=da['person\_income'] out2=(inc <= low) | (inc>= hig) print("Outliers:\n",out2.sum())

da.loc[out2, 'person\_income'] = 0

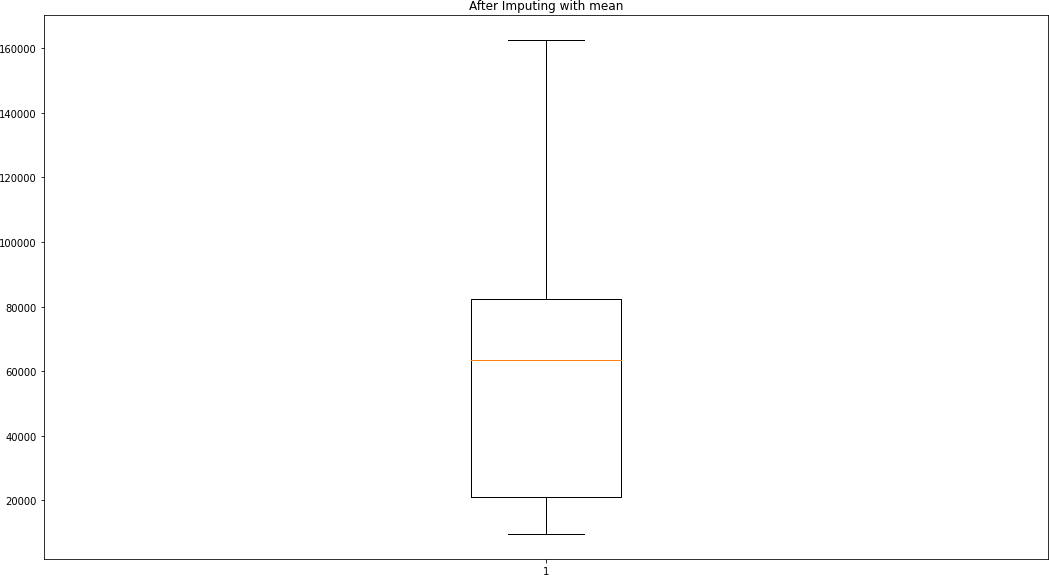
m.figure(figsize=(18,10)) m.boxplot(da['person\_income']) m.title("After Imputing with Zero [0]") m.show()

## Output:

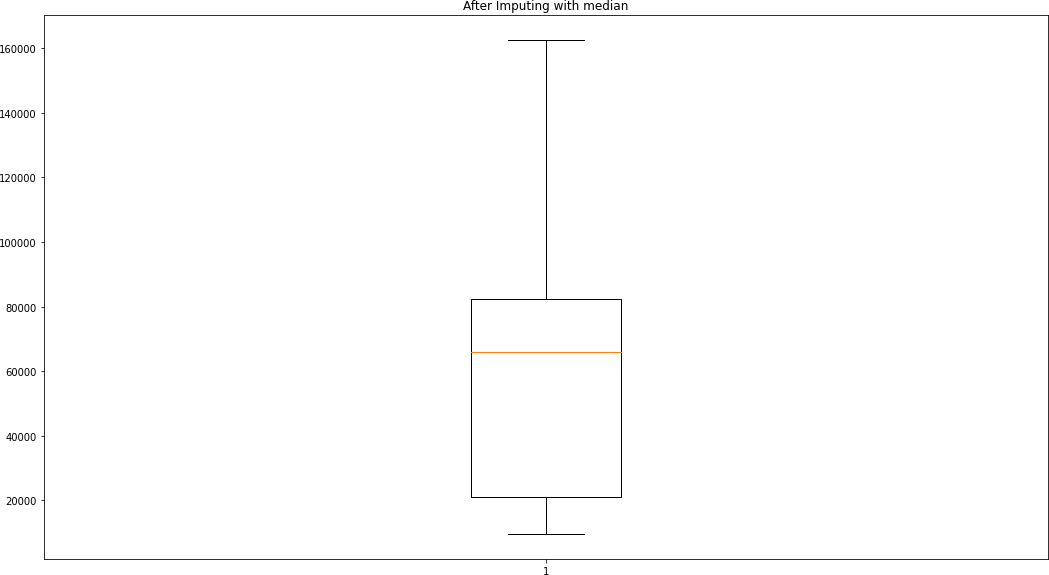




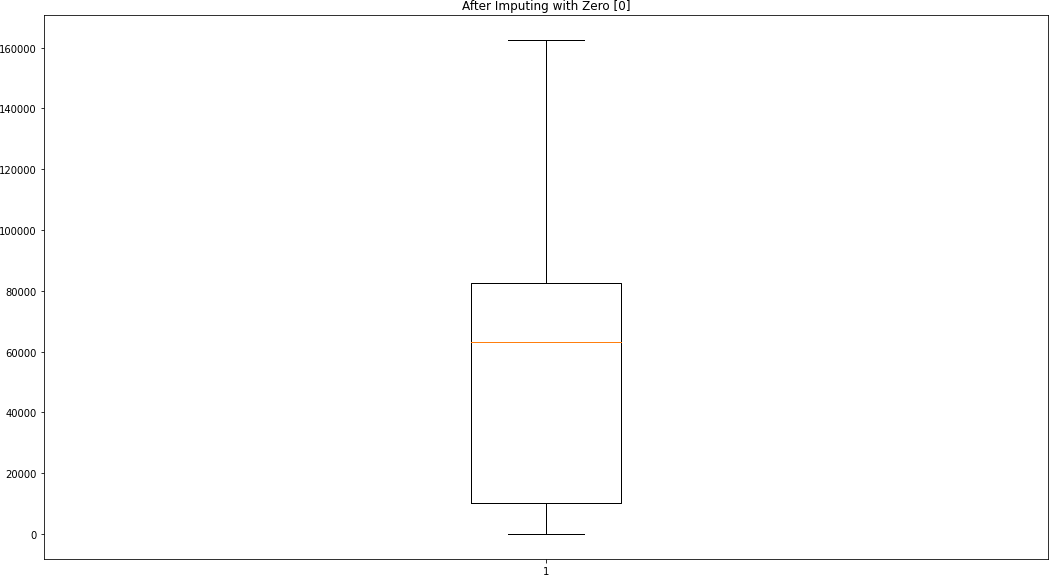
Outliers: 8



Outliers: 8



Outliers: 8



**Week-06**

1. **Salary Prediction according to Experience. Code:**

import pandas as p

import matplotlib.pyplot as m

from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

da=p.read\_csv("C:\\Users\\Desktop\\Data\\Salary\_LR.csv") da.info()

da.dropna(axis=0,inplace=True)

x=da['YearsExperience'].values.reshape(-1,1) y=da['Salary']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.3, random\_state=42)

model = LinearRegression() model.fit(x\_train, y\_train) y\_pred = model.predict(x\_test)

m.figure(figsize=(10, 6))

m.scatter(x\_test, y\_test, color='blue', label='Actual Data') m.plot(x\_test, y\_pred, color='red', linewidth=2, label='Regression Line') m.title('Linear Regression: Salary vs. Years of Experience') m.xlabel('Years of Experience');m.ylabel('Salary')

m.legend();m.show()

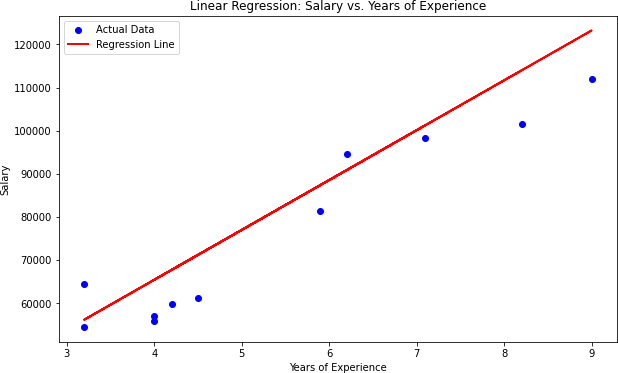
mse = mean\_squared\_error(y\_test, y\_pred) r2 = r2\_score(y\_test, y\_pred)

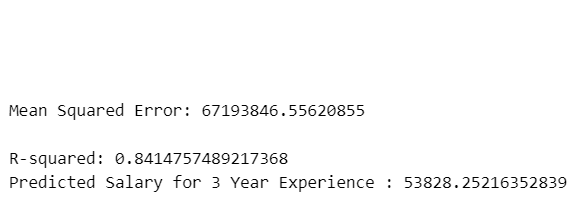
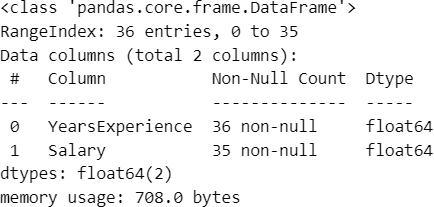
print("\nMean Squared Error:", mse) print("\nR-squared:", r2)

Expe= [[3]] # Replace with the desired experience value Sal = model.predict(Expe)

print(f"Predicted Salary for {Expe[0][0]} Year Experience : {Sal[0]}")

## Output:





1. **Marks Prediction according to Study Hours.**

## Code:

import pandas as p

import matplotlib.pyplot as m import seaborn as s

from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

da=p.read\_csv("C:\\Users\\Desktop\\Data\\Stud\_marks.csv") print(da.isna().sum())

s.boxplot(da['Marks']);m.title('Marks') m.show()

s.boxplot(da['time\_study']);m.title('Study Hours') m.show()

s.heatmap(data=da.corr(), annot=True, cmap='coolwarm') m.show()

X = da['time\_study'].values.reshape(-1,1) y = da['Marks']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = LinearRegression() model.fit(X\_train, y\_train) y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred) r2 = r2\_score(y\_test, y\_pred) print("Mean Squared Error:", mse) print("R-squared:", r2)

m.figure(figsize=(10, 6))

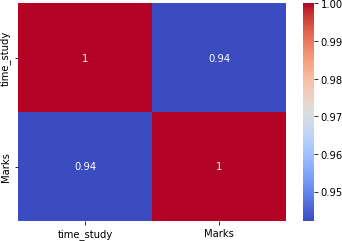
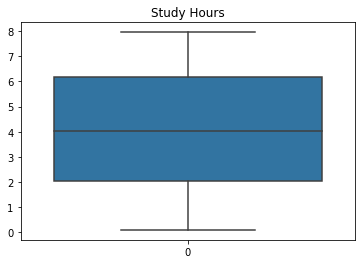
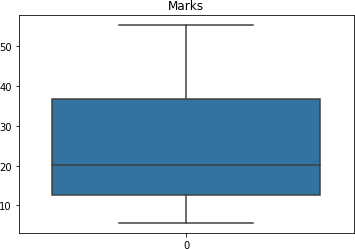
m.scatter(X\_test, y\_test, color='blue', label='Actual Data') m.plot(X\_test, y\_pred, color='red', linewidth=2, label='Regression Line') m.title('Linear Regression: Salary vs. Years of Experience') m.xlabel('Years of Experience');m.ylabel('Salary')

m.legend() m.show()

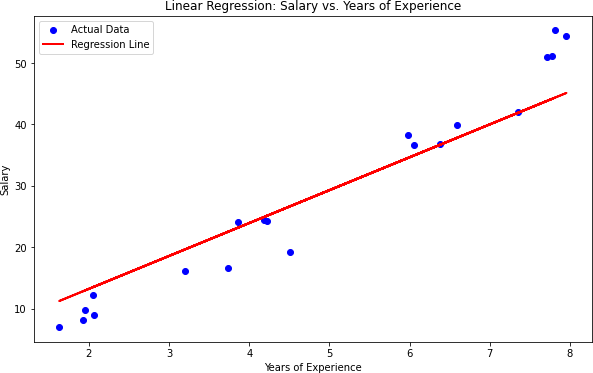
Hours= [[20]]# Replace with the desired experience value predicted\_marks = model.predict(Hours)

print(f"Predicted Marks for {Hours[0][0]} Hours : {predicted\_marks[0]}")

## Output:









1. **Boston housing price**

## Preprocessing & Exploration Code:

import pandas as p import seaborn as s

import matplotlib.pyplot as m

da=p.read\_csv("C:\\Users\\Desktop\\Data\\boston.csv") da.info()

print(da.describe()) m.figure(figsize=(18,10))

s.heatmap(data=da.corr(),annot=True,cmap='coolwarm')

m.show()

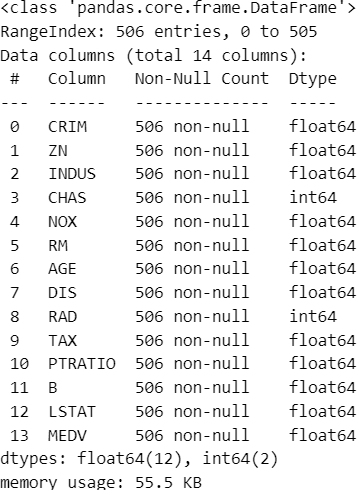
m.figure(figsize=(18,10)) s.histplot(data=da,x='MEDV',bins=30,kde=True) m.title('Distribution of Pricing')

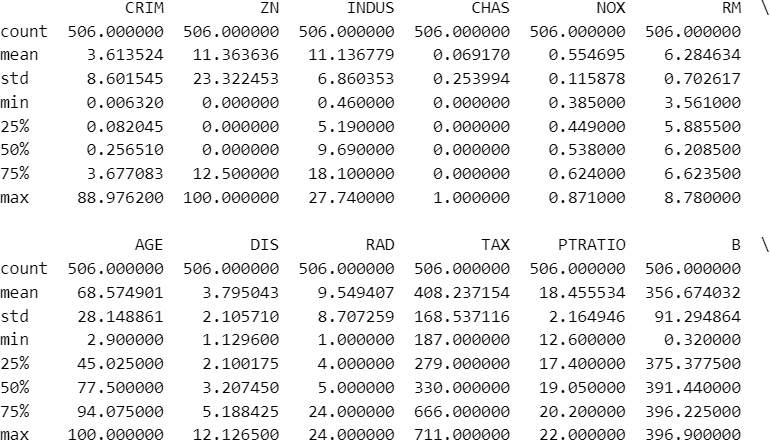
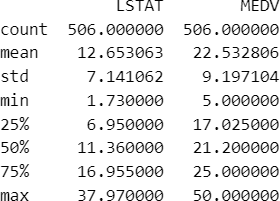
m.show()

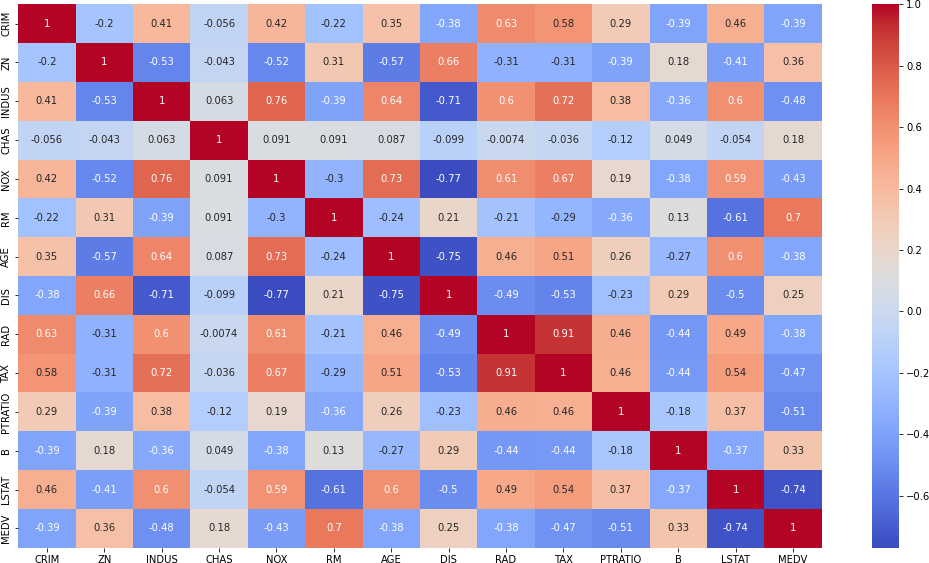
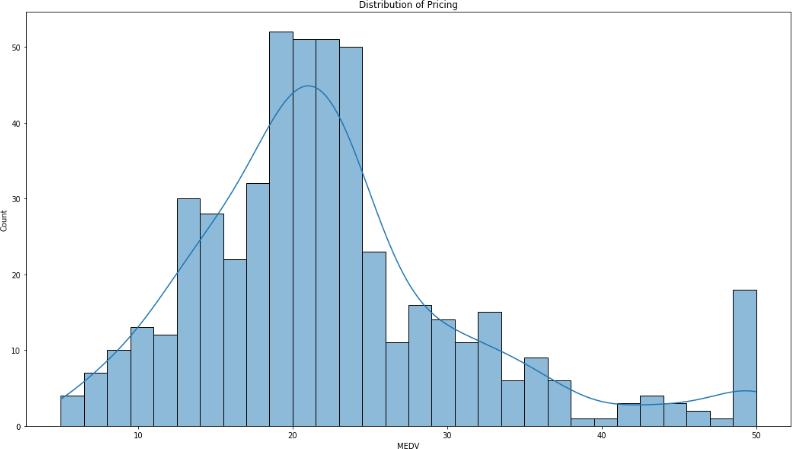
m.figure(figsize=(18,10)) s.scatterplot(data=da,x='RM',y='MEDV') m.title('Number of rooms to pricing') m.show()

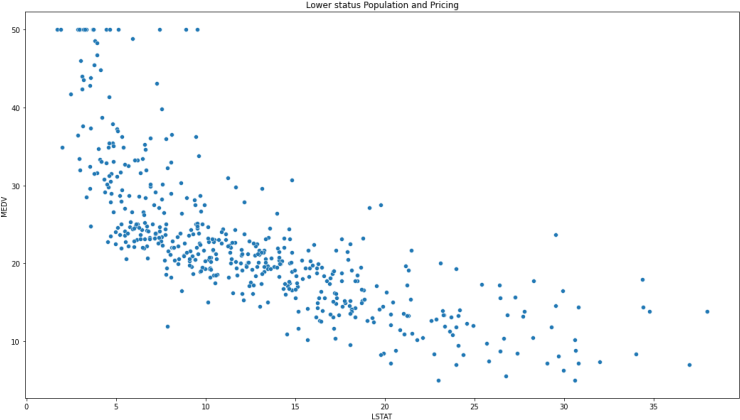
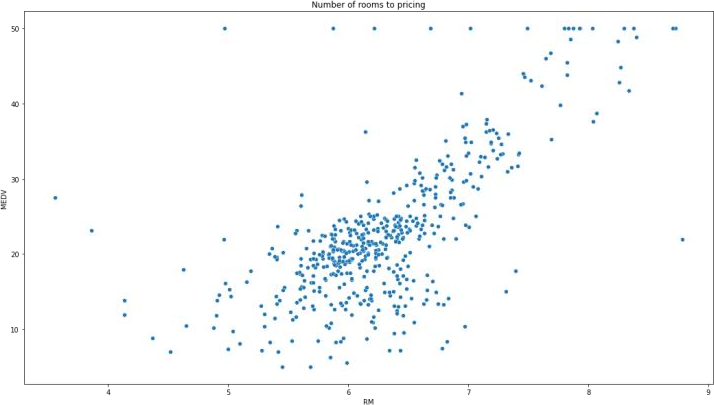
m.figure(figsize=(18,10)) s.scatterplot(data=da,x='LSTAT',y='MEDV') m.title('Lower status Population and Pricing') m.show()

## Output:







## Splitting

import numpy as n

from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

X = da.drop('MEDV', axis=1) y = da['MEDV']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

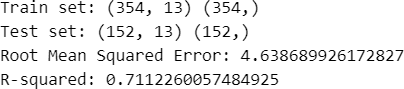
print("Train set:", X\_train.shape, y\_train.shape) print("Test set:", X\_test.shape, y\_test.shape)

model = LinearRegression() model.fit(X\_train, y\_train) y\_pred = model.predict(X\_test)

rmse = n.sqrt(mean\_squared\_error(y\_test, y\_pred)) r2 = r2\_score(y\_test, y\_pred)

print("Root Mean Squared Error:", rmse) print("R-squared:", r2)

## Output:



1. **Cricket match result**

## Data Preprocessing

**Code:**

import pandas as p da=p.read\_csv("C:\\Users\\Desktop\\Data\\matches.csv") da.info()

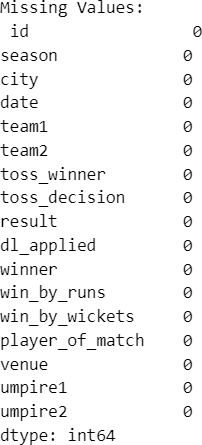
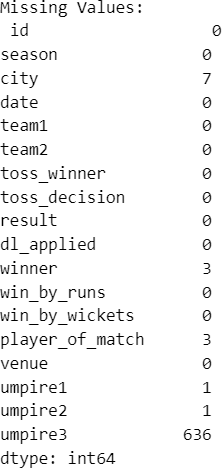
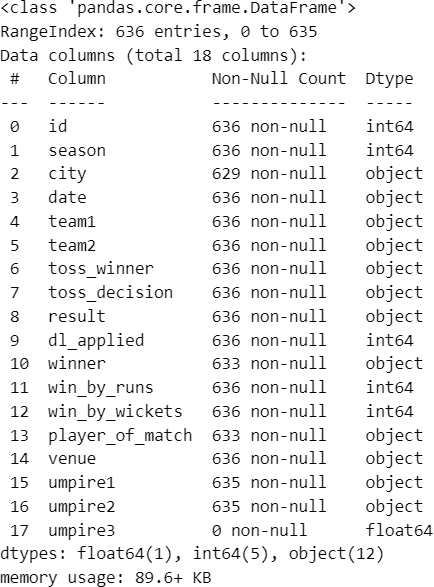
#PRE\_PROCESSING

miss=da.isna().sum() print("\nMissing Values:\n",miss)

da.drop('umpire3', axis=1, inplace=True) da.dropna(axis=0,inplace=True)

# da.drop('date',axis=1,inplace=True) print("\nMissing Values:\n",da.isna().sum())

## Output:



1. **Data Exploration**

## Code:

import matplotlib.pyplot as m import seaborn as s

# Function to create a histogram def histo(da, col, title, xlabel):

m.figure(figsize=(8, 6)) s.histplot(da[col], bins=20, kde=True) m.title(title)

m.xlabel(xlabel) m.ylabel('Frequency') m.show()

# Function to create a count plot def count\_plo(da, x, title, xlabel):

m.figure(figsize=(15, 7)) s.countplot(x=x, data=da, palette='Set2') m.title(title)

m.xlabel(xlabel) m.ylabel('Count') m.xticks(rotation=90) m.tight\_layout() m.show()

# Function to create a stacked bar chart

def sta\_bar(da, x, y, title, xlabel, ylabel, legend\_title=None): dp = da.groupby([x, y]).size().unstack()

dp.plot(kind='bar', stacked=True, figsize=(10, 6)) m.title(title)

m.xlabel(xlabel) m.ylabel(ylabel) if legend\_title:

m.legend(title=legend\_title) m.show()

#DATA\_EXPLORATION

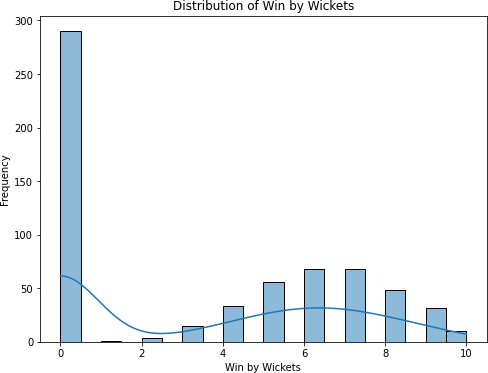
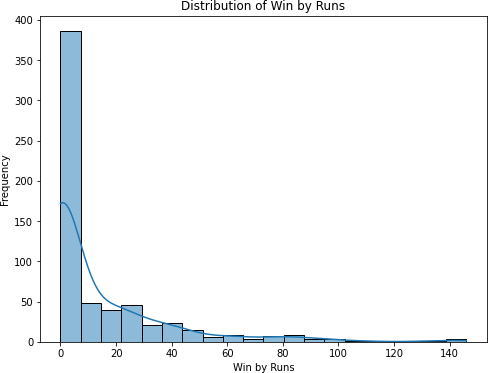
histo(da, 'win\_by\_runs', 'Distribution of Win by Runs', 'Win by Runs') histo(da, 'win\_by\_wickets', 'Distribution of Win by Wickets', 'Win by Wickets') count\_plo(da, 'city', 'Count of Matches at Each City', 'City')

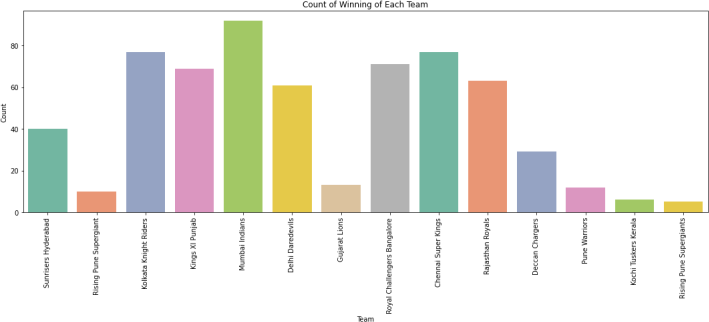
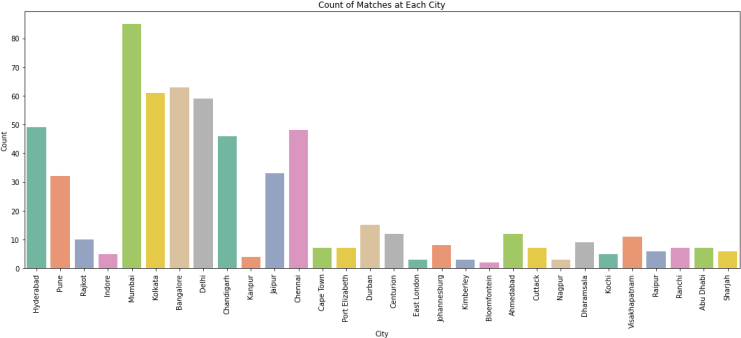
count\_plo(da, 'winner', 'Count of Winning of Each Team', 'Team')

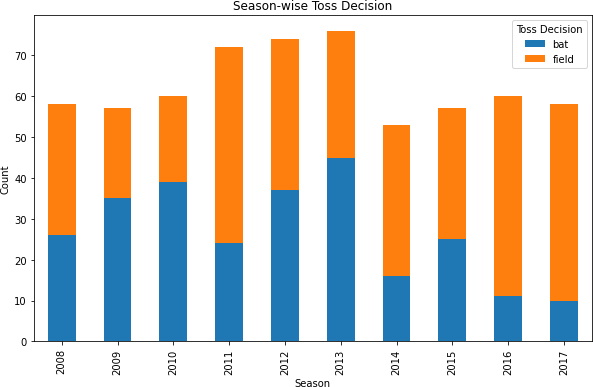
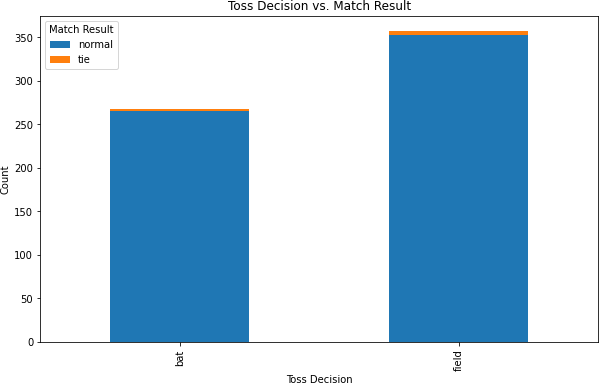
sta\_bar(da, 'toss\_decision', 'result', 'Toss Decision vs. Match Result', 'Toss Decision', 'Count', legend\_title='Match Result')

sta\_bar(da, 'season', 'toss\_decision', 'Season-wise Toss Decision', 'Season', 'Count', legend\_title='Toss Decision')

## Output:







1. **Splitting**

## Code:

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split

#Transformation

categorical\_columns = ['city', 'team1', 'team2', 'toss\_winner', 'toss\_decision', 'result', 'player\_of\_match', 'venue', 'umpire1', 'umpire2','winner','date']

label\_encoders = {}

for column in categorical\_columns: label\_encoders[column] = LabelEncoder()

da[column] = label\_encoders[column].fit\_transform(da[column])

X = da.drop('winner', axis=1) y = da['winner']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

print("Train set:", X\_train.shape, y\_train.shape) print("Test set:", X\_test.shape, y\_test.shape)

## Output:



1. **Performance of a cricket player**

## Preprocessing

**Code:**

import pandas as p da=p.read\_csv("C:\\Users\\Desktop\\Data\\Ms\_Dhoni.csv") da.info()

#PRE\_PROCESSING

miss=da.isna().sum() print("\nMissing Values:\n",miss) da.dropna(inplace=True)

print("\nMissing Values:\n",da.isna().sum())

## Output:

1. **Exploration Code:**

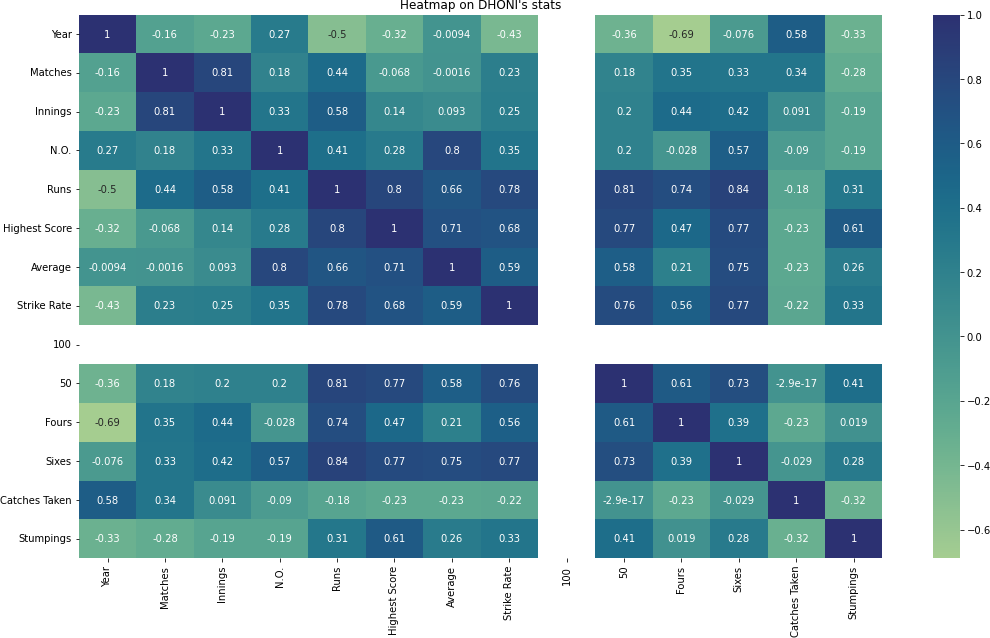
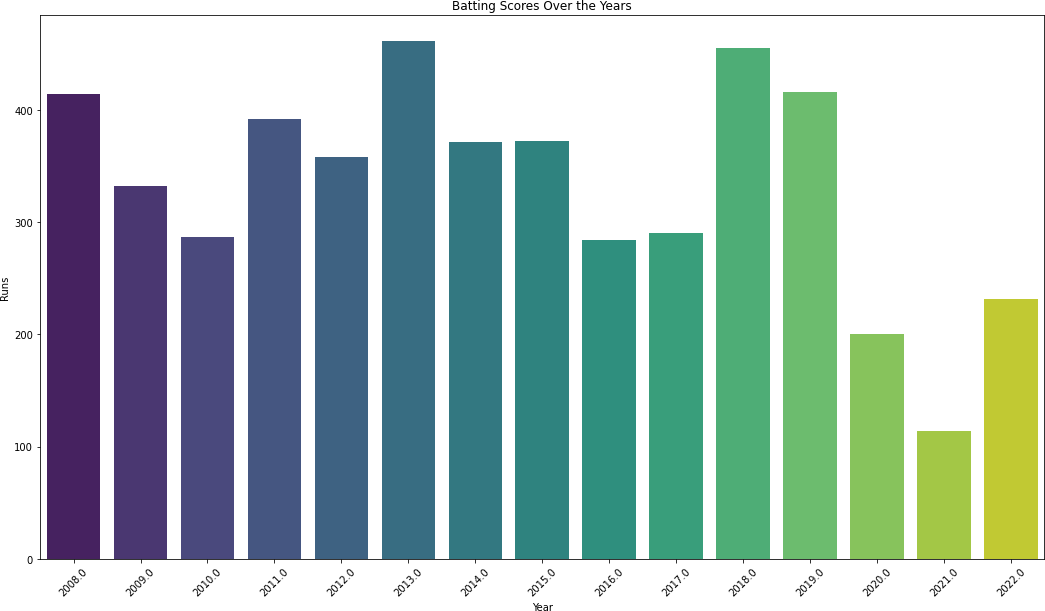
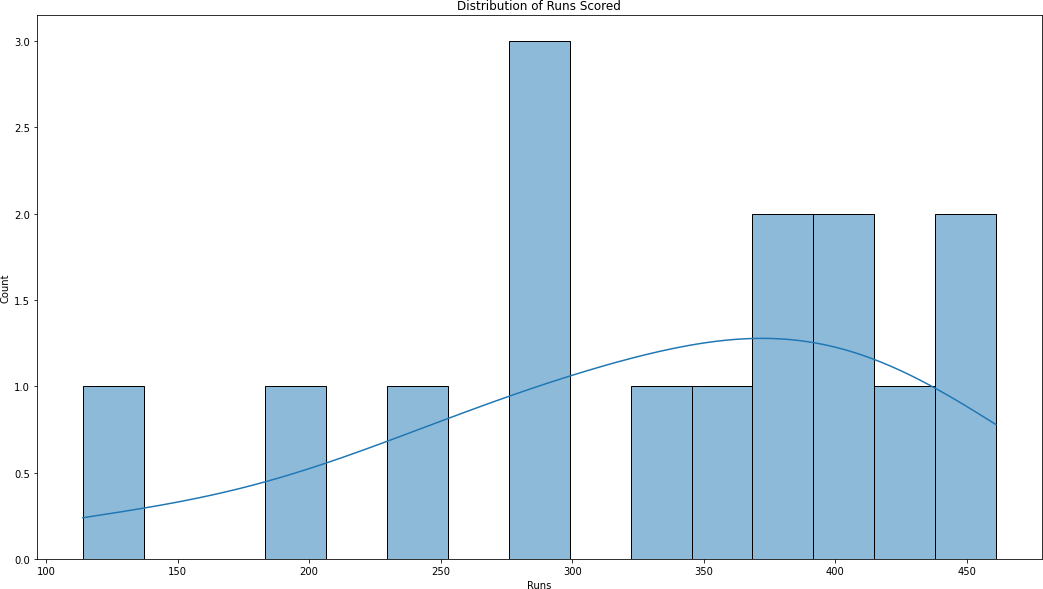
import matplotlib.pyplot as m import seaborn as s m.figure(figsize=(18,10))

s.histplot(da['Runs'], bins=15, kde=True) m.title('Distribution of Runs Scored') m.show()

m.figure(figsize=(18,10)) s.barplot(data=da,x='Year',y='Runs',palette='viridis') m.title('Batting Scores Over the Years') m.xlabel('Year');m.ylabel('Runs') m.xticks(rotation=45);m.show() m.figure(figsize=(18,10)) s.heatmap(data=da.corr(),annot=True,cmap='crest') m.title("Heatmap on DHONI's stats")

m.show()

## Output:



1. **Splitting**

## Code:

from sklearn.model\_selection import train\_test\_split X = da.drop('Average', axis=1)

y = da['Average']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

print("Train set:", X\_train.shape, y\_train.shape) print("Test set:", X\_test.shape, y\_test.shape)

## Output:



1. **Crop yield**

## Preprocessing

**Code:**

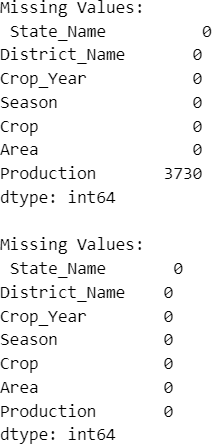
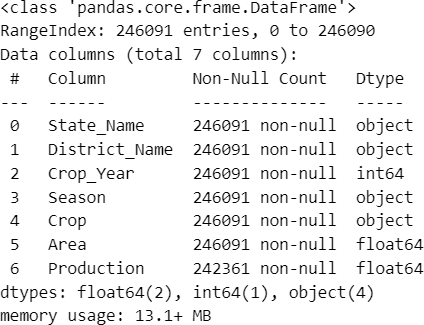
import pandas as p da=p.read\_csv("C:\\Users\\Desktop\\Data\\crop\_production.csv") da.info()

#PRE\_PROCESSING

miss=da.isna().sum() print("\nMissing Values:\n",miss) da.dropna(inplace=True)

print("\nMissing Values:\n",da.isna().sum())

## Output:



1. **Data Exploration Code:**

import matplotlib.pyplot as m import seaborn as s

m.figure(figsize=(18,10)) s.countplot(data=da, x='State\_Name') m.title("Sates Which grow Crops") m.xticks(rotation=90)

m.show()

m.figure(figsize=(18,10)) s.countplot(data=da,x='Crop\_Year') m.title('Crops Grown In an year') m.xticks(rotation=90)

m.show()

m.figure(figsize=(12, 6))

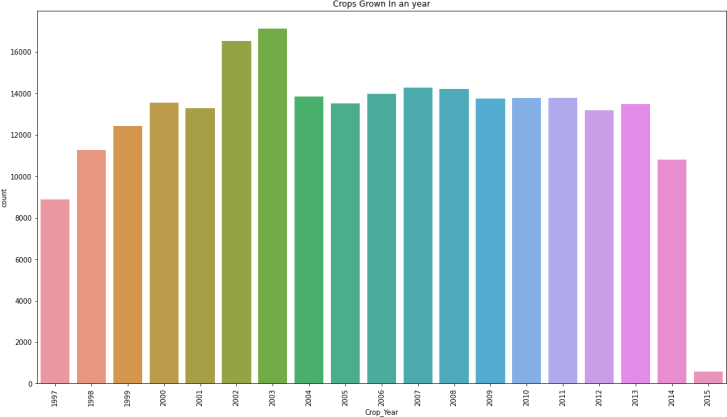
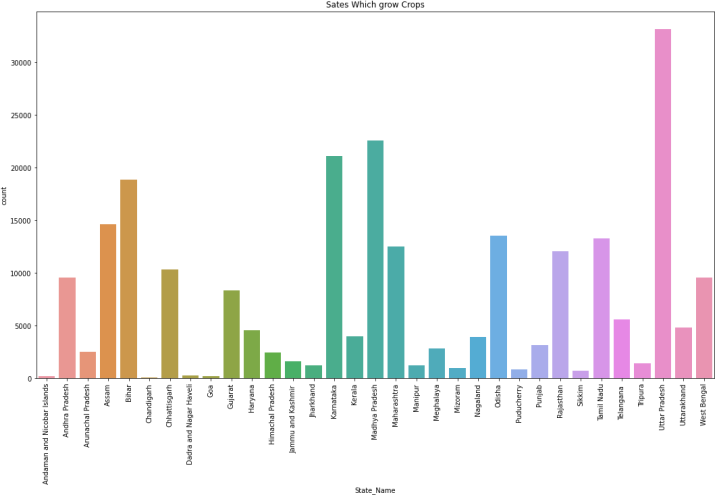
s.barplot(x=da['Crop\_Year'], y=da['Production'],data=da) m.title('Rice Production by Year')

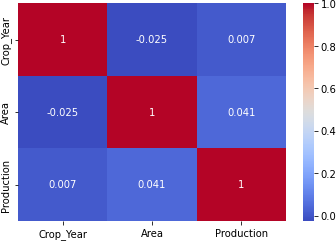
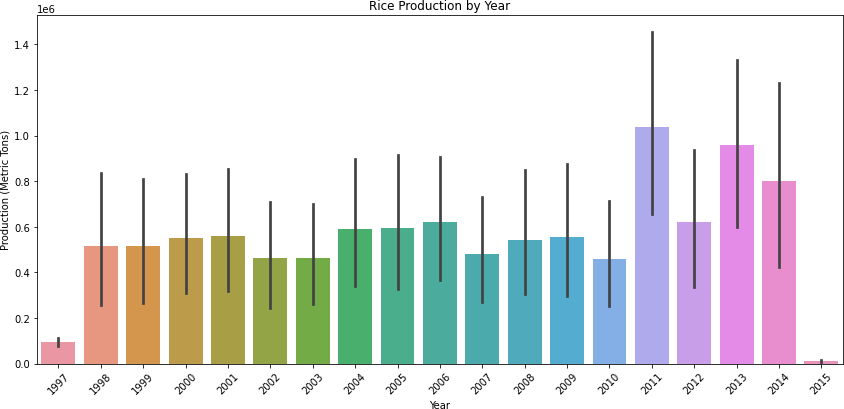
m.xlabel('Year') m.ylabel('Production (Metric Tons)') m.xticks(rotation=45) m.tight\_layout()

m.show()

sn=da.select\_dtypes(exclude=['object']) s.heatmap(data=sn.corr(),annot=True,cmap='coolwarm') m.show()

## Output:





1. **Splitting**

## Code:

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split

#Transformation

categorical\_columns = ['State\_Name','District\_Name','Season','Crop'] label\_encoders = {}

for column in categorical\_columns: label\_encoders[column] = LabelEncoder()

da[column] = label\_encoders[column].fit\_transform(da[column])

X = da.drop('Production', axis=1) y = da['Production']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

print("Train set:", X\_train.shape, y\_train.shape) print("Test set:", X\_test.shape, y\_test.shape)

**Output:**



# Week -07

1. **Regression algorithms**
   * **Decision Tree Regressor**

## Random Forest Regressor

* + **Support Vector Regression Code:**

import pandas as p

from sklearn.metrics import r2\_score

from sklearn.model\_selection import train\_test\_split

# Breast cancer dataset

data = p.read\_csv("C:\\Users\\Desktop\\Data\\brca1.csv") data.info()

data.drop(["id"],axis=1,inplace=True) M=data[data.diagnosis=="M"] B=data[data.diagnosis=="B"]

data.diagnosis=[1 if i == "M" else 0 for i in data.diagnosis] x=data.drop(["diagnosis"],axis=1)

y=data.diagnosis.values

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.3,random\_s tate=42)

from sklearn.tree import DecisionTreeRegressor model=DecisionTreeRegressor() model.fit(x\_train,y\_train) y\_pred=model.predict(x\_test)

print("\nAccuracy of the model using Decision tree regression algorithm is ",r2\_score(y\_test,y\_pred))

from sklearn.ensemble import RandomForestRegressor model1 = RandomForestRegressor() model1.fit(x\_train,y\_train)

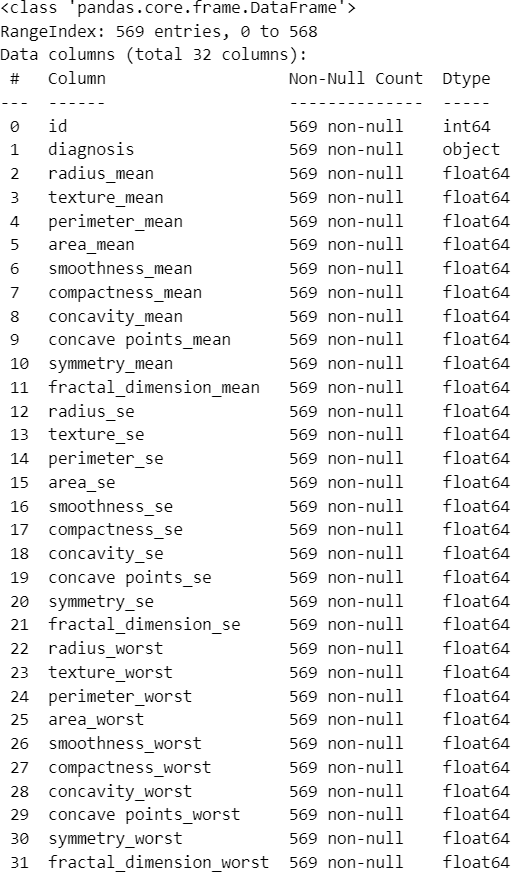
y\_pred1 = model1.predict(x\_test)

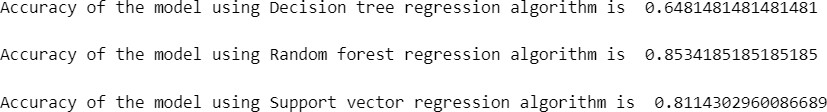
print("\nAccuracy of the model using Random forest regression algorithm is ",r2\_score(y\_test,y\_pred1))

from sklearn.svm import SVR model2 = SVR(kernel='rbf') model2.fit(x\_train,y\_train) y\_pred2 = model2.predict(x\_test)

print("\nAccuracy of the model using Support vector regression algorithm is ",r2\_score(y\_test,y\_pred2))

## Output:





1. **Build decision tree-based model for Breast Cancer Wisconsin (diagnostic) dataset.[Classifier]**

## Code:

import numpy as n import pandas as p import seaborn as s

import matplotlib.pyplot as m

from sklearn.metrics import r2\_score

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score,confusion\_matrix

data = p.read\_csv("C:\\Users\\Desktop\\Data\\brca1.csv") data.info()

data.drop(["id"],axis=1,inplace=True) M=data[data.diagnosis=="M"] B=data[data.diagnosis=="B"]

m.title("Malignant vs Benign Tumor") m.xlabel("Radius Mean"); m.ylabel("Texture Mean")

m.scatter(M.radius\_mean,M.texture\_mean,color='red',label='Malignant',alph a=0.3)

m.scatter(B.radius\_mean,B.texture\_mean,color='lime',label='Bengin',alpha= 0.4)

m.legend(); m.show()

data.diagnosis=[1 if i == "M" else 0 for i in data.diagnosis] x=data.drop(["diagnosis"],axis=1)

y=data.diagnosis.values

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.3,random\_s tate=42)

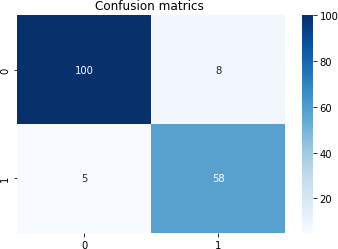
dt=DecisionTreeClassifier() dt.fit(x\_train,y\_train) y\_pred=dt.predict(x\_test)

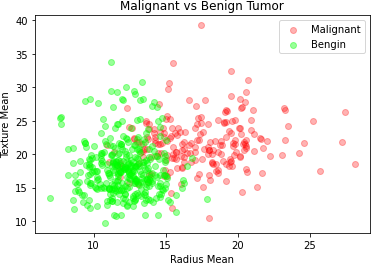
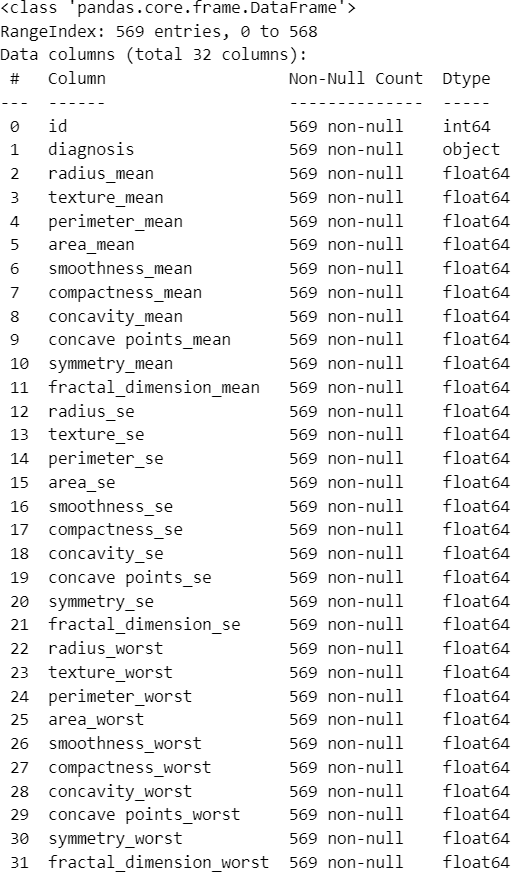
#Confusion Matrix

cm = confusion\_matrix(y\_test,y\_pred) s.heatmap(cm,annot=True,fmt='d',cmap="Blues") m.title("Confusion matrics ")

m.show()

print("Accuracy of the classifier model is",accuracy\_score(y\_test,y\_pred))



**Output:**



# Week – 08

## Logistic regression [using Fish prediction dataset] Code:

import pandas as p import seaborn as s

import matplotlib.pyplot as m

da=p.read\_csv("C:\\Users\\Desktop\\Data\\fish\_prediction.csv") print(da.isna().sum())

x=da.iloc[:,1:]

y=da.loc[:,'Species']

#Scaling

from sklearn.preprocessing import MinMaxScaler sca=MinMaxScaler()

sca.fit(x) x\_sca=sca.transform(x)

#Transformation of Y

from sklearn.preprocessing import LabelEncoder lb=LabelEncoder()

y=lb.fit\_transform(y)

from sklearn.model\_selection import train\_test\_split x\_train,x\_test,y\_train,y\_test=train\_test\_split(x\_sca,y,test\_size=0.2,rand om\_state=42)

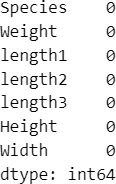
from sklearn.linear\_model import LogisticRegression mod=LogisticRegression()

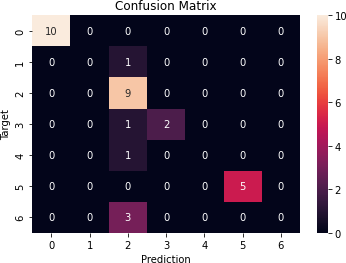
mod.fit(x\_train,y\_train) y\_pred=mod.predict(x\_test)

from sklearn.metrics import accuracy\_score print("\nAccuracy:",accuracy\_score(y\_test,y\_pred))

from sklearn.metrics import confusion\_matrix cm=confusion\_matrix(y\_test,y\_pred) s.heatmap(cm,annot=True) m.xlabel("Prediction"); m.ylabel('Target') m.title('Confusion Matrix'); m.show()

## Output:





1. **SVM Model [Using Fish prediction dataset]**

## Code:

import pandas as p import seaborn as s

import matplotlib.pyplot as m

da=p.read\_csv("C:\\Users\\Desktop\\Data\\fish\_prediction.csv") s.pairplot(data=da,hue='Species')

m.show()

x=da.iloc[:,1:]

y=da.loc[:,'Species']

#Transformation of Y

from sklearn.preprocessing import LabelEncoder lb=LabelEncoder()

y=lb.fit\_transform(y)

from sklearn.model\_selection import train\_test\_split x\_train,x\_test,y\_train,y\_test=train\_test\_split(x\_sca,y,test\_size=0.2,rand om\_state=42)

from sklearn.svm import SVC mod=SVC(kernel='rbf',random\_state=1,gamma='auto') mod.fit(x\_train,y\_train) y\_pred=mod.predict(x\_test)

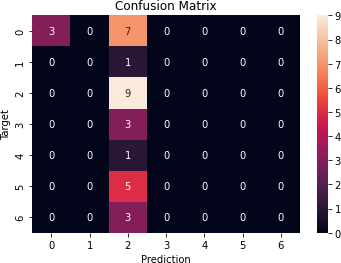
from sklearn.metrics import accuracy\_score print("\nAccuracy:",accuracy\_score(y\_test,y\_pred)\*100)

from sklearn.metrics import confusion\_matrix cm=confusion\_matrix(y\_test,y\_pred) s.heatmap(cm,annot=True) m.xlabel("Prediction"); m.ylabel('Target') m.title('Confusion Matrix'); m.show()

## Output:







1. **Random Forest Classifier model [Using Breast Cancer dataset] Code:**

import pandas as p import seaborn as s

import matplotlib.pyplot as m

from sklearn.model\_selection import train\_test\_split from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix

da=p.read\_csv("C:\\Users\\Desktop\\Data\\brca1.csv") da.drop(['id'],axis=1,inplace=True)

da.diagnosis=[1 if i=='M' else 0 for i in da.diagnosis]

x=da.drop(['diagnosis'],axis=1) y=da.diagnosis.values

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_s tate=30)

model1=RandomForestClassifier() model1.fit(x\_train,y\_train) y\_pred1=model1.predict(x\_test)

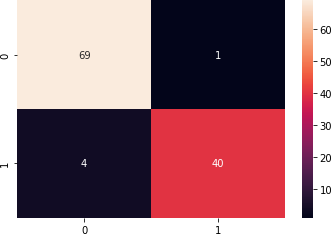
print("\nAccuracy of the model using Random Forest Regression alogorithm is",accuracy\_score(y\_test,y\_pred1))

cm=confusion\_matrix(y\_test,y\_pred1) s.heatmap(cm,annot=True)

m.show()

## Output:





**Week - 09**

## K-means [ using Mall Customers Data] Code:

import pandas as p

import matplotlib.pyplot as m da=p.read\_csv("C:\\Users\\Desktop\\Data\\Mall\_Customers.csv") x=da.iloc[:,[3,4]].values

from sklearn.cluster import KMeans li=[]

for i in range(1,11):

mod=KMeans(n\_clusters=i,init='k-means++', random\_state=42,n\_init=10) mod.fit(x)

li.append(mod.inertia\_)

m.plot(range(1,11), li) m.title("Elbow Method")

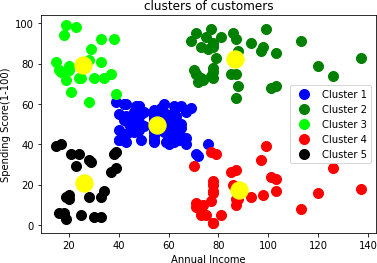
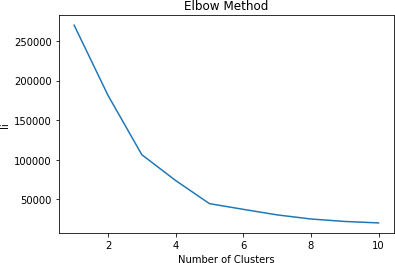
m.xlabel('Number of Clusters'); m.ylabel('li'); m.show()

mod=KMeans(n\_clusters=5,init='k-means++', random\_state=42,n\_init=10) y\_pred=mod.fit\_predict(x)

m.scatter(x[y\_pred==0,0],x[y\_pred==0,1],s=100, c='blue', label='Cluster 1') m.scatter(x[y\_pred==1,0],x[y\_pred==1,1],s=100,c='g', label='Cluster 2') m.scatter(x[y\_pred==2,0],x[y\_pred==2,1],s=100, c='lime', label='Cluster 3') m.scatter(x[y\_pred==3,0],x[y\_pred==3,1],s=100, c='red', label='Cluster 4') m.scatter(x[y\_pred==4,0],x[y\_pred==4,1],s=100, c='k', label='Cluster 5') m.scatter (mod.cluster\_centers\_[:,0],mod.cluster\_centers\_[:,1],s=300,c='yellow') m.title("clusters of customers")

m.xlabel("Annual Income"); m.ylabel("Spending Score(1-100)") m.legend();m.show()

## Output:



1. **Dimensionality Reduction [PCA] using iris Dataset Code:**

import numpy as np import pandas as pd

import matplotlib.pyplot as plt from sklearn import datasets

from sklearn.decomposition import PCA

iris=datasets.load\_iris() x=iris.data

y=iris.target print(x.shape) print(y.shape)

pca=PCA(n\_components=2) pca.fit(x) print(pca.components\_)

x=pca.transform(x) print(x.shape)

plt.scatter (x[:,0],x[:,1],c=y)

from sklearn.tree import DecisionTreeClassifier

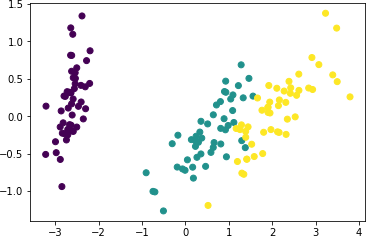
from sklearn.model\_selection import train\_test\_split from sklearn.metrics import accuracy\_score

x\_train, x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2) res=DecisionTreeClassifier()

res.fit(x\_train,y\_train)

y\_predict=res.predict(x\_test) print(accuracy\_score (y\_test,y\_predict))

## Output:



**Week - 10**

## Shallow Neural Network Code:

import matplotlib.pyplot as plt import tensorflow as tf

from keras.utils import to\_categorical layers = tf.keras.layers

from keras.datasets import mnist

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

x\_train = x\_train.astype('float32') / 255.0 x\_test = x\_test.astype('float32') / 255.0

# one-hot encode the labels

y\_train = to\_categorical(y\_train, 10) y\_test = to\_categorical(y\_test, 10)

plt.figure(figsize=(10,8)) for i in range(1, 15):

# image in ith position of grid plt.subplot(5, 5, i)

plt.imshow(x\_train[i], cmap=plt.get\_cmap('gray')) plt.show()

model = tf.keras.Sequential() model.add(layers.Flatten()) model.add(layers.Dense(5, activation='sigmoid')) model.add(layers.Dense(10, activation='softmax'))

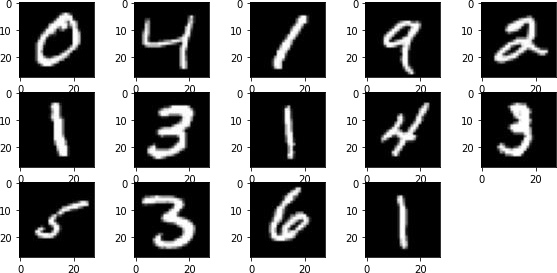
model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

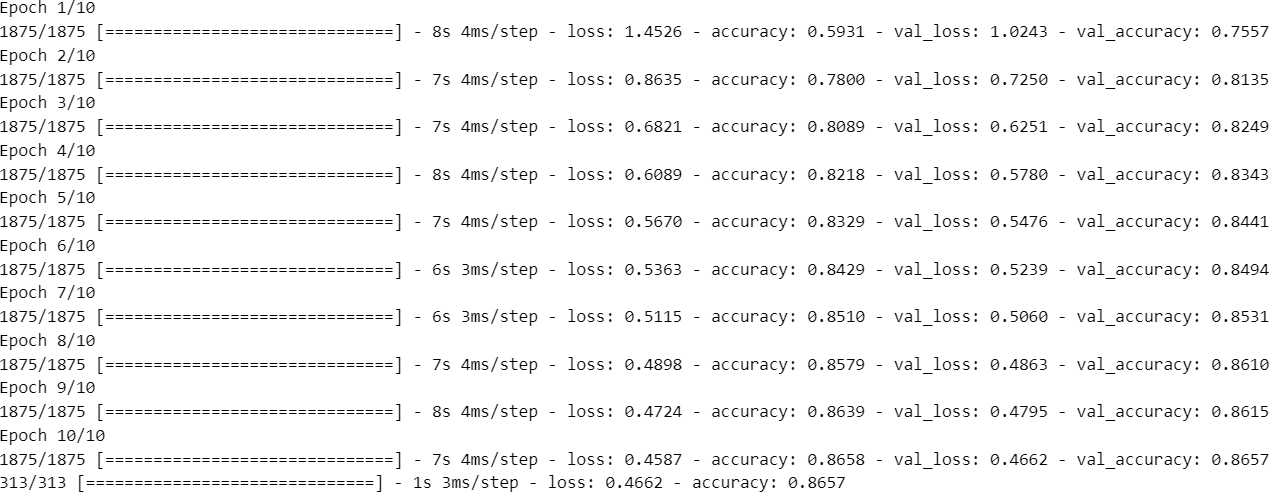
model.fit(x\_train, y\_train, epochs=10, batch\_size=32, validation\_data=(x\_test, y\_test))

loss, accuracy = model.evaluate(x\_test, y\_test) print('\n\nTest loss', loss)

print('Test accuracy', accuracy)

## Output:







1. **Deep Neural Network Code:**

import matplotlib.pyplot as plt import tensorflow as tf

from keras.utils import to\_categorical from keras.datasets import fashion\_mnist

layers = tf.keras.layers

(x\_train, y\_train), (x\_test, y\_test) = fashion\_mnist.load\_data() y\_train = to\_categorical(y\_train, num\_classes=10)

y\_test = to\_categorical(y\_test, num\_classes=10)

print('X\_train: ' + str(x\_train.shape)) print('Y\_train: ' + str(y\_train.shape)) print('X\_test: ' + str(x\_test.shape)) print('Y\_test: ' + str(x\_test.shape))

for i in range(1, 15): plt.subplot(5, 5, i)

plt.imshow(x\_train[i], cmap=plt.get\_cmap('gray')) plt.show()

model = tf.keras.Sequential() model.add(layers.Flatten()) model.add(layers.Dense(64, activation='relu')) model.add(layers.Dense(64, activation='relu')) model.add(layers.Dense(64, activation='relu')) model.add(layers.Dense(64, activation='relu')) model.add(layers.Dense(10, activation='softmax'))

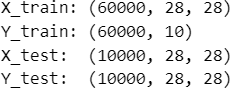
model.compile(optimizer='adam',loss='categorical\_crossentropy',metrics=[' accuracy'])

model.fit(x\_train, y\_train, epochs=10)

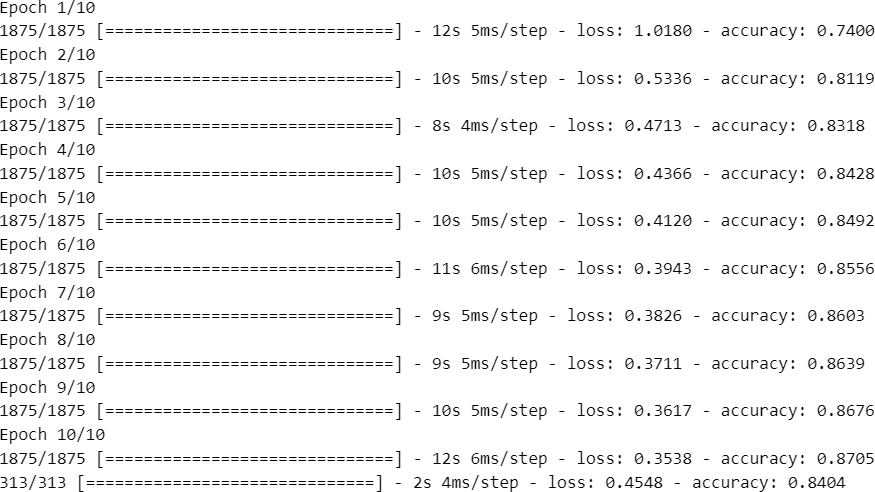
loss, accuracy = model.evaluate(x\_test, y\_test) print('\n\nTest loss', loss)

print('Test accuracy', accuracy)

## Output:









**Week-11**

## Tokenization [Sentence & Word]

**Code:**

from nltk.tokenize import sent\_tokenize from nltk.tokenize import word\_tokenize

f=open("C:\\Users\\Desktop\\Data\\da.txt") text=f.read()

print(text)

sent=sent\_tokenize(text)

print("Number of sentences:",len(sent)) for i in range(len(sent)):

print("\nSentence:",i+1,"\n",sent[i])

w=word\_tokenize(text) print("\nTotal Words:",len(w)) print(w)

## Output:



1. **N-Grams Code:**

from nltk.util import ngrams

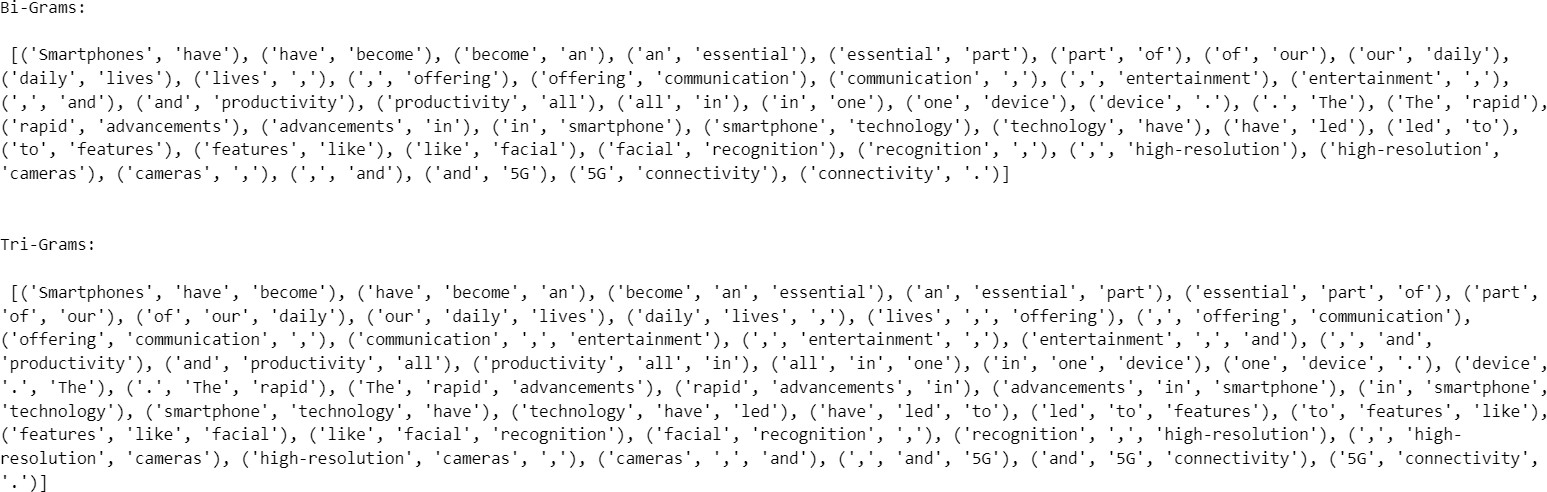
from nltk.tokenize import word\_tokenize

f=open("C:\\Users\\Desktop\\da.txt") text=f.read()

w=word\_tokenize(text)

print("Bi-Grams:\n\n",list(ngrams(w,2))) print("\n\nTri-Grams:\n\n",list(ngrams(w,3)))

## Output:



1. **Frequency Distribution of Words**

## Code:

import pandas as p

import matplotlib.pyplot as m

from nltk.probability import FreqDist

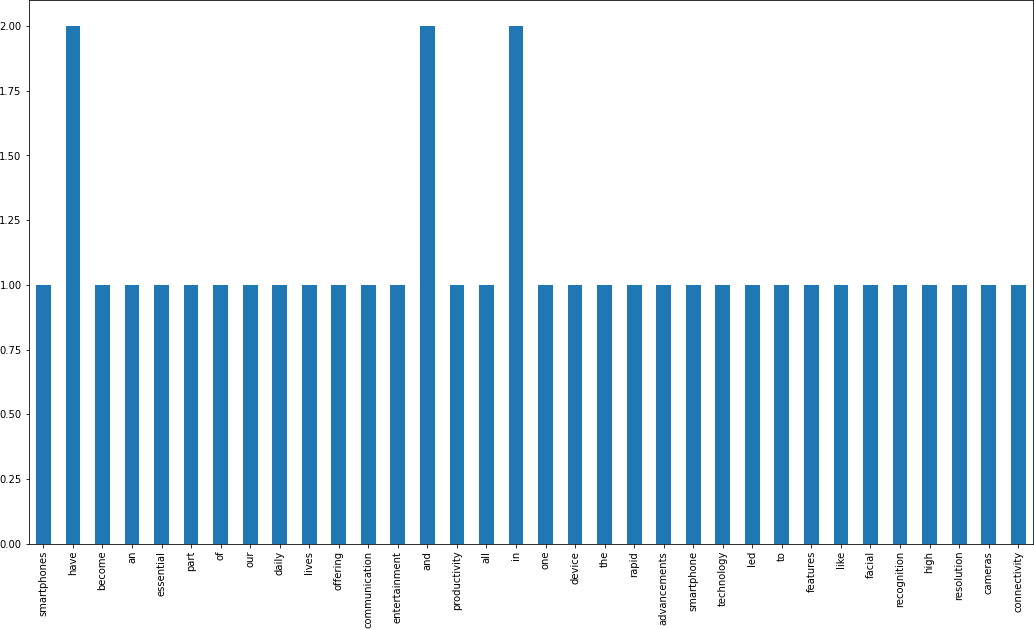
freq=FreqDist(w)

print("Count of and word:",freq['and'])

freq=p.Series(dict(freq)) m.figure(figsize=(18,10)) freq.plot(kind='bar') m.show()

## Output:





1. **Removing Stop-words.**

## Code:

import nltk

import matplotlib.pyplot as m import pandas as p

from nltk.tokenize import word\_tokenize from nltk.probability import FreqDist

w = word\_tokenize(text)

print("Before Removal of Stopwords words count:",len(w)) stop\_w = nltk.corpus.stopwords.words('english') removed\_stopw = []

for i in w:

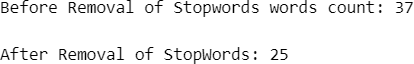
if i not in stop\_w: removed\_stopw.append(i)

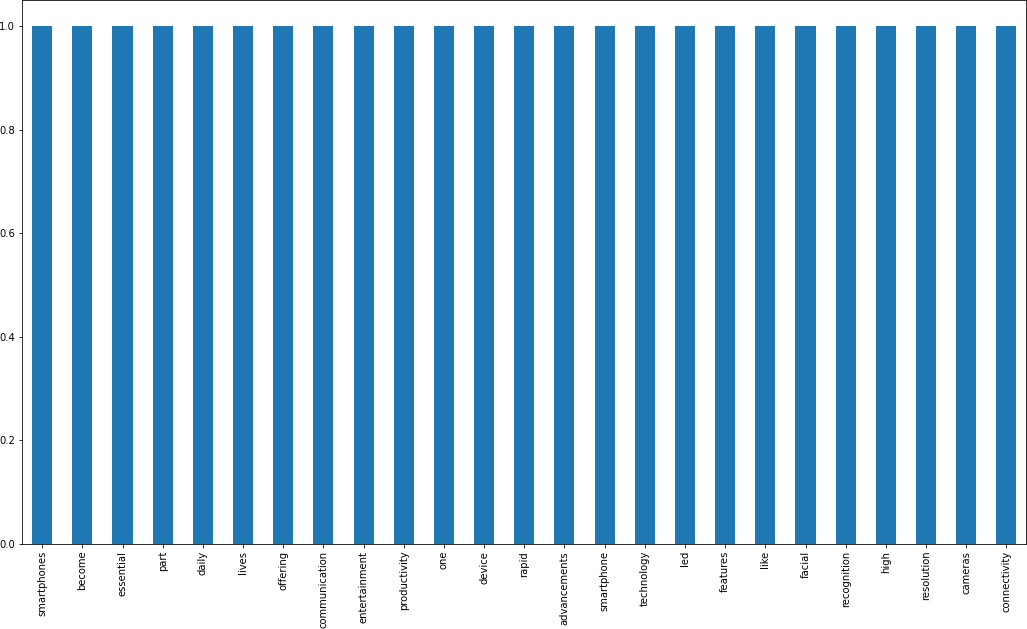
print("\nAfter Removal of StopWords:",len(removed\_stopw)) freq = FreqDist(removed\_stopw)

freq\_series = p.Series(dict(freq))

m.figure(figsize=(18, 10)) freq\_series.plot(kind='bar') m.show()

## Output:





1. **Word-cloud**

## Code:

import matplotlib.pyplot as m f=open("C:\\Users\\Desktop\\da.txt") text=f.read()

from wordcloud import WordCloud, STOPWORDS stop\_w=set(STOPWORDS)

wc=WordCloud(width=800,height=800,

background\_color='black',colormap='plasma', stopwords=stop\_w, min\_font\_size=10).generate(text)

m.figure(figsize=(8,8),facecolor=None) m.imshow(wc)

m.axis('off') m.tight\_layout(pad=0) m.show()

## Output:



1. **Stemming & Lemmatization Code:**

import re

from nltk.tokenize import word\_tokenize

f=open("C:\\Users\\Desktop\\da.txt") text=f.read()

text=text.lower()

text=re.sub('[^A-Za-z0-9]+',' ',text)

text=re.sub("\S\*\d\S\*","",text).strip() print(text)

w=word\_tokenize(text,preserve\_line=True)

from nltk.stem import PorterStemmer ps=PorterStemmer() ps\_st=[ps.stem(i) for i in w] print("\nStemming:\n\n",ps\_st)

from nltk import WordNetLemmatizer wnl=WordNetLemmatizer() lema=[wnl.lemmatize(u) for u in w]

print("\n Lemmatization:\n\n",lema)

## Output:

