# Experiment: 1 Working with NumPy and Pandas in Python

## Aim:

To understand the fundamentals and application of NumPy library In Machine Learning

## Operations:

1. Importing & Checking version
2. Array Creation in NumPy
3. Array Operations in NumPy
4. Importing Pandas Library
5. Creating n series in Pandas
6. Creating Data frame in Pandas
7. Data Frame Operations

## Algorithm:

1. Import the library
2. Check the version of the library
3. Create the variable with object and input data as input arguments
4. Create a series using Pandas library
5. Create a data frame using Pandas library
6. Print the output

## Program:

1. **Importing & Checking version**

import numpy as np np. version. version

**Output** '16.5' [Based on the version in the system]

## Array Creation in NumPy

* 1. **Creating ID array**

first\_array = np.array([ 1,2,3]) print(first\_array)

**Output** [1 2 3]

## Creating 2D array

second\_array = np.array ([(4,5,6),(7,8,9)]) print(second\_array)

**Output** [[4 5 6] [7 8 9]]

## Creating 3D array

third\_array=p.array([(10,11,12),(13,14,15),(16,17,18),(13,14,15)])

print(third\_array)

**Output** [[[10 11 12] [13 14 15]] [[16 17 18] [13 14 151]

## 2.4 Array of Zeros

zero\_array = np.zeros((2,2)) print(zero\_array)

**Output** [[0. 0] [0. 0.]

## Array of Ones

one\_array = np.ones((3,4)) print(one\_array)

Output [[1. 1. 1.1] [1. 1. 1.1] [1. 1. 1.1]]

* 1. **Matrix using NumPy** a = np.matrix('1 2; 3 4') print(a)

**Output** matrix (1, 2), (3, 4]]) **3.Array Operations in NumPy 3.1.Create a Matrix**

my\_matrix = np.array([(11,17),(23,25)]) print(my\_matrix)

**Output** [[11 17] [23 25]]

## Transpose Operation

matrix \_ transpose =np.transpose(my\_matrix) print(matrix\_transpose)

**Output** [ [11 23] [17 25]]

## Determinant Operation

det = np.linalg.det(my\_matrix)

print(det)

**Output** -115.99999999999999

## Inverse Operation

inverse = np. linalg.inv(my\_matrix) inverse

**Output** array([[-0.21551724, 0.14655172],

[ 0.19827586, -0.09482759]])

## Resize an Array

**Note**: Please use the array with ones which was created above arr\_ones.resize((4, 1 ))

art\_ones

**Output**array ([ [1. ],

[1. ],

[1. ],

[1. ]])

## Pandas:

1. **Importing & Checking version**

import pandas as pd

## Creating a series in Pandas

alphabet pd.Series(1,2,3,4),index=[‘A’,’B’,’C’,’D’]) print(alphabet)

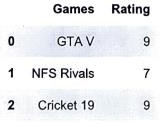
## Output

|  |  |
| --- | --- |
| A | 1 |
| B | 2 |
| C | 3 |
| D | 4 |

dtype : int64

## Creating a dataframe in Pandas

data {'Games': ['GTA V','NFS Rivals','Cricket 19'],’Rating’:[9,7,9]} dataframe =pd.DataFrame(data,columns=[‘Games’, ’Rating’]) dataframe



## Data Frame Operations

* 1. **Creating a Data frame with Random Numbers**

Random =pd.DataFrame(np.random.randint(0,300,size=(20,4)),columns=list('ABCDE')) random

## Output:



**1.2. Saving a Data frame**

random**.**to\_csv('C:/Users/Admin/Documents/VIKKI 4TH/Pandas.csv')

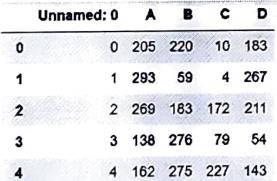
**Note :** Please give the location where you want to save the document along with document name and the extension. Upon saving, please go the given location and fetch the file

## Data Manipulation

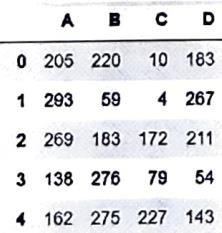
**5.1 Importing external data**

data**=**pd**.**read\_csv('C:/Users/Admin/Documents/VIKKI 4TH/Pandas.csv') data

## Output:



* 1. **Dropping a Data frame** data.drop(‘Unnamed: 0’, axis=1**) Output:**



## Shape of Data frame

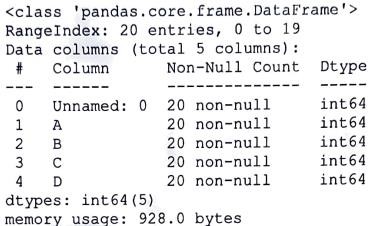
data.shape

**Output:**(20,5)

## Get information about the Data frame

data.info()

## Output:

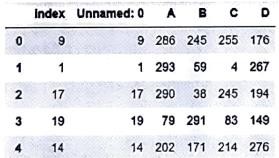


* 1. **Shuffling the data frame**

from sklearn.utils import shuffle

shuffle\_data = shuffle(data).reset\_index() shuffle data

## Output:



**Result:**

The experiment aimed at understanding the fundamentals and application of the NumPy library in machine learning.

# Experiment 2: Data Visualization using Matplotlib and Seaborn

**Aim:**To understand the fundamentals of Data Visualization and extracting insight using matplotlib and Seaborn

## Operations:

* + 1. Importing Matplotlib library
    2. Creating Data for visualization
    3. Data Visualization using Matplotlib
    4. Importing Seaborn library
    5. Advanced Data Visualization using Seaborn

## Algorithm:

1. Import the library
2. Create data
3. Perform data visualization
4. Print the graph

## Program:

1. **Import library**

import matplotlib.pyplot as pit

%matplotlib inline

## Creating data

movies = ['Interstellar', 'Inception', 'Infinity War' ,'Dune', 'Harry Potter','Oppie','FordvsFerrari']

percentage = [9.5,8.5,9,8,9,9,7]

**Output**: ['Interstellar', 'Inception', 'Infinity War' ,'Dune', 'Harry Potter','Oppie','FordvsFerrari']

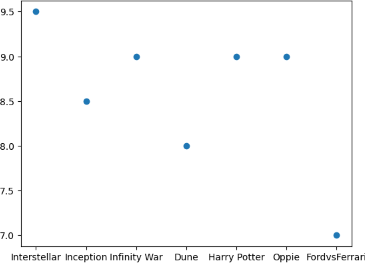
[9.5,8.5,9,8,9,9,7]

## Data Visualization using Matplotlib

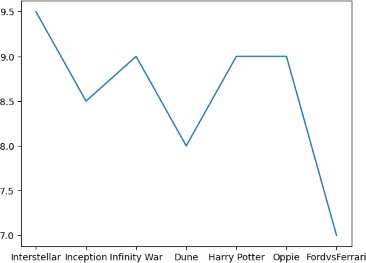
* 1. **Scatter Plot**

plt.scatter(movies,percentage)

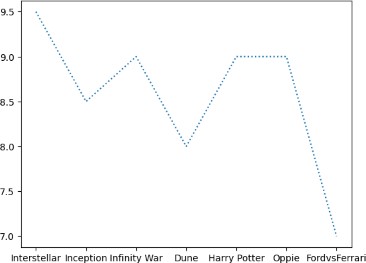
Output:



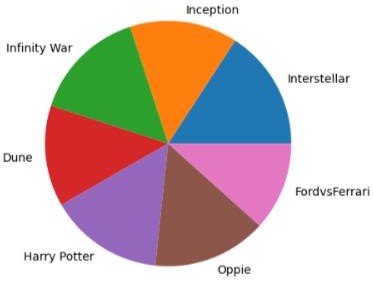
## 3.2 Scatter Plot

plt.plot(movies, percentage, linestyle= 'solid') plt.show()

**7.3. Line plot with dotted line** plt.plot(movies, percentage, linestyle= 'dotted') plt.show()



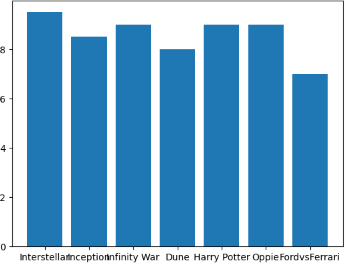
## 3.4. Pie Chart

plt.pie(percentage, labels = movies)

## 3.4 Bar Plot

plt.bar(movies , percentage, linestyle= 'dotted')

## Output:



1. **Importing Seaborn Library**

## Importing Seaborn Library and other required libraries

import numpy as np import pandas as pd

import matplotlib.pyplot as pit

%matplotlib inline

import seaborn as sns

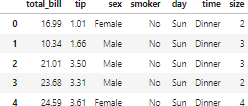
**Note**: Creating data, plotting using seaborn needs other dependent libraries

## Check for existing default datasets in seaborn

sns.get\_dataset\_names()

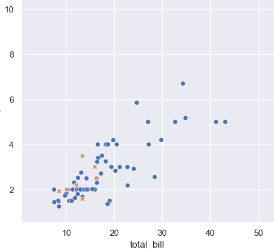
* 1. **Loading tips dataset** tip**=**sns**.**load\_dataset('tips') tip

tip.tail()



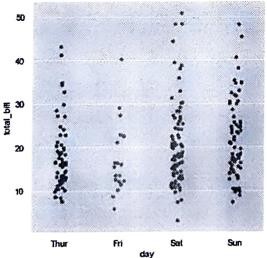
## Relational Plot

sns.relplot(x='total\_bill', y='tip', data=tip)



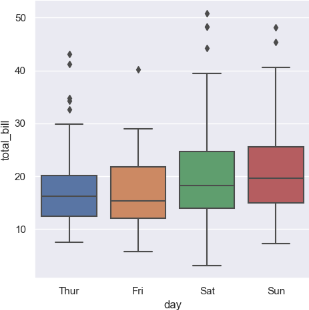
## Categorical Plot

sns.catpot(x='day', y='total\_bill', data=tips)



## Box Plot

sns.catplot(x='day', y='total\_bill', kind='box', data=tip)



# Result:

The experiment aimed to acquire a comprehensive understanding of data visualization fundamentals and extracting insights using Matplotlib and Seaborn.

# Experiment 3: Building a Data dashboard using Google Looker studio

**Aim:**To build a data dashboard in Google Looker Studio

## PROCEDURE/STEPS:

Step I : Open google chrome

Step 2: Sign in to the google account Step 3: Go to looker studio.

Step 4: Click on " + Create" option

Step 5: In add data to report pop up window Click on File Upload. Step 6: Click on "Authorize" button if prompted.

Step 7: Click on "Click to Upload File"

Step 8: Select a CSV or Excel worksheet which has the data.

Step 9: Once the file get uploaded, Click on Add button at the bottom. Step 10: Click on "Add to report" if prompted in pop up window.

Step 11 : Click on Add chart option.

step 12: From the list, select Table to insert table.

Step 13: Check for the setup tab in the right side panel.

Step 14: In dimension and parameter select the required data. Step 15: Click on Bar chart to insert Bar graph.

step 16: Similarly in the right side panel select the parameters in the Setup. step 17: Click on Pie chart.

Step 18: Select the parameters for Pie chart. Step 19: Click on tree map chart.

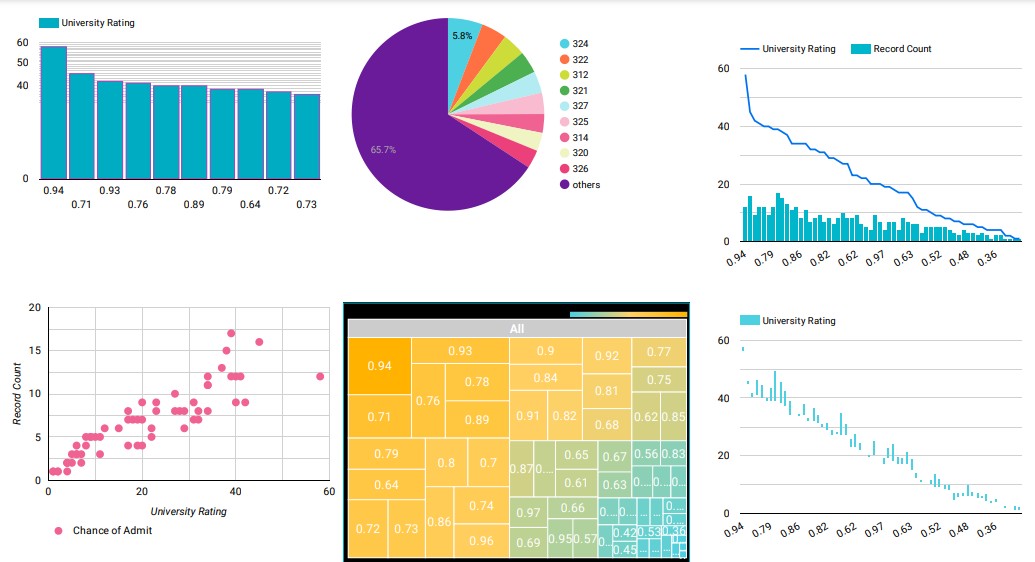
Step 20: Select the parameters for tree chart. Step21: Click on scatter chart.

step 22: Select the parameters for scatter chart. step 23: Click on Line chart.

step 24: Select the parameters for Line chart. Step 25: Click on File menu

Step 26: Click on "download as" step 27: Select PDF format

## OUTPUT:



**RESULT:**

The dashboard with different graphs are create using google looker studio.

# Experiment 4:Data Preprocessing & Feature Scaling in Python

**Aim:**To Clean data and perform feature scaling

## Algorithm:

1. Import required libraries & Data
2. Remove Missing values
3. Handle Categorical Data
4. Feature Scaling

## Importing libraries

import numpy as np import pandas as pd

## Importing data

dataset = pd.read\_csv('Data.csv') dataset

## Output:

**Handling Missing data Reshaping dataset to dataframe**

x **=** dataset**.**iloc[:,:**-**1]**.**values y **=** dataset**.**iloc[:,**-**1]**.**values x

**Finding Null Element:** dataset**.**isnull()**.**sum() Country 0

Age 3

Salary 3

Purchased 0

dtype: int64

## Importing Imputer Function

**from** sklearn.impute **import** SimpleImputer

## Applying simple imputer

imputer **=** SimpleImputer(missing\_values**=** np**.**nan, strategy**=**'mean')

**Filtering imputer**

imputer**.**fit(x[:,1:3])

x[:,1:3] **=** imputer**.**transform(x[:,1:3])

## Printing filled values

print(x)

[['France' 44.0 72000.0]

['Spain' 39.142857142857146 48000.0]

['Germany' 30.0 54000.0]

['Spain' 38.0 63714.28571428572]

['Germany' 40.0 63714.28571428572]

['France' 35.0 58000.0]

['Spain' 39.142857142857146 52000.0]

['France' 39.142857142857146 79000.0]

['Germany' 50.0 83000.0]

['France' 37.0 63714.28571428572]]

# Handling categorical Data

## Importing Libraries

**import** pandas **as** pd

**from** sklearn.preprocessing **import** LabelEncoder

salary\_class **=**

pd**.**DataFrame({'Salary':[5000,84000,22000,8000,75000],'Class':

['Low','High','Medium','Low','High']}) salary\_class



## Applying lab encoder

lab\_encode **=** LabelEncoder() salary\_class['Class'] **=**

lab\_encode**.**fit\_transform(salary\_class['Class']) salary\_class

**Feature Scaling**

**import** pandas **as** pd

**import** numpy **as** np

stand\_scaler**=** pd**.**DataFrame({'[x1':np**.**random**.**normal(0,2,100),

'x2':np**.**random**.**normal(3,5,100),

'x3':np**.**random**.**normal(**-**2,2,100)})

**# Instantiate MinMaxScaler**

scaler = MinMaxScaler()

**# Fit the scaler to the data**

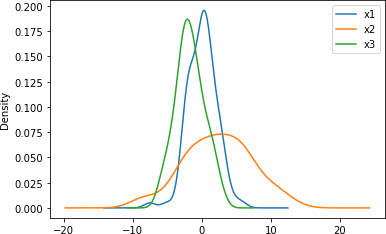
scaler.fit(stand\_scaler)

**# Transform the data**

scaled\_data = scaler.transform(stand\_scaler)

**# Convert the scaled data back to a DataFrame** scaled\_df = pd.DataFrame(scaled\_data, columns=stand\_scaler.columns)

print(scaled\_df.head())

**Plot Density Plot** stand\_scaler.plot.kde() **OUTPUT:**

## Result:

Successfully executed data preprocessing tasks and applied feature scaling techniques to enhance data quality and prepare it for machine learning models.

# Experiment 5: Working with Descriptive Statistics using SciPy

**Aim:**To perform statistical analysis on data SciPy library

## Algorithm

* 1. Importing the necessary library for descriptive statistics
  2. Load the dataset we want to calculate descriptive statistics
  3. Calculate the descriptive statistics parameters using scipy:

## Program :

**Import Libraries**

import numpy as np import pandas as pd

import matplotlib .pyplot as pit

%matplotlib inline

## Import & View the data

mtcars pd. read csv ("mtcars .csv") mtcars

mtcars = mtcars. rename (columns= { ' Unnamed: 0’ : ‘model’})

mtcars

## Remove unnecessary data

del mtcars ("model") mtcars . head ( )

## Measure of Central Tendency Mean

mtcars . mean ( ) mtcars. mean (axis=1)

## Median

mtcars .median () mtcars .median (axis=1)

## Mode

mtcars . mode ( )

## Measure of Spread Range

max(mtcars['mpg'])**-**min(mtcars['mpg']) 23.5

## Variance

mtcars["mpg"]**.**var() 36.32410282258064

## Standard Deviation

mtcars["mpg"].std () 6.026948052089104

## Measure of Shape

**Skewness**

mtcars ["mpg”].skew()

0.6723771376290805

## Kurtosis

mtcars["mpg"].kurt()

-0.0220062914240855

|  |  |  |  |
| --- | --- | --- | --- |
| **Output:** |  |  |  |
| **Mean:** |  |  |
|  |  | **Median** |
| mpg | 20.090625 |  |
| cyl | 6.187500 | mpg | 19.200 |
| disp | 230.721875 | cyl | 6.000 |
| hp | 146.687500 | disp | 196.300 |
| drat | 3.596563 | hp | 123.000 |
| wt | 3.217250 | drat | 3.695 |
| qsec | 17.848750 | wt | 3.325 |
| vs | 0.437500 | qsec | 17.710 |
| am | 0.406250 | vs | 0.000 |
| gear | 3.687500 | am | 0.000 |
| carb | 2.812500 | gear | 4.000 |
| dtype: | float64 | carb | 2.000 |
|  |  | dtype: | float64 |

## Result:

The experiment successfully executed statistical analysis on the dataset using SciPy, providing essential descriptive statistics parameters for further analysis.

# Experiment: 6 Inferential Statistics and Hypothesis Test

## Aim:

To Show the data Inferential Statistics and Hypothesis test using scipy

## Algorithm:

1. Import Libraries
2. Point Interval Estimation:
3. Confidence Interval
4. Hypothesis Test
5. One-Tailed T-Test
6. Two-Tailed T-Test

## Program:

**Point Interval Estimation**

**import**numpy**as**np **from**scipy.stats**import**t

**def**mean\_confidence\_interval(data,confidence**=**0.80): n**=**len(data)

mean**=**np**.**mean(data) std\_err**=**np**.**std(data,ddof**=**1)**/**np**.**sqrt(n) margin\_of\_error**=**std\_err**\***t**.**ppf((1**+**confidence)**/**2,n**-**1) lower\_bound**=**mean**-**margin\_of\_error upper\_bound**=**mean**+**margin\_of\_error **return**mean,lower\_bound,upper\_bound

data**=**[1,2,4,8,12,13,22,33,42,52]

confidence\_level**=**0.80 mean,lower\_bound,upper\_bound**=**mean\_confidence\_interval(data, confidence\_level)

print(f"Mean: {mean}")

print(f"Confidence Interval ({int(confidence\_level**\***100)}%): [{lower\_bound}, {upper\_bound}]")

## Output:

Mean: 18.9

Confidence Interval (80%): [11.094221522552104, 26.7057784774478

93]

## Confidence Intervel

**def**compare\_means\_and\_confidence\_interval(data1,data2,confidence**=**

0.80):

n1**=**len(data1) n2**=**len(data2) mean1**=**np**.**mean(data1) mean2**=**np**.**mean(data2) std1**=**np**.**std(data1,ddof**=**1) std2**=**np**.**std(data2,ddof**=**1)

std\_err**=**np**.**sqrt((std1**\*\***2**/**n1)**+**(std2**\*\***2**/**n2)) t\_critical**=**t**.**ppf((1**+**confidence)**/**2,n1**+**n2**-**2)

### # Hypothesis testing

t\_statistic**=**(mean1**-**mean2)**/**std\_err reject\_null**=**np**.**abs(t\_statistic)**>**t\_critical

***# Confidence interval calculation*** mean\_diff**=**mean1**-**mean2 margin\_of\_error**=**t\_critical**\***std\_err lower\_bound**=**mean\_diff**-**margin\_of\_error upper\_bound**=**mean\_diff**+**margin\_of\_error

**return**reject\_null,(lower\_bound,upper\_bound)

### # Example usage:

data1**=**[35,45,50,65,75] data2**=**[25,32,44,55,66]

confidence\_level**=**0.80 reject\_null,confidence\_interval**=**compare\_means\_and\_confidence\_int erval(data1,data2,confidence\_level)

**if**reject\_null:

print("Null hypothesis rejected: There is a significant difference between the means.")

**else**:

print("Null hypothesis not rejected: There is no significant difference between the means.")

print(f"Confidence Interval ({int(confidence\_level**\***100)}%):

{confidence\_interval}")

## Output:

Null hypothesis not rejected: There is no significant difference between the means.

Confidence Interval (80%): (-4.812264284107039, 24.0122642841070

4)

## Hypothesis Test

def student\_t\_test(sample1, sample2, a=0.07): n1 = len(sample1)

n2 = len(sample2)

mean1 = np.mean(sample1) mean2 = np.mean(sample2)

std1 = np.std(sample1, ddof=1) std2 = np.std(sample2, ddof=1)

pooled\_std = np.sqrt((std1\*\*2 / n1) + (std2\*\*2 / n2)) t\_statistic = (mean1 - mean2) / pooled\_std

degrees\_of\_freedom = n1 + n2 - 2

p\_value = 2 \* (1 - t.cdf(abs(t\_statistic), df=degrees\_of\_freedom)) reject\_null = p\_value< a

return reject\_null, t\_statistic, p\_value sample1 = [22, 35, 20, 72, 77]

sample2 = [44, 27, 45, 10, 34]

a= 0.07

reject\_null, t\_statistic, p\_value = student\_t\_test(sample1, sample2, a) if reject\_null:

print("Reject the null hypothesis: There is a significant difference between the means.") else:

print("Fail to reject the null hypothesis: There is no significant difference between the means.")

print(f"t-statistic: {t\_statistic}") print(f"p-value: {p\_value}")

## Output:

Fail to reject the null hypothesis: There is no significant diff erence between the means.

t-statistic: 0.9535222907320946

p-value: 0.368244058919005

# One Tailed T-Test

**def**one\_tailed\_t\_test(sample,null\_mean,alternative**=**'Greater',a**=**0.07):

n**=**len(sample) sample\_mean**=**np**.**mean(sample) sample\_std**=**np**.**std(sample,ddof**=**1)

t\_statistic**=**(sample\_mean**-**null\_mean)**/**(sample\_std**/**np**.**sqrt(n))

**if**alternative**==**'Greater':

p\_value**=**1**-**t**.**cdf(t\_statistic,df**=**n**-**1) reject\_null**=**p\_value**<**a **elif**alternative**==**'Less':

p\_value**=**t**.**cdf(t\_statistic,df**=**n**-**1) reject\_null**=**p\_value**<**a

**else**:

**raise**ValueError("Invalid alternative hypothesis. Choose either 'greater' or 'less'.")

**return**reject\_null,t\_statistic,p\_value

sample**=**[10,11,18,21,24,17,47,33]

null\_mean**=**12 alternative**=**'Greater' a**=**0.07

reject\_null,t\_statistic,p\_value**=**one\_tailed\_t\_test(sample,null\_mean,alternative,a)

**if**reject\_null:

print("Reject the null hypothesis: The sample mean is significantly greater than the null mean")

**else**:

print("Fail to reject the null hypothesis: The sample mean is not significantly greater than the null mean")

print(f"t-statistic: {t\_statistic}") print(f"p-value: {p\_value}")

## Output:

Reject the null hypothesis: The sample mean is significantly gre ater than the null mean

t-statistic: 2.449223405732373

p-value: 0.02207876149853427

# Two Tailed T-Test

**def**two\_tailed\_t\_test(sample,null\_mean,a**=**0.07):

n**=**len(sample) sample\_mean**=**np**.**mean(sample) sample\_std**=**np**.**std(sample,ddof**=**1)

t\_statistic**=**(sample\_mean**-**null\_mean)**/**(sample\_std**/**np**.**sqrt(n)) degrees\_of\_freedom**=**n**-**1

p\_value**=**2**\***(1**-**t**.**cdf(abs(t\_statistic),df**=**degrees\_of\_freedom)) reject\_null**=**p\_value**<**a

**return**reject\_null,t\_statistic,p\_value sample**=**[10,14,15,20,22,17,45,21]

null\_mean**=**12 a**=**0.07

reject\_null,t\_statistic,p\_value**=**two\_tailed\_t\_test(sample,null\_me an,a)

**if**reject\_null:

print("Reject the null hypothesis: The sample mean is significantly different from the null mean.")

**else**:

print("Fail to reject the null hypothesis: The sample mean is not significantly different from the null mean.")

print(f"t-statistic: {t\_statistic}") print(f"p-value: {p\_value}")

## Output:

Reject the null hypothesis: The sample mean is significantly dif ferent from the null mean.

t-statistic: 2.2517050070105746

p-value: 0.05904942200368035

## Result:

The experiment successfully showcased the application of inferential statistics and hypothesis testing techniques using the SciPy library.

# Experiment 7: Building a Simple Linear Regression Model using Scikit Learn

**Aim:**To build a simple linear regression model using Scikit learn library

## Algorithm:

1. Import all the required python libraries
2. Import Dataset
3. View the dataset
4. Remove unnecessary columns
5. Reshape the dataset
6. Divide dataset into training set and testing set
7. Import linear regression class
8. Create an object of the linear regression class
9. Fitting the data
10. Predicting the output

## Program:

import warnings

warnings . simplefilter (" ignore") import numpy as np

import pandas as pd

dataset pd. read csv ("Admission\_predict\_Verl . 1. csv") dataset

dataset**=**dataset**.**drop(['Serial No.','TOEFLScore','UniversityRating','SOP','LOR ','CGPA','Research'],axis**=**1)

dataset

x=dataset.iloc[:,0].values.reshape(-1,1) y=dataset.iloc[:,1].values.reshape(-1,1)

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

x\_train, x\_test, y\_train, y\_test= train test split (x, y, test size=0.2, random state=0)

from sklearn . linear model import LinearRegression lm = LinearRegression ( )

lm. fit (x train, y \_ train) y\_pred=lm.predict (x\_test)

## Output:

array ( [ [0.73841192),[0.76942347),[0.76942347], [0.84178376]

[0.56267979],

[0.69706318],

[0.53166823],

[0.57301697])

**Note :** This is the sample output. The output we displayed is the predicted probability ofgetting admission. Students are expected to compare the actual test set output with the predicted output to appreciate prediction model

## Result:

The experiment achieved its aim by successfully constructing a simple linear regression model using the Scikit-learn library.

# Experiment 8: Building a Multiple Linear Regression Model using Scikit Learn

**Aim:** To build a Multiple linear regression model using Scikit library

## Algorithm:

1. Import all the required python libraries
2. Import Dataset
3. View the dataset
4. Remove unnecessary columns
5. Reshape the dataset
6. Divide dataset into training set and testing set
7. Import linear regression class
8. Create an object of the linear regression class
9. Fitting the data
10. Predicting the output

## Program:

**import**warnings warnings**.**simplefilter('ignore') import numpy as np

import pandas as pd dataset=pd.read\_csv("Admission\_Predict.csv") dataset

dataset = dataset.drop(['Serial No.'],axis=1)

x = dataset.iloc[:,:-2].values.reshape(-1,1) y = dataset.iloc[:,-1].values

from sklearn.model\_selection import train\_test\_split x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size = 0.2,random\_state=0)

from sklearn.linear\_model import LinearRegression lm = LinearRegression()

lm.fit(x\_train,y\_train) y\_predict = lm.predict(x\_test)

## Output:

Array([[0.73841192], [0.76942347],

[0.76942347],

[0.84178376],

[0.56267979],

[0.69706318] ,

[0.53166823],

[0.57301697]])

**Note :** This is the sample output. The output we displayed is the predicted probability of getting admission. Students are expected to compare the actual test set output with the predicted output to appreciate prediction model

## Result:

Successfully implemented a multiple linear regression model using Scikit-learn, allowing for analysis of the relationship between multiple independent variables and a dependent variable

# Experiment 9: Building a Logistic Regression Model in Scikit Learn

**Aim:**To build a Logistic regression model using Scikit learn Library

## Algorithm:

1. Import libraries
2. Import Data
3. Perform Exploratory Data Analysis
4. Identify dependent and independent data
5. Divide Dataset into training and test set
6. Fit the model
7. Perform Prediction using Test set

## Program:

### #Import libraries

importnumpyasnp importpandasaspd from sklearn.preprocessing import matplotlib .pyplot as plt

from sklearn.linear\_model import LogisticRegression importseabornassns

%matplotlib inline

### #Import data

data—pd. read\_csv ( diabetes.csv )

### #Exploratory Data Analysis

data. shape data. Columns data info()

data [‘Outcome’].value\_counts() data.corr(method=’spearman’)

### #Identifying dependent and independent data

feature\_cols = [‘Pregnancies’, ‘Glucose’, ‘Blood pressure’, ‘SkinThickness’, ‘Insulin’, ‘BMI’, ‘DiabetesPedigreeFunction’, ‘Age’]

X data [feature\_cols] y data . Outcome

### # Dividing the dataset into training set and testing set

from skiearn. model selection import train\_test\_split X\_train,X\_test,y\_train,y\_test= train\_test\_split(X,y,test\_size=0.25,random\_state=30)

### # Fitting the model

from sklearn.linear\_model import LogisticRegression model LogisticRegression ()

model model.fit(X\_train,y\_train)

***# Perform Prediction using test set*** Y\_predmodel.preduct(X\_test) Y\_pred

## Output:

array ( [0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0..])= int64

# Result:

The experiment was successful in building a logistic regression model using the Scikit-learn library.

# Experiment 10: Building an Image recognition model using SVM and PCA

**Aim**:To build an Image recognition model using SVM and PCA

## Algorithm:

1. Import required libraries
2. Assign directories for dataset
3. Read Images
4. View the Output images
5. Convert Images to gray scale image
6. Resize the images
7. Flatten the images
8. Stack the images
9. Convert the dataset into Data frame
10. Add label to the flatten images
11. Perform the same for other set Of images
12. Merge all the three sets
13. Save the file
14. Identify the dependent and independent data
15. Divide the dataset into training set and testing set
16. Import PCA model
17. Fit the PCA model with independent data
18. Extract Eigen components
19. Fit data into support vector machines model
20. Predict on new images
21. Visualize the images

## Program:

### #Import required libraries

**import** os

**import** warnings warnings**.**simplefilter('ignore')

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

**%matplotlib** inline

**from** skimage.io **import**imread, imshow **from**skimage.transform**import** resize **from**skimage.color**import** rgb2gray

leo**=**os**.**listdir("C:\\Users\\Admin\\Documents\\VIKKI 4TH\\Cropped DS\\leo")

maldini**=**os**.**listdir("C:\\Users\\Admin\\Documents\\VIKKI 4TH\\Cropped DS\\maldini") david**=**os**.**listdir("C:\\Users\\Admin\\Documents\\VIKKI 4TH\\Cropped DS\\david")

limit=10 leo\_img=[**None**]\*limit j=0

**for**i in leo:

if(j<limit):

leo\_img[j]=imread("C:\\Users\\Admin\\Documents\\VIKKI 4TH\\Cropped DS\\leo\\"+i)

j**+**=1

## else:

**break**

limit**=**10 maldini\_img**=**[**None**]**\***limit j**=**0

**for**i**in**maldini:

**if**(j**<**limit): maldini\_img[j]**=**imread("C:\\Users\\Admin\\Documents\\VIKKI 4TH\\Cropped DS\\maldini\\"**+**i)

j**+=**1

## else:

**break**

limit**=**10 david\_img**=**[**None**]**\***limit j**=**0

**for**i**in**david:

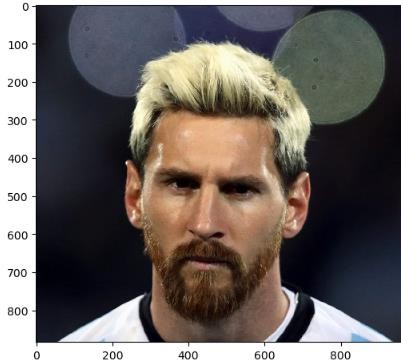
**if**(j**<**limit): david\_img[j]**=**imread("C:\\Users\\Admin\\Documents\\VIKKI 4TH\\Cropped DS\\david\\"**+**i)

j**+=**1

## else:

**break**

imshow(leo\_img[0])



**B-3 GRAY RGB2**

leo\_gray**=**[**None**]**\***limit j**=**0

**for**i**in**leo: **if**(j**<**limit):

leo\_gray[j]**=**rgb2gray(leo\_img[j]) j**+=**1

## else:

**break**

david\_gray**=**[**None**]**\***limit j**=**0

**for**i**in**david: **if**(j**<**limit):

david\_gray[j]**=**rgb2gray(david\_img[j]) j**+=**1

## else:

**break**

maldini\_gray**=**[**None**]**\***limit j**=**0

**for**i**in**maldini:

**if**(j**<**limit): maldini\_gray[j]**=**rgb2gray(maldini\_img[j][:,:,:3])

j**+=**1

## else:

**break**

**for** j **in** range (10): lm**=**leo\_gray[j] leo\_gray[j]**=**resize(lm,(512,512))

**for** j **in** range (10): pm**=**maldini\_gray[j]

maldini\_gray[j]**=**resize(pm,(512,512))

**for** j **in** range (10):

db**=**david\_gray[j] david\_gray[j]**=**resize(db,(512,512))

leo\_gray[0]**.**shape out[18]:(512, 512)

## Find out the number of gray\_scale img For Leo

len\_of\_img\_leo**=**len(leo\_gray) len\_of\_img\_leo*#output:10*

img\_size\_leo**=**leo\_gray[1]**.**shape img\_size\_leo #*output:(512, 512)*

## Flatten Size

flatten\_size\_leo**=**img\_size\_leo[0]**\***img\_size\_leo[1] flatten\_size\_leo*#output:262144*

**for**i**in** range(len\_of\_img\_leo): leo\_gray[i]**=**np**.**ndarray**.**flatten(leo\_gray[i])**.**reshape(flatten\_size

\_leo,1)

np**.**ndarray**.**flatten

leo\_gray**=**np**.**dstack(leo\_gray) leo\_gray

leo\_gray**.**shape (262144, 1, 10)

leo\_gray**=**np**.**rollaxis(leo\_gray,axis**=**2,start**=**0) leo\_gray**.**shape

(10, 262144, 1)

leo\_data**=**pd**.**DataFrame(leo\_gray) leo\_data

leo\_data["Label"]**=**"leo" leo\_data

**For Maldini**

len\_of\_img\_maldini**=**len(maldini\_gray) len\_of\_img\_maldini*#Output: 10*

img\_size\_maldini**=**maldini\_gray[1]**.**shape img\_size\_maldini

flatten\_size\_maldini**=**img\_size\_maldini[0]**\***img\_size\_maldini[1] flatten\_size\_maldini

**for**i**in**range(len\_of\_img\_maldini): maldini\_gray[i]**=**np**.**ndarray**.**flatten(maldini\_gray[i])**.**reshape(flat ten\_size\_maldini,1)

np**.**ndarray**.**flatten

maldini\_gray**=**np**.**dstack(maldini\_gray) maldini\_gray

maldini\_gray**.**shape (262144, 1, 10)

maldini\_gray**=**np**.**rollaxis(maldini\_gray,axis**=**2,start**=**0) maldini\_gray**.**shape

(10, 262144, 1)

maldini\_gray**=**maldini\_gray**.**reshape(len\_of\_img\_maldini,flatten\_siz e\_maldini)

maldini\_gray**.**shape

maldini\_data**=**pd**.**DataFrame(maldini\_gray) maldini\_data

maldini\_data["Label"]**=**"maldini" maldini\_data

## For David

len\_of\_img\_david**=**len(david\_gray) len\_of\_img\_david*#Output: 10*

img\_size\_david**=**david\_gray[1]**.**shape img\_size\_david

flatten\_size\_david**=**img\_size\_david[0]**\***img\_size\_david[1] flatten\_size\_david

**for**i**in**range(len\_of\_img\_david): david\_gray[i]**=**np**.**ndarray**.**flatten(david\_gray[i])**.**reshape(flatten\_ size\_david,1)

np**.**ndarray**.**flatten

david\_gray**=**np**.**dstack(david\_gray) david\_gray

david\_gray**.**shape (262144, 1, 10)

david\_gray**=**np**.**rollaxis(david\_gray,axis**=**2,start**=**0) david\_gray**.**shape

(10, 262144, 1)

david\_gray**=**david\_gray**.**reshape(len\_of\_img\_david,flatten\_size\_davi d)

david\_gray**.**shape

david\_data**=**pd**.**DataFrame(david\_gray) david\_data

david\_data["Label"]**=**"david" david\_data

## Merge Images

man\_1**=**pd**.**concat([leo\_data,maldini\_data]) man**=**pd**.**concat([man\_1,david\_data])

man

## Shuffling

**from**sklearn.utils**import**shuffle fb\_indexed**=**shuffle(man)**.**reset\_index() fb\_indexed fb\_man**=**fb\_indexed**.**drop(['index'],axis**=**1)

fb\_man**.**to\_csv("Players.csv")

x **=**fb\_man**.**values[:,:**-**1] y **=**fb\_man**.**values[:,**-**1]

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

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x,y,test\_size**=**0.2

,random\_state**=**0) x\_train**.**shape

(24, 262144)

x\_test**.**shape (6, 262144)

## Decomposition

**from**sklearn**import** decomposition

pca**=**decomposition**.**PCA(n\_components**=**20, whiten**=True**, random\_state**=**1)

## Fitting Training Set

pca**.**fit(x\_train)

## Change Train set

x\_train\_pca**=**pca**.**transform(x\_train) x\_test\_pca**=**pca**.**transform(x\_test) x\_train\_pca**.**shape

(24, 20)

x\_test\_pca**.**shape (6, 20)

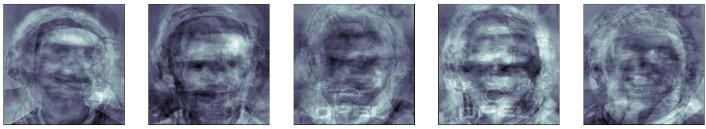
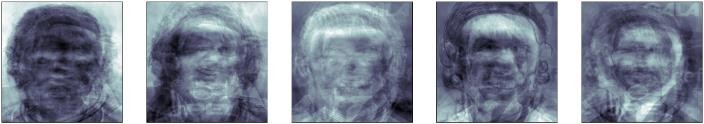
## Viewing the Principle components or eigen faces

eigen**=**(np**.**reshape(x[10],(512,512))**.**astype(np**.**float64)) eigen

## Plotting

fig**=**plt**.**figure(figsize**=**(30,30)) **for**i**in**range(10): ax**=**fig**.**add\_subplot(2,5,i**+**1,xticks**=**[],yticks**=**[])

ax**.**imshow(pca**.**components\_[i]**.**reshape(eigen**.**shape),cmap**=**plt**.**cm**.** bone)



## Support vector Machine Implementation

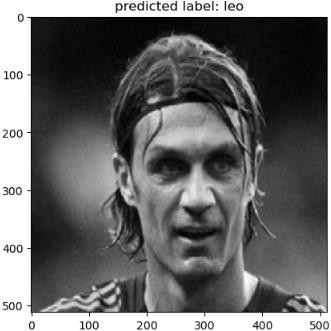
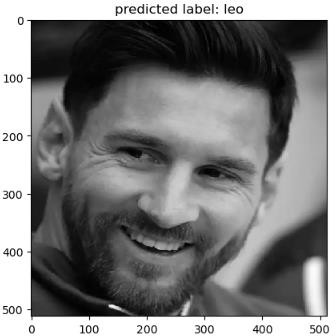
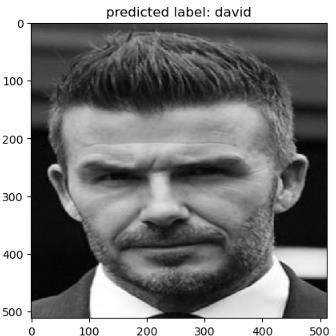
**from**sklearn**import**svm clf**=**svm**.**SVC(C**=**2,gamma**=**0.006,kernel**=**'rbf') clf**.**fit(x\_train\_pca,y\_train)

# Image Prediction

y\_predict**=**clf**.**predict(x\_test\_pca) y\_predict

**for in**(np**.**random**.**randint(0,6,6)): predicted\_images**=**(np**.**reshape(x\_test[i],(512,512))**.**astype (np**.**float64))

plt**.**title('predicted label: {0}'**.**format(y\_predict[i])) plt**.**imshow(predicted\_images,interpolation**=**'nearest',cmap**=**'gray') plt**.**show()



**from**sklearn**import**metrics accuracy**=**metrics**.**accuracy\_score(y\_test,y\_predict) accuracy

Output:

# Result:

The experiment successfully resulted in an image recognition model using SVM and PCA, demonstrating accurate classification of images.

# Experiment 11: Building an Emoji Classification model using SVM and PCA

**Aim**:

To build an Emojis Classification using SVM and PCA

## Algorithm:

* + 1. Import required libraries
    2. Assign directories for dataset
    3. Read Images
    4. View the Output images
    5. Convert Images to gray scale image
    6. Resize the images
    7. Flatten the images
    8. Stack the images
    9. Convert the dataset into Data frame
    10. Add label to the flatten images
    11. Perform the same for other set of images
    12. Merge all the three sets
    13. Save the file
    14. Identify the dependent and independent data
    15. Divide the dataset into training set and testing set
    16. Import PCA model
    17. Fit the PCA model with independent data
    18. Extract Eigen components
    19. Fit data into support vector machines model
    20. Predict on new images
    21. Visualize the images

## Program:

**import** os

**import** warnings warnings**.**simplefilter('ignore')

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

**%matplotlib** inline

**from** skimage.io **import**imread, imshow **from**skimage.transform**import** resize **from**skimage.color**import** rgb2gray

smile**=**os**.**listdir("C:\\Users\\Vikneshraj\\Documents\\IV SEMESTER\\ML\\Data Emojis\\Smile")

anger**=**os**.**listdir("C:\\Users\\Vikneshraj\\Documents\\IV SEMESTER\\ML\\Data Emojis\\Anger")

sad**=**os**.**listdir("C:\\Users\\Vikneshraj\\Documents\\IV SEMESTER\\ML\\Data Emojis\\Sad")

limit**=**10 smile\_img**=**[**None**]**\***limit j**=**0

**for**i**in**smile:

**if**(j**<**limit): smile\_img[j]**=**imread("C:\\Users\\Vikneshraj\\Documents\\IV SEMESTER\\ML\\Data Emojis\\Smile\\"**+**i)

j**+=**1

**else**:

**break**

limit**=**10 anger\_img**=**[**None**]**\***limit j**=**0

**for**i**in**anger:

**if**(j**<**limit): anger\_img[j]**=**imread("C:\\Users\\Vikneshraj\\Documents\\IV SEMESTER\\ML\\Data Emojis\\Anger\\"**+**i)

j**+=**1

**else**:

**break**

limit**=**10 sad\_img**=**[**None**]**\***limit j**=**0

**for**i**in**sad:

**if**(j**<**limit): sad\_img[j]**=**imread("C:\\Users\\Vikneshraj\\Documents\\IV SEMESTER\\ML\\Data Emojis\\Sad\\"**+**i)

j**+=**1

**else**:

**break**

imshow(smile\_img[0])

## Gray RGB2

smile\_gray**=**[**None**]**\***limit j**=**0

**for**i**in**smile:

**if**(j**<**limit): smile\_gray[j]**=**rgb2gray(smile\_img[j][:,:,:3]) j**+=**1

**else**:

**break**

anger\_gray**=**[**None**]**\***limit j**=**0

**for**i**in** anger:

**if**(j**<**limit): anger\_gray[j]**=**rgb2gray(anger\_img[j][:,:,:3])

j**+=**1

## else:

**break**

sad\_gray**=**[**None**]**\***limit j**=**0

**for**i**in** sad:

**if**(j**<**limit): sad\_gray[j]**=**rgb2gray(sad\_img[j][:,:,:3])

j**+=**1

## else:

**break**

**for** j **in** range (10): se**=**smile\_gray[j]

smile\_gray[j]**=**resize(se,(512,512))

**for** j **in** range (10): ae**=**anger\_gray[j]

anger\_gray[j]**=**resize(ae,(512,512))

**for** j **in** range (10): he**=**sad\_gray[j]

sad\_gray[j]**=**resize(he,(512,512))

smile\_gray[0]**.**shape anger\_gray[0]**.**shape

sad\_gray[0]**.**shape

## For Smil Emoji

len\_of\_img\_smile**=**len(smile\_gray) len\_of\_img\_smile

img\_size\_smile**=**smile\_gray[1]**.**shape img\_size\_smile

**Flatten** flatten\_size\_smile**=**img\_size\_smile[0]**\***img\_size\_smile[1] flatten\_size\_smile

**for**i**in** range(len\_of\_img\_smile): smile\_gray[i]**=**np**.**ndarray**.**flatten(smile\_gray[i])**.**reshape(flatten\_ size\_smile,1)

np**.**ndarray**.**flatten

smile\_gray**=**np**.**dstack(smile\_gray) smile\_gray

smile\_gray**.**shape smile\_gray**=**np**.**rollaxis(smile\_gray,axis**=**2,start**=**0) smile\_gray**.**shape

smile\_gray**=**smile\_gray**.**reshape(len\_of\_img\_smile,flatten\_size\_smil e)

smile\_gray**.**shape

smile\_data**=**pd**.**DataFrame(smile\_gray) smile\_data smile\_data["Label"]**=**"smile" smile\_data

## For Angry Emoji

len\_of\_img\_anger**=**len(anger\_gray) len\_of\_img\_anger

img\_size\_anger**=**anger\_gray[1]**.**shape img\_size\_anger

flatten\_size\_anger**=**img\_size\_anger[0]**\***img\_size\_anger[1] flatten\_size\_anger

**for**i**in**range(len\_of\_img\_anger):anger\_gray[i]**=**np**.**ndarray**.**flatten(a nger\_gray[i])**.**reshape(flatten\_size\_anger,1)

np**.**ndarray**.**flatten

anger\_gray**=**np**.**dstack(anger\_gray) anger\_gray

anger\_gray.shape

anger\_gray**=**np**.**rollaxis(anger\_gray,axis**=**2,start**=**0) anger\_gray**.**shape

anger\_gray**=**anger\_gray**.**reshape(len\_of\_img\_anger,flatten\_size\_ange r)

anger\_gray**.**shape

anger\_data**=**pd**.**DataFrame(anger\_gray) anger\_data

anger\_data["Label"]**=**"anger" anger\_data

**For Sad Emoji** len\_of\_img\_sad**=**len(sad\_gray) len\_of\_img\_sad

img\_size\_sad**=**sad\_gray[1]**.**shape img\_size\_sad

flatten\_size\_sad**=**img\_size\_sad[0]**\***img\_size\_sad[1] flatten\_size\_sad

**for**i**in**range(len\_of\_img\_sad): sad\_gray[i]**=**np**.**ndarray**.**flatten(sad\_gray[i])**.**reshape(flatten\_size

\_sad,1) np**.**ndarray**.**flatten

sad\_gray**=**np**.**dstack(sad\_gray)

sad\_gray sad\_gray**.**shape

sad\_gray**=**np**.**rollaxis(sad\_gray,axis**=**2,start**=**0) sad\_gray**.**shape

sad\_gray**=**sad\_gray**.**reshape(len\_of\_img\_sad,flatten\_size\_sad) sad\_gray**.**shape

sad\_data**=**pd**.**DataFrame(sad\_gray) sad\_data

sad\_data["Label"]**=**"sad" sad\_data

## Merge Images

a\_1**=**pd**.**concat([smile\_data,sad\_data]) a**=**pd**.**concat([a\_1,anger\_data])

a

## Shuffling

**from**sklearn.utils**import**shuffle fb\_indexed**=**shuffle(man)**.**reset\_index() fb\_indexed

fb\_man**=**fb\_indexed**.**drop(['index'],axis**=**1) fb\_man**.**to\_csv("Emojis.csv") x**=**fb\_man**.**values[:,:**-**1] y**=**fb\_man**.**values[:,**-**1]

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

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x,y,test\_size**=**0.2

,random\_state**=**1) x\_train**.**shape x\_test**.**shape

## Decomposition

**from**sklearn**import** decomposition

pca**=**decomposition**.**PCA(n\_components**=**10, whiten**=True**, random\_state**=**1)

## Fitting Training Set

pca**.**fit(x\_train)

## Change Training Set

x\_train\_pca**=**pca**.**transform(x\_train) x\_test\_pca**=**pca**.**transform(x\_test)

x\_train\_pca**.**shape x\_test\_pca**.**shape

## Viewing The Princeple Components or eigen

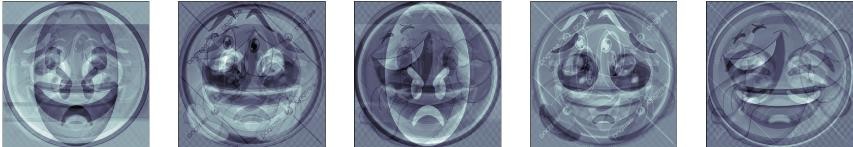
eigen **=** (np**.**reshape(x[10],(512,512))**.**astype(np**.**float64)) eigen

## Plotting

fig **=**plt**.**figure(figsize**=**(30,30)) **for**i**in** range(10):

ax **=**fig**.**add\_subplot(2, 5, i**+**1, xticks**=**[], yticks**=**[]) ax**.**imshow(pca**.**components\_[i]**.**reshape(eigen**.**shape),cmap**=**plt**.**cm**.**

bone) Output:



## Support Vector Machine

**From** sklearn.ensemble**import**RandomForestClassifier

clf**=**RandomForestClassifier(n\_estimators**=**100, random\_state**=**42) clf**.**fit(x\_train\_pca, y\_train)

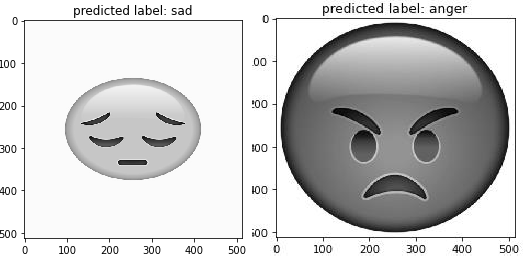
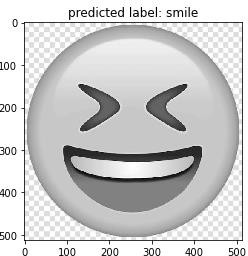
## Image Prediction

y\_predict**=**clf**.**predict(x\_test\_pca)

**for in** (np**.**random**.**randint(0,6,6)): predicted\_images**=** (np**.**reshape(x\_test[i], (512,512))**.**astype(np**.**float64))

plt**.**title('predicted label: {0}'**.** format(y\_predict[i])) plt**.**imshow(predicted\_images,interpolation**=** 'nearest',cmap**=** 'gray')

plt**.**show()



**from**sklearn**import**metrics accuracy**=**metrics**.**accuracy\_score(y\_test,y\_predict) accuracy

**OUTPUT:**

# Result:

The experiment successfully resulted in an image recognition model using SVM and PCA, demonstrating accurate classification of Emojis

**Experiment 12: Spam Detection method using Naïve Bayes Method Aim:**To build a Classification model to detect Spam using Naïve Bayes method **Algorithm:**

## Import required libraries

* + 1. **Import data**

## Exploratory Dato Analysis

* + 1. **Applying Count vectorizer**

## Identity dependent and independent data

* + 1. **Dividing cell for training and testing set**

## Import naive bayes classifier

* + 1. **Fit the data**

## Predict the output

* + 1. **Plot confusion matrix Program:**

### # Import required libraries

import pandas as pd import numpy as np import seaborn as sns

import matplotlib.pyplot as pit

### #Import data

email**=**pd**.**read\_csv('emails.csv') email

***# Data Analysis*** email.describe ( ) email.info ( )

spam0= email [email ['spam'] == 0] spam1 = email[email['spam'] == 1] sns.countplot (x = email( spam'), label

print('spam percentage =' ,(len(spam0) **/** len(email))**\***100, '%')

*#spam percentage = 76.11731843575419 %*

print('spam percentage =' ,(len(spam1) **/** len(email))**\***100, '%')

*#spam percentage = 23.88268156424581 %*

sns**.**countplot(x **=** email['spam'],label **=** 'spam vs spam0')

### # Apply Count Vectorizer

**from**sklearn.feature\_extraction.text**import**CountVectorizer vectorizer**=**CountVectorizer() spam1spam0\_countVectorizer**=**vectorizer**.**fit\_transform(email['text'

])

print(vectorizer**.**get\_feature\_names\_out()) spamlspam0\_countVectorizer.shape

### # Identify Dependent and Independent Data

label = email[‘spam’]

X = spam1spam0\_CountVectorizer Y = label

***# Dividing the data into training set and testing set* from**sklearn.model\_selection**import**train\_test\_split **from**sklearn.naive\_bayes**import**MultinomialNB x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x,y,test\_size**=**0.2

)

***#Fit the data*** NB\_classifier**=**MultinomialNB() NB\_classifier**.**fit(x\_train,y\_train)

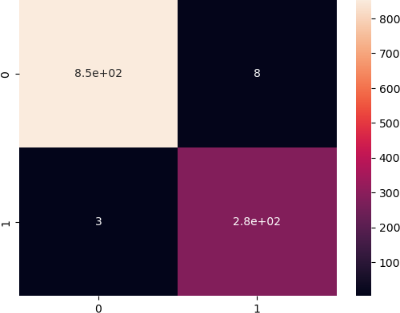
***#Predict the output*** y\_predict\_test**=**NB\_classifier**.**predict(x\_test) y\_predict\_test

### #Plot Confusion Matrix

cm**=**confusion\_matrix(y\_test,y\_predict\_test)

**Output:**

array([0, 0, 0, ..., 1, 0, 1], dtype=int64)



# Result:

The experiment successfully built a classification model for spam detection using the Naïve Bayes method, achieving reliable performance metrics

# Experiment 13: Building an Unsupervised Learning Model using Hierarchical Clustering

**Aim:**To build a Programthat would solveFibonacci series dynamic programming

## Algorithm:

1. Import Libraries
2. DataCleaning
3. Fitting the Model
4. Visualizing Clusters

## Program:

### #lmport libraries

import pandas as pd import numpy as np

import mat plotlib.pyplot as pit import matplotlib.pyplot as pit import scipy.cluster . hierarchy as sc from s k learn import datasets

from sklearn.cluster import AgglomerativeC1ustering

### Import dataset

iris**=**datasets**.**load\_iris() iris

#***Convert to Dataframe*** iris\_data**=**pd**.**DataFrame(iris**.**data) iris\_data

**#*Removing Label fromdataset*** iris\_X**=**iris\_data**.**iloc[:,[0,1,2,3]]**.**values iris\_Y**=**iris\_data**.**iloc[:,4]**.**values

### # Fit the Model

cluster AgglomerativeCiustering(n\_clusters= 3, affinity =

‘euclidean’,

linkage 'ward' ) cluster. fit (iris \_ X)

### # Print the labels

labels cluster. labels labels

### #Visuailze the classes

plt**.**figure(figsize**=**(10,7)) plt**.**scatter(iris\_X[iris\_Y**==**0,0],iris\_X[iris\_Y**==**0,1],s**=**100,c**=**'yel low',label**=**'set-1') plt**.**scatter(iris\_X[iris\_Y**==**1,0],iris\_X[iris\_Y**==**1,1],s**=**100,c**=**'lim e',label**=**'set-2') plt**.**scatter(iris\_X[iris\_Y**==**2,0],iris\_X[iris\_Y**==**2,1],s**=**100,c**=**'aqu a',label**=**'set-3')

plt**.**legend() plt**.**xlabel('Sample Index')

plt**.**ylabel('Euclidean Distance') plt**.**show()

## Output:

**Result:**

The program successfully implements dynamic programming to solve the Fibonacci series, demonstrating improved time complexity compared to naive recursive approaches.

# Experiment 14: Building an Recommender Systems in Python

**Aim:**To build a Recommender System to suggest movies

## Algorithm:

1. Import libraries
2. Import Data
3. Exploratory data Analysis
4. Framing Pivot Table
5. Displaying the sorted tables
6. Extracting desired movie ratings
7. Correlation
8. Viewing Recommendation

## Import data

**Import numpy as np Import Pandas as py**

**import**matplotlib.pyplot**as**plt **import**seaborn**as**sns sns**.**set\_style('white')

**%matplotlib** inline

column\_names**=**['user\_id','item\_id','rating','timestamp'] df**=**pd**.**read\_csv('u.data',sep**=**'\t',names**=**column\_names) df**.**head() movie\_titles**=**pd**.**read\_csv("Movie\_Id\_Titles.csv") movie\_titles**.**head() df**=**pd**.**merge(df,movie\_titles,on**=**'item\_id')

df**.**head()

***#Exploratory Data Analysis*** df**.**groupby('title')['rating']**.**mean()**.**sort\_values (ascending**=False**) df**.**groupby('title')['rating']**.**count()**.**sort\_values (ascending**=False**)

ratings **=**pd**.**DataFrame(df**.**groupby('title')['rating']**.**mean()) ratings

ratings['num of ratings']**=**pd**.**DataFrame(df**.**groupby('title')['rating']**.**count()) ratings.head()

***#Framing Pivot Table*** moviemat**=**df**.**pivot\_table(index**=**'user\_id',columns**=**'title',values**=**' rating')

moviemat

### #Display The sorted tables

ratings**.**sort\_values('num of ratings',ascending**=False**)**.**head(10) ratings

### #Extrating desired movie ratings

starwars\_user\_ratings**=**moviemat['Star Wars (1977)']

liarliar\_user\_ratings**=**moviemat['Liar Liar (1997)'] starwars\_user\_ratings**.**head(50)

***#Correlation*** similar\_to\_starwars**=**moviemat**.**corrwith(starwars\_user\_ratings) similar\_to\_liarliar**=**moviemat**.**corrwith(liarliar\_user\_ratings)

#***Viewing Recommendation*** corr\_starwars**=**pd**.**DataFrame(similar\_to\_starwars,columns**=**['correla tion'])

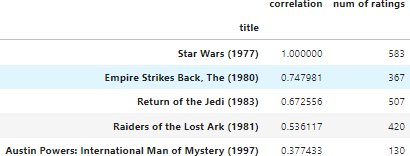
corr\_starwars**.**dropna(inplace**=True**) corr\_starwars**.**head()

corr\_starwars**.**sort\_values('correlation',ascending**=False**)**.**head(50

)

corr\_starwars**=**corr\_starwars**.**join(ratings['num of ratings']) corr\_starwars**.**head()

corr\_starwars[corr\_starwars['num of ratings']**>**100]**.**sort\_values('correlation',ascending**=False**)**.**head()

**Output:**

# Result:

Successfully developed a recommender system in Python that effectively suggests movies based on user preferences, enhancing user experience and engagement with the platform.

**Experiment 15: Implementation of Q- learning in Python Aim:**To build Q learning program to solve the cartpole problem using python **Algorithm:**

1. Initialize Environment:
2. Create the OpenAI Gym environment (CartPole – v1 in cage).
3. Initialize Q-Learning Agent:
4. Define a Q-Learning Agent class With methods for choosing actions and updating the Q-table based rewards.
5. The Q-learning agent has parameters such a learning rate (alpha), discount factor (gamma) and exploration rate (epsilon)
6. Initialize the Q-table with zeros
7. Training Loop:
8. For a specified number of episodes:
9. Reset the environment to the initial state.
10. Initialize the total reward for the episode to zero.
11. While the episode is not done:
12. Choose an action using epsilon-greedy policy (with exploration rate epsilon).
13. Take the chosen action and observe the next state and reward.
14. Update the Q-table using the Q-learning update equation.
15. Update the total reward for the episode,
16. Transition to the next state,
17. Print Progress:
18. Optionally, print the total reward obtained in each episode to track the agent's progress.
19. Close Environment:
20. Close the environment after training is completed.

# Program:

## def eps\_greedy(Q, s, eps=0.1):

ifnp.random.uniform(0,1) < eps: returnnp.random.randint(Q.shape[1]) else:

return greedy(Q, s)

defgreedy (Q, s):

return np.argmax(Q[s])

defrun\_episodes(env, Q, num\_episodes=100, to\_print=False):

tot\_rew = []

state = env.reset()

for \_ inrange(num\_episodes): done = False

game\_rew = 0

whilenot done:

next\_state, rew, done, \_ = env.step(greedy(Q, state))

state = next\_state game\_rew += rew

if done:

state = env.reset() tot\_rew.append(game\_rew)

ifto\_print:

print('Mean score: %.3f of %i games!' % (np.mean(tot\_rew), num\_episodes))

returnnp.mean(tot\_rew)

defQ\_learning(env, lr=0.01, num\_episodes=10000, eps=0.3, gamma=0.95, eps\_decay=0.00005):

nA = env.action\_space.n

nS = env.observation\_space.n

Q = np.zeros((nS, nA)) games\_reward = [] test\_rewards = []

for ep inrange(num\_episodes): state = env.reset()

tot\_rew = 0

if eps >0.01:

eps -= eps\_decay

done = False whilenot done:

action = eps\_greedy(Q, state, eps) next\_state, rew, done, \_ = env.step(action)

Q[state][action] = Q[state][action] + lr \* (rew + gamma \* np.max(Q[next\_state]) - Q[state][action])

state = next\_state tot\_rew += rew

if done:

games\_reward.append(tot\_rew)

if (ep % 300) == 0:

test\_rew = run\_episodes(env, Q, 1000)

print("Episode:{:5d} Eps:{:2.4f} Rew:{:2.4f}".format(ep, eps, test\_rew))

test\_rewards.append(test\_rew) return Q

if name == ' main ': env = gym.make('Taxi-v3')

print("Q-Learning")

Q\_learning = Q\_learning(env, lr=.1, num\_episodes= 5000, eps= 0.4

, gamma = 0.95, eps\_decay=0.001)

## Output

Q-Learning

|  |  |  |  |
| --- | --- | --- | --- |
| Episode: | 0 | Eps:0.3990 | Rew:-241.3190 |
| Episode: | 300 | Eps:0.0990 | Rew:-212.2510 |
| Episode: | 600 | Eps:0.0100 | Rew:-227.5580 |
| Episode: | 900 | Eps:0.0100 | Rew:-190.4110 |
| Episode: | 1200 | Eps:0.0100 | Rew:-119.1710 |
| Episode: | 1500 | Eps:0.0100 | Rew:-73.5610 |
| Episode: | 1800 | Eps:0.0100 | Rew:-54.2760 |
| Episode: | 2100 | Eps:0.0100 | Rew:-21.2480 |
| Episode: | 2400 | Eps:0.0100 | Rew:-5.1300 |
| Episode: | 2700 | Eps:0.0100 | Rew:0.1800 |
| Episode: | 3000 | Eps:0.0100 | Rew:4.6000 |
| Episode: | 3300 | Eps:0.0100 | Rew:2.9400 |
| Episode: | 3600 | Eps:0.0100 | Rew:7.8860 |
| Episode: | 3900 | Eps:0.0100 | Rew:7.9900 |
| Episode: | 4200 | Eps:0.0100 | Rew:7.8780 |
| Episode: | 4500 | Eps:0.0100 | Rew:7.7900 |
| Episode: | 4800 | Eps:0.0100 | Rew:7.9870 |

## Result:

The Q-learning program to solve the CartPole problem using Python has been successfully verified.