

VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS

Grade A++ Accredited Institution by NAAC

NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;
Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE

An ISO 9001:2015 Certified Institution

SCHOOL OF ENGINEERING & TECHNOLOGY

B. Tech Programme: AI-ML (A) (4th Semester)

**Course Title: Fundamentals of Machine
Learning Lab**

Course Code: AIML - 258

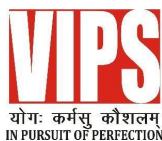
Submitted To:

Dr. Shalini Gambhir

Submitted By:

Name: Kunsh Sabharwal

Enrolment No: 01117711623



VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS

Grade A++ Accredited Institution by NAAC

NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;

Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE

An ISO 9001:2015 Certified Institution

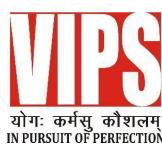
SCHOOL OF ENGINEERING & TECHNOLOGY

VISION OF INSTITUTE

To be an educational institute that empowers the field of engineering to build a sustainable future by providing quality education with innovative practices that supports people, planet and profit.

MISSION OF INSTITUTE

To groom the future engineers by providing value-based education and awakening students' curiosity, nurturing creativity and building capabilities to enable them to make significant contributions to the world.



VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS

Grade A++ Accredited Institution by NAAC

NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;
Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE

An ISO 9001:2015 Certified Institution

SCHOOL OF ENGINEERING & TECHNOLOGY

INDEX

S. N O.	Experiment	Date	Marks			Remarks	Updated Marks	Faculty Signature
			Laboratory Assessment (15 Marks)	Class Participation (5 Marks)	Viva (5 Marks)			
1.	Study & Implement Linear Regression							
2.	Study & Implement Logistic Regression							
3.	Study & Implement K-Nearest Neighbour (KNN)							
4.	Study & Implement classification using SVM							
5.	Study & Implement Bagging using Random Forests							
6.	Study & Implement Naïve-Bayes							
7.	Study & Implement Decision Trees							
8.	Study & Implement K- Means Clustering to find natural patterns in Data							
9.	Study & Implement Classification based on association rules							



VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS

Grade A++ Accredited Institution by NAAC

NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;
Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE

An ISO 9001:2015 Certified Institution

SCHOOL OF ENGINEERING & TECHNOLOGY

Sr. No.	EXPERIMENT TITLE (Beyond Curriculum)	Date	Faculty Signature
1.	Perform the Data Preprocessing life cycle on any sample data.		
2.	Project on Machine Learning		

EXPERIMENT 0

Problem statement: Perform the Data Preprocessing life cycle on any sample data.

Theory:

Dataset:

	A	B	C	D
1	Country	Age	Salary	Purchased
2	France	44	72000	No
3	Spain	27	48000	Yes
4	Germany	30	54000	No
5	Spain	38	61000	No
6	Germany	40		Yes
7	France	35	58000	Yes
8	Spain		52000	No
9	France	48	79000	Yes
10	Germany	50	83000	No
11	France	37	67000	Yes
12				

Source Code with Outputs:

DATA.CSV Dataset

```

[2] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

[3] df1=pd.read_csv("Data.csv")

[4] print(df1)

   Country  Age  Salary Purchased
0  France  44.0  72000.0      No
1  Spain   27.0  48000.0     Yes
2  Germany 30.0  54000.0      No
3  Spain   38.0  61000.0      No
4  Germany 40.0    NaN      Yes
5  France  35.0  58000.0     Yes
6  Spain   NaN   52000.0      No
7  France  48.0  79000.0     Yes
8  Germany 50.0  83000.0      No
9  France  37.0  67000.0     Yes

[5] x=df1.iloc[:, :-1].values
y=df1.iloc[:, -1].values

[6] from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer.fit(x[:, 1:3])
x[:, 1:3] = imputer.transform(x[:, 1:3])

```

```

[7] print(x)

[[['France' 44.0 72000.0],
 ['Spain' 27.0 48000.0],
 ['Germany' 30.0 54000.0],
 ['Spain' 38.0 61000.0],
 ['Germany' 40.0 63777.77777777778],
 ['France' 35.0 58000.0],
 ['Spain' 38.777777777777778 52000.0],
 ['France' 48.0 79000.0],
 ['Germany' 50.0 83000.0],
 ['France' 37.0 67000.0]]]

[8] print(y)

[No' 'Yes' 'No' 'No' 'Yes' 'Yes' 'No' 'Yes']

[9] from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder='passthrough')
x = np.array(ct.fit_transform(x))

[10] print(x)

[[1.0 0.0 0.0 44.0 72000.0]
 [0.0 0.0 1.0 27.0 48000.0]
 [0.0 1.0 0.0 30.0 54000.0]
 [0.0 0.0 1.0 38.0 61000.0]
 [0.0 1.0 0.0 40.0 63777.77777777778]
 [1.0 0.0 0.0 35.0 58000.0]
 [0.0 0.0 1.0 38.777777777777778 52000.0]
 [1.0 0.0 0.0 48.0 79000.0]
 [0.0 1.0 0.0 50.0 83000.0]
 [1.0 0.0 0.0 37.0 67000.0]]

```

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** Program_1_FML_Prac_File.ipynb
- Toolbar:** File, Edit, View, Insert, Runtime, Tools, Help, All changes saved.
- Code Cells:**
 - [11] from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
print(y)
Output: [0 1 0 0 1 1 0 1 0 1]
 - [12] 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)
 - [13] print(x_train)
Output: [[0.0 0.0 1.0 38.77777777777778 52000.0]
[0.0 1.0 0.0 40.0 63777.7777777778]
[1.0 0.0 0.0 44.0 72000.0]
[0.0 0.0 1.0 38.0 61000.0]
[0.0 0.0 1.0 27.0 48000.0]
[1.0 0.0 0.0 48.0 79000.0]
[0.0 1.0 0.0 50.0 83000.0]
[1.0 0.0 0.0 35.0 58000.0]]
 - [14] print(x_test)
Output: [[0.0 1.0 0.0 30.0 54000.0]
[1.0 0.0 0.0 37.0 67000.0]]
 - [15] print(y_train)
Output: [0 1 0 0 1 1 0 1]
 - [16] print(y_test)
Output: [0 1]
- Runtime Status:** 0s completed at 3:18PM

Learning Outcome:

EXPERIMENT 1

Problem statement: Study and implement Linear Regression.

Theory:

Dataset:

YearsExp	Salary
1.1	39343
1.3	46205
1.5	37731
2	43525
2.2	39891
2.9	56642
3	60150
3.2	54445
3.2	64445
3.7	57189
3.9	63218
4	55794
4	56957
4.1	57081
4.5	61111
4.9	67938
5.1	66029
5.3	83088
5.9	81363
6	93940
6.8	91738
7.1	98273
7.9	101302
8.2	113812
8.7	109431
9	105582
9.5	116969
9.6	112635
10.3	122391
10.5	121872

Source Code with Outputs:

```

CO Program1_FML_Prac_File.ipynb ⚡ Saving...
File Edit View Insert Runtime Tools Help
+ Code + Text
[1] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
[2] df=pd.read_csv("Salary_Data (1).csv")
[3] print(df.head())
      YearsExperience Salary
0            1.1    39343.0
1            1.3    46205.0
2            1.5    37731.0
3            2.0    43525.0
4            2.2    39891.0
[4] x=df.iloc[:, :-1].values
y=df.iloc[:, -1].values
[5] 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)
[6] print(x_train)

```

✓ 0s completed at 8:47PM

```

CO Program1_FML_Prac_File.ipynb ⚡
File Edit View Insert Runtime Tools Help
+ Code + Text
[6] [[ 9.5]
 [ 2. ]
 [ 8.7]
 [ 7.9]
 [ 8.2]
 [ 2.2]
 [ 1.5]
 [ 9. ]
 [ 3. ]
 [ 5.9]
 [ 4.1]
 [ 3.2]
 [ 9.6]
 [ 1.3]
 [ 5.1]
 [ 1.1]
 [ 4.9]
 [10.5]
 [10.3]
 [ 3.7]
 [ 3.2]
 [ 4. ]
 [ 4. ]
 [ 2.9]]
[7] print(x_test)
[[5.3]
 [ 2.9]
 [ 3.2]
 [ 4. ]
 [ 4. ]
 [ 2.9]]

```

✓ 0s completed at 8:47PM

Program1_FML_Prac_File.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

```

[5.3]
[7.1]
[3.9]
[6. ]
[4.5]
[6.8]

[8] print(y_train)
[116969. 43525. 109431. 101302. 113812. 39891. 37731. 105582. 60150.
81363. 57081. 54445. 112635. 46205. 66029. 39343. 67938. 121872.
122391. 57189. 64445. 56957. 55794. 56642.]

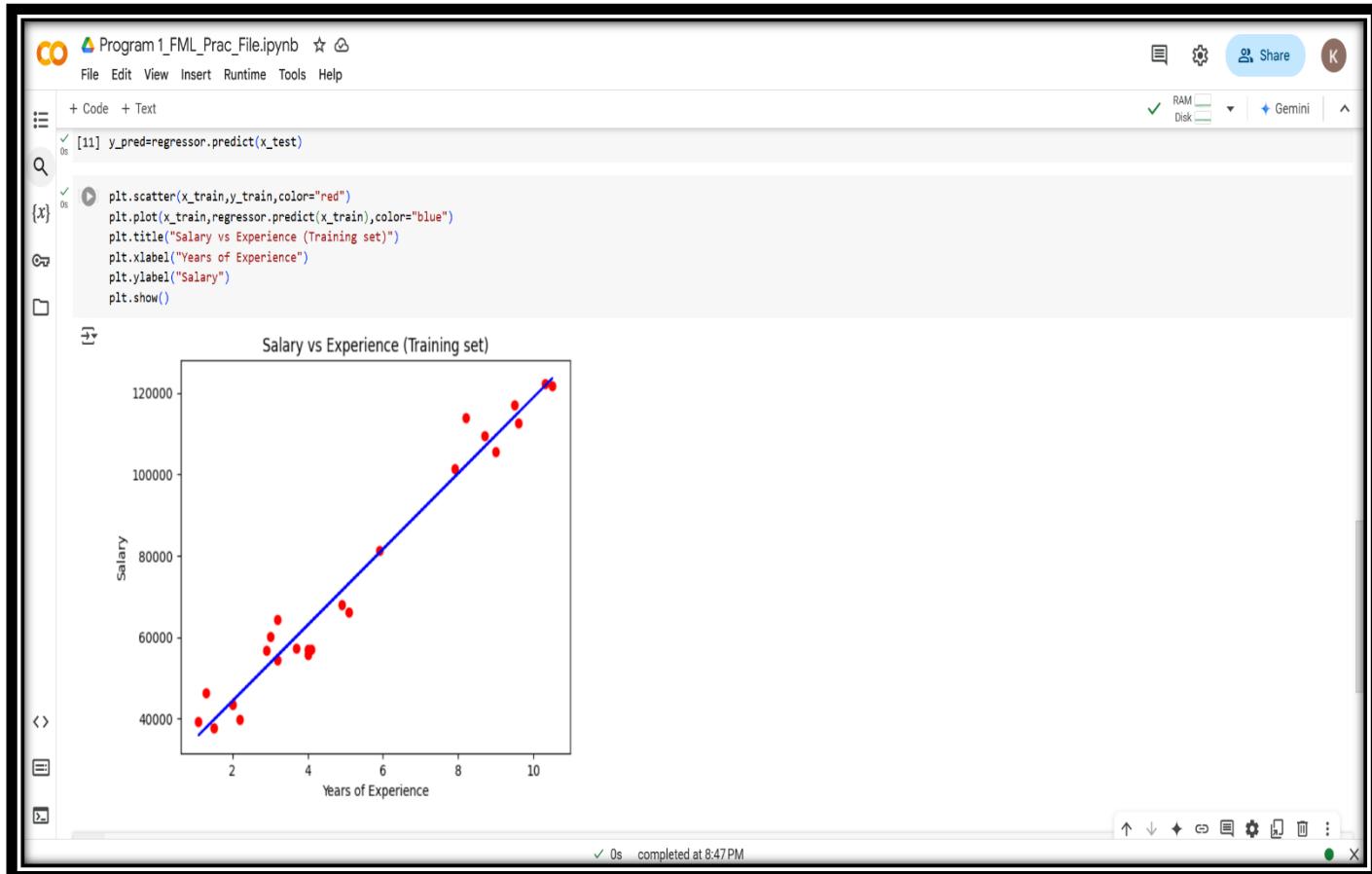
[9] print(y_test)
[83088. 98273. 63218. 93940. 61111. 91738.]

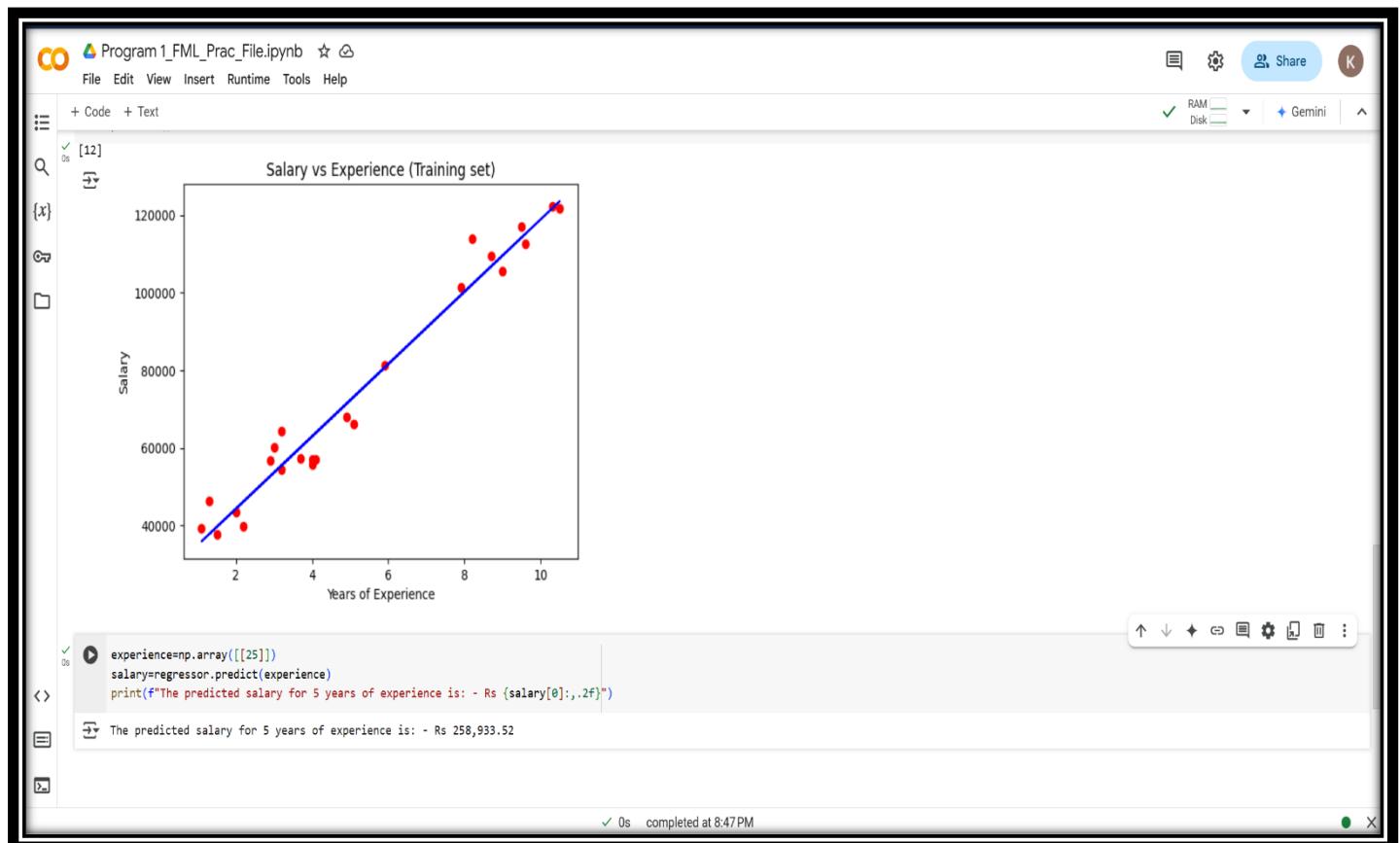
[10] from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train, y_train)

LinearRegression()

```

0s completed at 8:47PM





Learning Outcome:

EXPERIMENT 2

Problem statement: Study and implement Logistic Regression.

Theory:

Dataset:

Age	EstimatedSalary	Purchased
19	19000	0
35	20000	0
26	43000	0
27	57000	0
19	76000	0
27	58000	0
27	84000	0
32	150000	1
25	33000	0
35	65000	0
26	80000	0
26	52000	0
20	86000	0
32	18000	0
18	82000	0
29	80000	0
47	25000	1
45	26000	1
46	28000	1
48	29000	1
45	22000	1
47	49000	1
48	41000	1
45	22000	1
46	23000	1
47	20000	1
49	28000	1

KUNSH SABHARWAL

Program 2_FML_Prac_File.ipynb

```
[1] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

[2] df=pd.read_csv("Social_Network_Ads (1).csv")

[3] print(df.head())

   Age EstimatedSalary Purchased
0   19          19000      0
1   35          20000      0
2   26          43000      0
3   27          57000      0
4   19          76000      0

[4] x=df.iloc[:, :-1].values
y=df.iloc[:, -1].values

[5] from sklearn.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 = 1)

[6] print(x_train)

[[ 36 126000]
 [ 47 20000]
 [ 30 79000]
 [ 59 83000]
 [ 29 80000]
 [ 43 112000]
 [ 38 88000]
 [ 58 144000]
 [ 26 150000]]
```

Program 2_FML_Prac_File.ipynb

```
print(x_test)

[[ 34 115000]
 [ 23 66000]
 [ 56 60000]
 [ 31 118000]
 [ 48 35000]
 [ 47 113000]
 [ 39 79000]
 [ 52 38000]
 [ 24 58000]
 [ 37 53000]
 [ 42 80000]
 [ 46 28000]
 [ 42 73000]
 [ 37 62000]
 [ 60 42000]
 [ 36 52000]
 [ 58 95000]
 [ 43 129000]
 [ 27 89000]
 [ 23 82000]
 [ 38 112000]
 [ 35 50000]
 [ 36 99000]
 [ 37 144000]
 [ 26 35000]
 [ 42 70000]
 [ 43 133000]
 [ 38 50000]
 [ 46 96000]
 [ 35 44000]
 [ 38 113000]
 [ 39 71000]
 [ 26 52000]
 [ 54 108000]
 [ 33 51000]
 [ 26 16000]]
```

KUNSH SABHARWAL

Program 2_FML_Prac_File.ipynb

```
+ Code + Text

[8] print(y_train)
[8] [0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 0 1 1 1 0 0
0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 1 0 1 1 0 0 0 1 1 0 0 1 1 0 1 0 0 1
0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 1 0 0 1 1 0 1 0 0 1
1 1 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 0 1 1 0 0 0 1 0 0 1 0 0 0 0 1
1 1 0 0 0 0 0 1 0 0 1 1 0 0 0 1 0 1 0 0 1 0 1 0 0 0 1 0 1 0 0 1 0
0 0 0 1 0 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 1 0 0 1 1 0 0 0 1 0 0 1
1 0 1 0 1 0 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 0 1 1 0 0 0 1 0 0 0 1
0 0 0 1 0 1 0 1 0 1 1 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1
0 1 1 0]

[9] print(y_test)
[9] [0 0 1 0 0 0 1 0 0 0 0 0 1 1 1 1 0 0 1 0 1 1 0 1 0 1 0 1 0 0 0 0 1 0 0
0 1 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 1 0 1 0 0 0 1 1 0 0 0 0 0 1 0
0 1 0 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 1 0 0 1 1 0 0 0 1 0 1 1 0 0 1 0 0 1
0 1 0 0 0 0 0 1 1 0 0 0 1 0 1 0 1 1 0 0 0 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1
0 1 1 0]

[10] from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

[11] print(x_train)
[11] [[-1.11474707e-01  1.6890916e+00]
 [ 9.67312783e-01 -1.45199398e+00]
 [-6.99904247e-01  2.96801165e-01]
 [ 2.14417186e+00  4.15363548e-01]
 [-7.97975837e-01  3.26441761e-01]
 [ 5.75026423e-01  1.2749482e+00]
 [ 8.46684727e-02  3.26441761e-01]
 [ 2.04610027e+00  2.22343989e+00]
 [-1.11474707e-01  1.6890916e+00]
 [ 1.16345596e+00 -1.21486922e+00]]
```

✓ 0s completed at 6:58PM

Program 2_FML_Prac_File.ipynb

```
+ Code + Text

[8] print(x_test)
[8] [[-0.30761789  1.36386261]
 [-1.38640538 -0.08832658]
 [ 1.84995789 -0.26637015]
 [-0.60183266  1.4527844]
 [ 1.06538437 -1.00738505]
 [ 0.96731278  1.30458142]
 [ 0.18274086  0.29680117]
 [ 1.45767073 -0.91846326]
 [-1.28833379 -0.32565134]
 [-0.01340312 -0.47385432]
 [ 0.47695483  0.32644176]
 [ 0.86924119 -1.21486922]
 [ 0.47695483  0.11895759]
 [-0.01340312 -0.20708896]
 [ 2.24224345 -0.79990088]
 [-0.11147471 -0.58349492]
 [ 2.04610027  0.7710507]
 [ 0.57502642  1.77883095]
 [-0.99411982  0.59320712]
 [-1.38640538  0.38572295]
 [ 0.084665847  1.27494882]
 [-0.2095463 -0.56277611]
 [-0.11147471  0.88961308]
 [-0.01340312  2.22343989]
 [-1.09219061 -1.00738505]
 [ 0.47695483  0.03000358]
 [ 0.57502642  1.89739333]
 [ 0.084665847 -0.56277611]
 [ 0.86924119  0.80069129]
 [-0.2095463 -0.74061968]
 [ 0.084665847  1.30458142]
 [ 0.18274086  0.0596764]
 [-1.09219061 -0.58349492]
 [ 1.65381391  1.15637844]
 [-0.40568948 -0.53313551]
 [-1.09219061 -1.57055636]]
```

✓ 0s completed at 6:58PM

KUNSH SABHARWAL

A screenshot of a Jupyter Notebook interface. The title bar says "Program 2_FML_Prac_File.ipynb". The menu bar includes File, Edit, View, Insert, Runtime, Tools, Help. The toolbar has icons for Code, Text, Share, and a profile picture. The code cell contains Python code for logistic regression:

```
[13]: from sklearn.linear_model import LogisticRegression  
classifier = LogisticRegression(random_state = 0)  
classifier.fit(x_train, y_train)  
  
[14]: purchased=classifier.predict(sc.transform([[30,87000]]))  
print(purchased)  
  
[15]: purchased=classifier.predict(sc.transform([[40,84453]]))  
print(purchased)  
  
[16]: y_pred = classifier.predict(x_test)  
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

The output cell shows the predicted values:

```
[0 0]  
[0 0]  
[1 1]  
[0 1]  
[0 1]  
[1 1]  
[0 0]  
[1 1]  
[0 0]  
[0 0]  
[1 1]  
[0 1]  
[0 1]  
[1 1]  
[0 0]  
[0 0]  
[1 1]  
[0 0]  
[0 0]  
[1 1]  
[0 0]  
[0 0]  
[0 0]
```

At the bottom, it says "✓ 0s completed at 6:58PM".

A screenshot of a Jupyter Notebook interface, identical to the one above, showing the same code execution and output. The title bar says "Program 2_FML_Prac_File.ipynb". The menu bar includes File, Edit, View, Insert, Runtime, Tools, Help. The toolbar has icons for Code, Text, Share, and a profile picture. The code cell contains Python code for logistic regression:

```
[14]: purchased=classifier.predict(sc.transform([[30,87000]]))  
print(purchased)  
  
[15]: purchased=classifier.predict(sc.transform([[40,84453]]))  
print(purchased)  
  
[16]: y_pred = classifier.predict(x_test)  
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

The output cell shows the predicted values:

```
[0 0]  
[0 0]  
[1 1]  
[0 1]  
[0 1]  
[1 1]  
[0 0]  
[1 1]  
[0 0]  
[0 0]  
[1 1]  
[0 1]  
[0 1]  
[1 1]  
[0 0]  
[0 0]  
[1 1]  
[0 0]  
[0 0]  
[1 1]  
[0 0]  
[0 0]  
[0 0]
```

At the bottom, it says "✓ 0s completed at 6:58PM".

Learning Outcome:

EXPERIMENT 3 (Beyond Curriculum)

Problem statement: Study and implement Multiple Regression and Polynomial Linear Regression.

Theory:

Datasets:**1) 50_Startups Dataset**

1	A	B	C	D	E	
	R&D Spend	Administration	Marketing Spend	Spend	State	Profit
2	165349.2	136897.8	471784.1	New York	192261.83	
3	162597.7	151377.59	443898.53	California	191792.06	
4	153441.51	101145.55	407934.54	Florida	191050.39	
5	144372.41	118671.85	383199.62	New York	182901.99	
6	142107.34	91391.77	366168.42	Florida	166187.94	
7	131876.9	99814.71	362861.36	New York	156991.12	
8	134615.46	147198.87	127716.82	California	156122.51	
9	130298.13	145530.06	323876.68	Florida	155752.6	
10	120542.52	148718.95	311613.29	New York	152211.77	
11	123334.88	108679.17	304981.62	California	149759.96	
12	101913.08	110594.11	229160.95	Florida	146121.95	
13	100671.96	91790.61	249744.55	California	144259.4	
14	93863.75	127320.38	249839.44	Florida	141585.52	
15	91992.39	135495.07	252664.93	California	134307.35	
16	119943.24	156547.42	256512.92	Florida	132602.65	
17	114523.61	122616.84	261776.23	New York	129917.04	
18	78013.11	121597.55	264346.06	California	126992.93	
19	94657.16	145077.58	282574.31	New York	125370.37	
20	91749.16	114175.79	294919.57	Florida	124266.9	
21	86419.7	153514.11	0	New York	122776.86	
22	76253.86	113867.3	298664.47	California	118474.03	
23	78389.47	153773.43	299737.29	New York	111313.02	
24	73994.56	122782.75	303319.26	Florida	110352.25	
25	67532.53	105751.03	304768.73	Florida	108733.99	
26	77044.01	99281.34	140574.81	New York	108552.04	
27	64664.71	139553.16	137962.62	California	107404.34	
28	75328.87	144135.98	134050.07	Florida	105733.54	

2) Position_Salaries Dataset

A	B	C	
1	Position	Level	Salary
2	Business Analyst	1	45000
3	Junior Consultant	2	50000
4	Senior Consultant	3	60000
5	Manager	4	80000
6	Country Manager	5	110000
7	Region Manager	6	150000
8	Partner	7	200000
9	Senior Partner	8	300000
10	C-level	9	500000
11	CEO	10	1000000

Source Code with Outputs:

MULTIPLE REGRESSION

```
[{x} 1s [1] import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt

[0s] 2s [2] df=pd.read_csv("50_Startups (1).csv")

[0s] 3s [3] print(df.head())

    ↗ R&D Spend Administration Marketing Spend      State   Profit
0  165349.20    136897.80    471784.10  New York  192261.83
1  162597.70    151377.59    443898.53  California 191792.06
2  153441.51    181145.55    407934.54  Florida   191050.39
3  144372.41    118671.85    383199.62  New York  182901.99
4  142187.34    91391.77    366168.42  Florida   166187.94

[0s] 4s [4] x=df.iloc[:, :-1].values
     y=df.iloc[:, -1].values

[0s] 5s [5] print(x)

    ↗ [[165349.2 136897.8 471784.1 'New York']
[162597.7 151377.59 443898.53 'California']
[153441.5 181145.55 407934.54 'Florida']
[144372.41 118671.85 383199.62 'New York']
[142187.34 91391.77 366168.42 'Florida']
[131876.9 99814.71 362861.36 'New York']
[134615.46 147198.87 127716.82 'California']
[130298.13 145530.06 323876.68 'Florida']
[120542.52 148718.95 311613.29 'New York']]
```

✓ 0s completed at 8:26PM

[{x} 0s [6] print(y)

```
    ↗ [[192261.83 191792.06 191050.39 182901.99 166187.94 156991.12 156122.51
155752.6 152211.77 149759.96 146121.95 144259.4 141585.52 134307.35
132602.65 129917.04 126992.93 125370.37 124266.9 122776.86 118474.03
111313.02 110352.25 108733.99 108552.04 108748.34 105733.54 105008.31
103282.38 101084.64 99937.59 97483.56 97427.84 96778.92 96712.8
96479.51 90708.19 89949.14 81229.06 81005.76 78239.91 77798.83
71498.49 69758.98 65200.33 64926.08 49498.73 35673.41
14681.4 ]]
```

[{x} 1s [7] from sklearn.compose import ColumnTransformer
 from sklearn.preprocessing import OneHotEncoder
 ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])], remainder='passthrough')
 x = np.array(ct.fit_transform(x))

[0s] 8s [8] print(x)

```
    ↗ [[0.0 0.0 1.0 165349.2 136897.8 471784.1]
[1.0 0.0 0.0 162597.7 151377.59 443898.53]
[0.0 1.0 0.0 153441.51 181145.55 407934.54]
[0.0 0.0 1.0 144372.41 118671.85 383199.62]
[0.0 1.0 0.0 142187.34 91391.77 366168.42]
[0.0 0.0 1.0 131876.9 99814.71 362861.36]
[1.0 0.0 0.0 134615.46 147198.87 127716.82]
[0.0 1.0 0.0 130298.13 145530.06 323876.68]
[0.0 0.0 1.0 120542.52 148718.95 311613.29]
[1.0 0.0 0.0 123334.88 108679.17 304981.62]
[0.0 1.0 0.0 101913.08 118594.11 229160.95]
[1.0 0.0 0.0 100671.96 91790.61 249744.55]
[0.0 1.0 0.0 93863.75 127320.38 249839.44]
[1.0 0.0 0.0 91992.39 135495.07 252664.93]
[0.0 1.0 0.0 119943.24 156547.42 256512.92]
[0.0 0.0 1.0 114523.61 122616.84 261776.23]
[1.0 0.0 0.0 78013.11 121597.55 264346.06]
[0.0 0.0 1.0 10457.14 115077.58 382571.31]]
```

✓ 0s completed at 8:26PM

KUNSH SABHARWAL

Multiple/Polynomial Linear Regression_FML_Prac_File.ipynb

File Edit View Insert Runtime Tools Help

RAM Disk Gemini

```
+ Code + Text
print(y)
[192261.83 191792.06 191050.39 182901.99 166187.94 156991.12 156122.51
155752.6 152211.77 149759.96 146121.95 144259.4 141585.52 134307.35
132602.65 129917.04 126992.93 125370.37 124266.9 122776.86 118474.03
11313.02 110352.25 108733.99 108552.04 107484.34 105733.54 105008.31
103282.38 101804.64 99937.59 97483.56 97427.84 96778.92 96712.8
96479.51 90708.19 89949.14 81229.06 81005.76 78239.91 77798.83
71498.49 69758.98 65200.33 64926.08 49490.75 42559.73 35673.41
14681.4 ]

[10] from sklearn.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 = 1)

[11] print(x_train)
[[0.0 1.0 0.0 28663.76 127056.21 201126.82]
[0.0 0.0 1.0 86419.7 153514.11 0.0]
[1.0 0.0 0.0 23640.93 96189.63 148001.11]
[1.0 0.0 0.0 0.0 116983.8 45173.06]
[0.0 1.0 0.0 75328.87 144135.98 134850.07]
[0.0 1.0 0.0 73994.56 122782.75 303319.26]
[1.0 0.0 0.0 91992.39 135495.07 252664.93]
[0.0 1.0 0.0 27892.92 84710.77 164470.71]
[0.0 0.0 1.0 94657.16 145077.58 282574.31]
[0.0 0.0 1.0 1000.23 124153.04 1983.93]
[0.0 0.0 1.0 77044.01 99281.34 140574.81]
[0.0 1.0 0.0 67532.53 105751.03 384768.73]
[0.0 1.0 0.0 142107.34 91391.77 366168.42]
[0.0 1.0 0.0 55493.95 103057.49 214634.81]
[0.0 1.0 0.0 119943.24 156547.42 256512.92]
[0.0 1.0 0.0 61994.48 115641.28 91131.24]
[0.0 1.0 0.0 101913.08 110594.11 229160.95]
[0.0 1.0 0.0 66051.52 182645.56 118148.2]
[1.0 0.0 0.0 22177.74 154806.14 28334.72]

0s completed at 8:26 PM
```

Multiple/Polynomial Linear Regression_FML_Prac_File.ipynb

File Edit View Insert Runtime Tools Help

RAM Disk Gemini

```
+ Code + Text
print(x_test)
[[0.0 0.0 1.0 72107.6 127864.55 353183.81]
[0.0 0.0 1.0 46014.02 85047.44 205517.64]
[1.0 0.0 0.0 28754.33 118546.05 172795.67]
[0.0 0.0 1.0 20229.59 65947.93 185265.1]
[0.0 1.0 0.0 153441.51 101145.55 407934.54]
[0.0 0.0 1.0 144372.41 118671.85 383199.62]
[0.0 0.0 1.0 542.85 51743.15 0.0]
[0.0 0.0 1.0 65605.48 153032.06 107138.38]
[0.0 1.0 0.0 1315.46 115816.21 297114.46]
[0.0 0.0 1.0 61136.38 152781.92 88218.23]
[1.0 0.0 0.0 63408.86 129219.61 46085.25]
[1.0 0.0 0.0 38558.51 82882.89 174999.3]
[0.0 0.0 1.0 78389.47 153773.43 299737.29]

[13] print(y_train)
[ 98708.19 122776.86 71498.49 14681.4 105733.54 110352.25 134307.35
77798.83 125370.37 64926.08 108552.04 106733.99 166187.94 96778.92
132602.65 99937.59 146121.95 103282.38 65200.33 96712.8 124266.9
118474.03 107404.34 156122.51 155752.6 42559.73 191792.06 126992.93
192261.83 129917.04 156991.12 144259.4 149759.96 152211.77 141585.52
69758.98 89949.14]

[14] print(y_test)
[ 105008.31 96479.51 78239.91 81229.06 191050.39 182901.99 35673.41
101804.64 49490.75 97483.56 97427.84 81005.76 111313.02]

[15] from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train, y_train)

LinearRegression()
```

0s completed at 8:26 PM

KUNSH SABHARWAL

Multiple/Polynomial Linear Regression_FML_Prac_File.ipynb

File Edit View Insert Runtime Tools Help

RAM Disk Gemini

```
[16] y_pred = regressor.predict(x_test)
    np.set_printoptions(precision=2)
    print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))

{x}
[[116262.61 105008.31]
 [ 91489.96 96479.51]
 [ 76360.93 78239.91]
 [ 71194.69 81229.06]
 [179498.9 191050.39]
 [172318.4 182901.99]
 [ 50088.92 35673.41]
 [18938.27 101004.64]
 [ 59896.19 49490.75]
 [ 99000.9 97483.56]
 [ 98497.76 97427.84]
 [ 83942.43 81005.76]
 [119227.03 111313.02]]
```

POLYNOMIAL LINEAR REGRESSION

```
[17] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt

[18] df1=pd.read_csv("Position_Salaries (1).csv")

[19] print(df1.head())
Position Level Salary
0 Business Analyst 1 45000
1 Junior Consultant 2 50000
2 Senior Consultant 3 60000
3 Manager 4 80000
4 Country Manager 5 110000
```

0s completed at 8:26 PM

Multiple/Polynomial Linear Regression_FML_Prac_File.ipynb

File Edit View Insert Runtime Tools Help

RAM Disk Gemini

```
[20] x1=df1.iloc[:,1:-1].values
    y1=df1.iloc[:, -1].values

{x}
print(x1)

[[ 1]
 [ 2]
 [ 3]
 [ 4]
 [ 5]
 [ 6]
 [ 7]
 [ 8]
 [ 9]
 [10]]
```

```
[22] print(y1)
[ 45000 50000 60000 80000 110000 150000 200000 300000 500000
1000000]
```

```
[23] from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(x1, y1)

LinearRegression()
```

```
[24] from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 4) #answer upto the 4th degree of x
x_poly = poly_reg.fit_transform(x1)
lin_reg_2 = LinearRegression()
```

0s completed at 8:26 PM

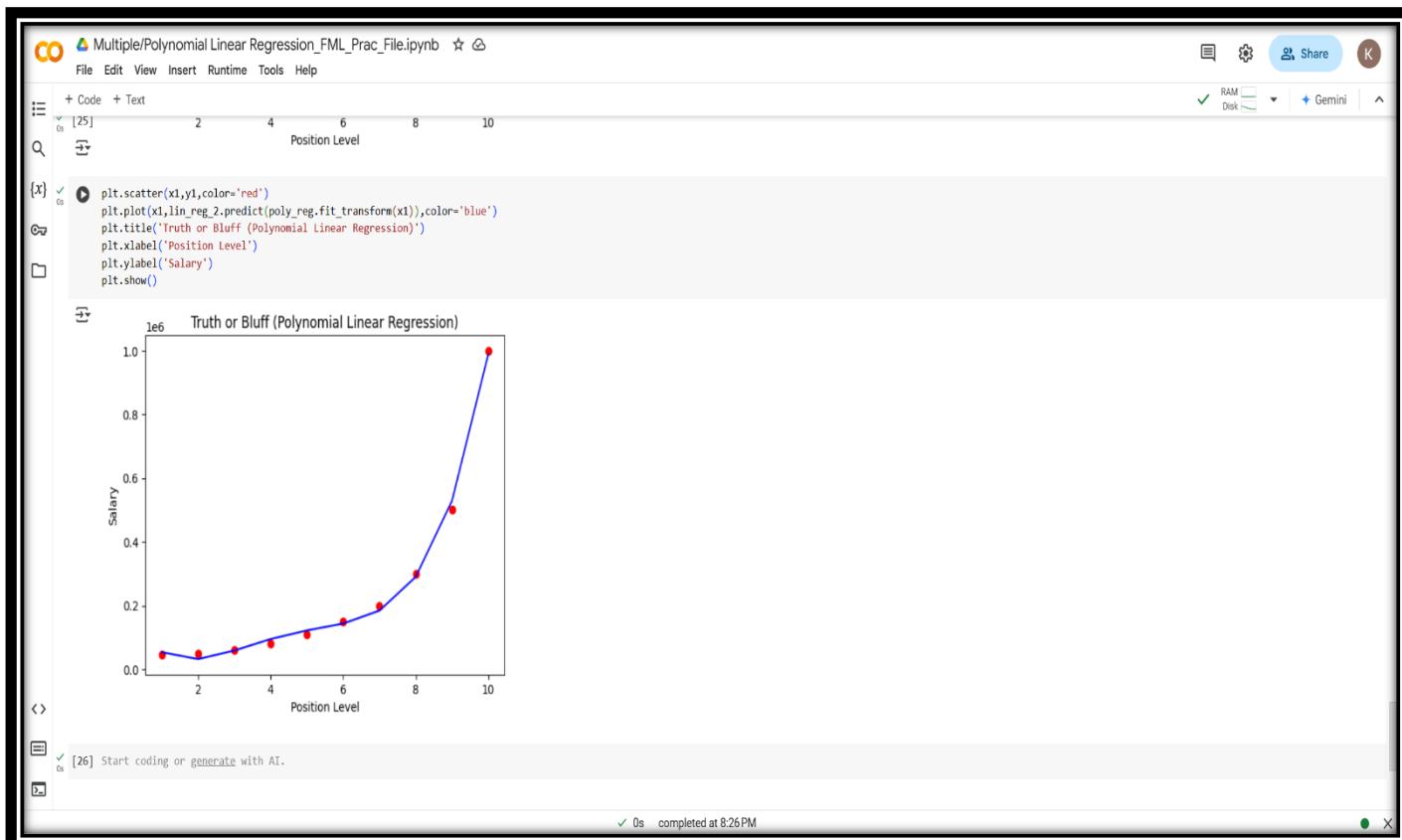
Multiple/Polynomial Linear Regression_FML_Prac_File.ipynb

```

File Edit View Insert Runtime Tools Help
+ Code + Text
[20] x1=df1.iloc[:,1:-1].values
y1=df1.iloc[:, -1].values
{x}
[21] print(x1)
[[ 1
[ 2
[ 3
[ 4
[ 5
[ 6
[ 7
[ 8
[ 9
[10]]
[22] print(y1)
[ 45000 50000 60000 80000 110000 150000 200000 300000 500000
1000000]
[23] from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(x1, y1)
LinearRegression()
[24] from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 4) #answer upto the 4th degree of x
x_poly = poly_reg.fit_transform(x1)
lin_reg_2 = LinearRegression()

```

0s completed at 8:26PM



Learning Outcome:

EXPERIMENT 3

Problem statement: Study & Implement K-Nearest Neighbour (KNN).

Theory:

Dataset:

Age	EstimatedSalary	Purchased
19	19000	0
35	20000	0
26	43000	0
27	57000	0
19	76000	0
27	58000	0
27	84000	0
32	150000	1
25	33000	0
35	65000	0
26	80000	0
26	52000	0
20	86000	0
32	18000	0
18	82000	0
29	80000	0
47	25000	1
45	26000	1
46	28000	1
48	29000	1
45	22000	1
47	49000	1
48	41000	1
45	22000	1
46	23000	1
47	20000	1
49	28000	1

Source Code with Outputs:

Program 3_FML_Prac_File.ipynb

```

File Edit View Insert Runtime Tools Help
+ Code + Text
RAM Disk Gemini
[7] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

[8] df=pd.read_csv("Social_Network_Ads (1).csv")

[9] print(df.head())
   Age  EstimatedSalary  Purchased
0    19           19000       0
1    35           20000       0
2    26           43000       0
3    27           57000       0
4    19           76000       0

[10] x=df.iloc[:, :-1].values
     y=df.iloc[:, -1].values

[11] from sklearn.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 = 1)

[12] print(x_train)
[[ 20  49000]
 [ 46  88000]
 [ 31  34000]
 [ 47  30000]
 [ 35  50000]
 [ 39  96000]
 [ 33 113000]
 [ 49  86000]
 [ 45  79000]
 [ 44  39000]
 [ 41  59000]
 [ 42  53000]
 [ 35  73000]
 [ 41  72000]
 [ 27  96000]
 [ 30 116000]
 [ 41  52000]
 [ 41  52000]
 [ 20  82000]
 [ 46  41000]
 [ 27  31000]
 [ 35  71000]
 [ 49  28000]
 [ 35  91000]
 [ 37  75000]
 [ 32 117000]
 [ 36  75000]
 [ 20  86000]
 [ 38  50000]
 [ 49  36000]
 [ 40  65000]
 [ 37  77000]]

```

0s completed at 8:43PM

Program 3_FML_Prac_File.ipynb

```

File Edit View Insert Runtime Tools Help
+ Code + Text
RAM Disk Gemini
[12] print(x_train)
[[ 20  49000]
 [ 46  88000]
 [ 31  34000]
 [ 47  30000]
 [ 35  50000]
 [ 39  96000]
 [ 33 113000]
 [ 49  86000]
 [ 45  79000]
 [ 44  39000]
 [ 41  59000]
 [ 42  53000]
 [ 35  73000]
 [ 41  72000]
 [ 27  96000]
 [ 30 116000]
 [ 41  52000]
 [ 41  52000]
 [ 20  82000]
 [ 46  41000]
 [ 27  31000]
 [ 35  71000]
 [ 49  28000]
 [ 35  91000]
 [ 37  75000]
 [ 32 117000]
 [ 36  75000]
 [ 20  86000]
 [ 38  50000]
 [ 49  36000]
 [ 40  65000]
 [ 37  77000]]

```

0s completed at 8:43PM

KUNSH SABHARWAL

Program 3_FML_Prac_File.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

[14] print(y_train)

```
[0 1 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 0 1 1 1 0 0  
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 1 0 1 1 0 0 0 1 1 0 0 1 1 0 1  
0 0 0 0 0 0 1 0 0 1 0 0 1 0 1 0 1 0 0 0 1 0 0 1 0 0 1 1 0 0 0 1  
1 1 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 0 0 1 1 0 1 1 0 0 0 1 0 0 1 0 0 0 0 1  
1 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 1 0 0 1 0  
0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 1 0 1 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 0 1  
1 0 1 0 1 0 0 0 0 0 1 0 1 0 0 1 1 0 0 1 1 0 0 0 1 0 1 1 1 0 0 0 1 0 0 0 1  
0 0 0 1 0 1 0 1 0 1 1 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0  
0 1 1 0]
```

[14] print(x_test)

```
[[ 36 33000]  
[ 39 61000]  
[ 36 118000]  
[ 39 122000]  
[ 26 118000]  
[ 38 65000]  
[ 20 36000]  
[ 49 89000]  
[ 31 18000]  
[ 48 141000]  
[ 34 72000]  
[ 39 73000]  
[ 35 72000]  
[ 48 131000]  
[ 53 82000]  
[ 56 133000]  
[ 60 83000]  
[ 27 58000]  
[ 28 87000]]
```

0s completed at 8:43 PM

Program 3_FML_Prac_File.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

[15] print(y_test)

```
[0 1 0 1 0 0 0 1 0 0 0 0 0 1 1 1 1 0 0 1 0 1 1 0 1 0 1 0 1 0 0 0 0 1 0 0 0  
0 1 0 1 1 0 0 1 1 1 1 0 1 0 0 1 1 1 0 1 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0 1 0  
0 1 0 0 0 0 0 1 1 0 0 0 1 0 1 0 1 1 1 0 0 1 0 0]
```

[16] from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

[17] print(x_train)

```
[[ -1.68062015e+00 -5.92416706e-01]  
[ 8.69241193e-01 5.63566526e-01]  
[-6.01832657e-01 -1.03702564e+00]  
[ 9.67312783e-01 -1.15558802e+00]  
[-2.09546297e-01 -5.62776110e-01]  
[ 1.82740063e-01 8.00691292e-01]  
[-4.05689477e-01 1.30458142e+00]  
[ 1.16345596e+00 5.04285335e-01]  
[ 7.71169603e-01 2.96801165e-01]  
[ 6.730988013e-01 -8.88822663e-01]  
[ 3.78883243e-01 -2.96018749e-01]  
[ 4.76954833e-01 -4.73854323e-01]  
[-2.09546297e-01 1.18957591e-01]  
[ 3.78883243e-01 8.93169951e-02]  
[-9.94119817e-01 8.00691292e-01]  
[-6.99904247e-01 1.39358321e+00]  
[ 3.78883243e-01 -5.03494919e-01]  
[ 3.78883243e-01 -5.03494919e-01]  
[-1.68062015e+00 3.85722952e-01]]
```

0s completed at 8:43 PM

KUNSH SABHARWAL

Program 3_FML_Prac_File.ipynb

```
+ Code + Text
print(x_test)
[[-0.11147471 -1.06666624]
 [ 0.18274006 -0.23672956]
 [-0.11147471  1.4527844 ]
 [ 0.18274006  1.57134678]
 [-1.09219061  1.4527844 ]
 [ 0.08466847 -0.11816717]
 [-1.68062015 -0.97774445]
 [ 1.16345596  0.59320712]
 [-0.60183266 -1.51127517]
 [ 1.06538437  2.1345181 ]
 [-0.30761789  0.089317 ]
 [ 0.18274006  0.11895759]
 [-0.2095463  0.089317 ]
 [ 1.06538437  1.83811214]
 [ 1.55574232  0.38572295]
 [ 1.84995709  1.89739333]
 [ 2.24224345  0.41536355]
 [-0.99411902 -0.32565134]
 [-0.89604743  0.53392593]
 [ 2.24224345  0.97853487]
 [ 0.28081165  0.17823878]
 [ 1.26152755  0.56356653]
 [ 0.67309801  2.07523691]
 [ 0.96731278 -0.77026028]
 [ 0.7711694 -1.27415041]
 [-1.89219061 -1.60019696]
 [ 2.04610027 -0.6516979 ]
 [ 1.16345596  0.14859819]
 [ 1.55574232 -1.03702564]
 [ 1.45767073  1.33422201]
 [ 0.18274006 -0.79990088]
 [-1.77869174  0.20787938]
```

✓ 0s completed at 8:43 PM

Program 3_FML_Prac_File.ipynb

```
+ Code + Text
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(x_train, y_train)

KNeighborsClassifier()

purchased=classifier.predict(sc.transform([[30,87000]]))
print(purchased)

[0]

y_pred = classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))

[[0 0]
 [0 0]
 [1 1]
 [1 1]
 [1 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [1 1]]
```

✓ 0s completed at 8:43 PM

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** CO Program 3_FML_Prac_File.ipynb
- File Menu:** File Edit View Insert Runtime Tools Help
- Toolbar:** Share, RAM Disk Gemini
- Code Cells:**
 - Cell 21: Prints a list of 20 [0, 0] pairs.
 - Cell 22: Prints a confusion matrix and accuracy score.

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print("Accuracy: - ", 100*accuracy_score(y_test, y_pred), "%")
```
- Output:** The output of Cell 22 shows a confusion matrix [[50 8], [4 38]] and an accuracy of 88.0%.
- Bottom Status:** 0s completed at 8:43 PM

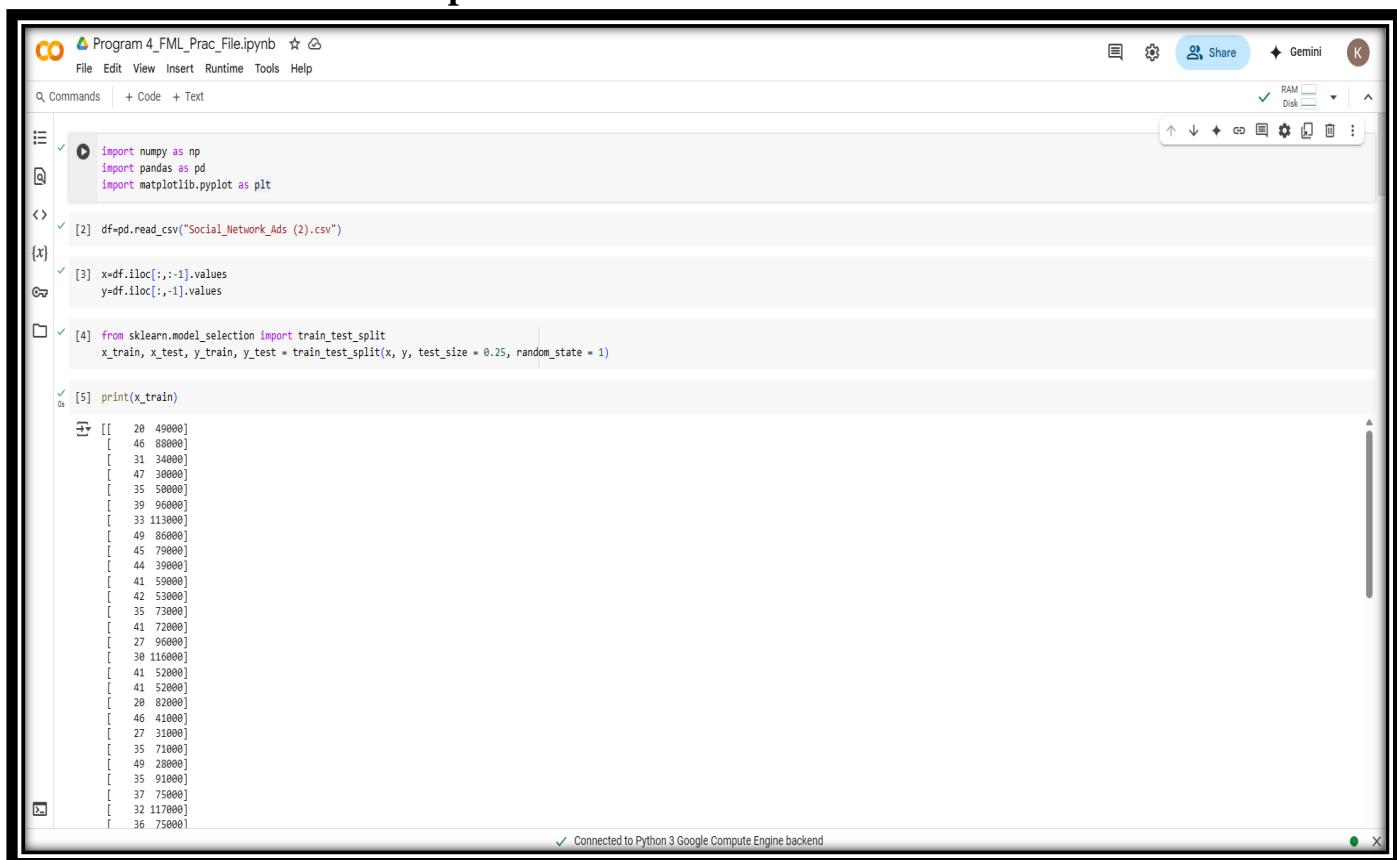
Learning Outcome:

EXPERIMENT 4

Problem statement: Study & Implement classification using SVM.

Theory:

Source Code with Outputs:



```

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

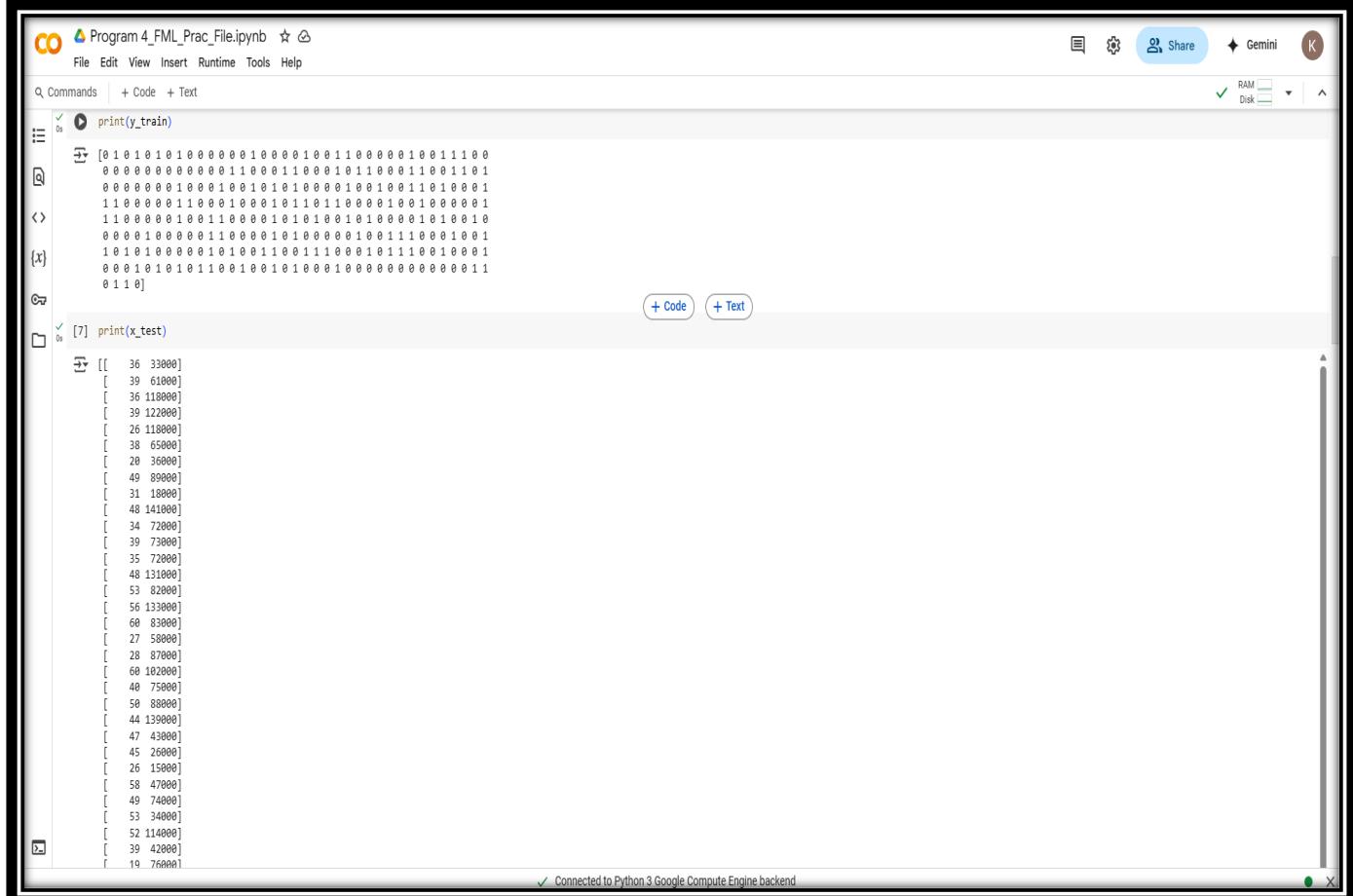
df=pd.read_csv("Social_Network_Ads (2).csv")
x=df.iloc[:, :-1].values
y=df.iloc[:, -1].values

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 1)

print(x_train)

```

[[28 49000]
[46 88000]
[31 34000]
[47 30000]
[35 50000]
[39 96000]
[33 113000]
[49 86000]
[45 79000]
[44 39000]
[41 50000]
[42 53000]
[35 73000]
[41 72000]
[27 96000]
[38 116000]
[41 52000]
[41 52000]
[20 82000]
[46 41000]
[27 31000]
[35 71000]
[49 28000]
[35 91000]
[37 75000]
[32 117000]
[36 75000]



```

print(y_train)

```

[0 1 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 0 0 1 1 0 0 1
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 0 1 1 0 0 1
0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1 0 0 1 1 0 0 0 1
1 1 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 0 1 1 0 1 1 0 0 0 0 1 0 0 1 0 0 0 0 1
1 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 1
0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 1 0 1 0 0 0 0 1 0 0 1 1 0 0 0 1 0 0 1
1 0 1 0 1 0 0 0 0 0 1 0 1 0 0 1 0 0 1 1 0 0 0 1 0 1 1 0 0 1 0 0 0 1
0 0 0 1 0 1 0 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1
0 1 1 0]

```

print(x_test)

```

[[36 33000]
[39 61000]
[36 118000]
[39 122000]
[26 118000]
[38 65000]
[28 36000]
[49 89000]
[31 18000]
[48 141000]
[34 72000]
[39 73000]
[35 72000]
[48 131000]
[53 82000]
[56 133000]
[60 83000]
[27 58000]
[28 87000]
[60 102000]
[40 75000]
[50 88000]
[44 139000]
[47 43000]
[45 26000]
[26 15000]
[58 47000]
[49 74000]
[53 34000]
[52 114000]
[39 42000]
[19 76000]

KUNSH SABHARWAL

A screenshot of a Jupyter Notebook interface. The title bar says "Program 4_FML_Prac_File.ipynb". The code cell [8] contains:

```
print(y_test)
```

The output shows a large list of numerical values:

```
[ 0 0 1 1 0 0 0 1 0 0 0 0 1 1 1 1 0 0 1 0 1 0 1 0 1 0 0 0 1 0 0  
0 1 0 1 0 0 1 1 1 0 1 0 1 0 1 1 0 0 0 1 1 0 0 0 0 1 0  
0 1 0 0 0 0 1 1 0 0 1 0 1 0 1 1 0 0 1 1 0 0 0 0 1 0]
```

Cell [9] contains:

```
from sklearn.preprocessing import StandardScaler  
sc = StandardScaler()  
x_train = sc.fit_transform(x_train)  
x_test = sc.transform(x_test)
```

Cell [10] contains:

```
print(x_train)
```

The output shows a large list of numerical values:

```
[[ -1.60062015e+00 -5.92416706e-01  
8.69241193e-01 5.6556526e-01  
-6.01832657e-01 -1.03702564e+00  
[ 9.67312783e-01 -1.15558802e+00  
[-2.09546297e-01 -5.62776110e-01  
[ 1.82740863e-01 8.00691292e-01  
[-4.0589477e-01 1.30453142e+00  
[ 1.16345596e+00 5.04283335e-01  
[ 7.71169683e-01 2.9680165e-01  
[ 6.73098013e-01 -8.88822663e-01  
[ 3.7883243e-01 -2.96810749e-01  
[ 4.76954833e-01 -4.73854323e-01  
[-2.09546297e-01 1.18957591e-01  
[ 3.7883243e-01 8.91169951e-02  
[-9.94119817e-01 8.00691292e-01  
[-6.09904247e-01 1.39350321e+00  
[ 3.7883243e-01 -5.03494919e-01  
[ 3.7883243e-01 -5.03494919e-01  
[-1.68662015e+00 3.85722952e-01  
[ 8.69241193e-01 -8.29541472e-01  
[-9.94119817e-01 -1.12594743e+00  
[-2.09546297e-01 5.96763994e-02  
[ 1.16345596e+00 -1.21486922e+00  
[-2.09546297e-01 6.52488313e-01  
[-1.34031173e-02 1.78238782e-01  
[-5.03761867e-01 1.42314380e+00  
[-1.11474707e-01 1.78238782e-01  
[-1.68662015e+00 5.04283335e-01  
[ 8.46684727e-02 -5.62776110e-01  
[ 1.16345596e+00 -9.77744450e-01  
[ 2.08011553e-01 -1.18167175e-01]
```

At the bottom, it says "Connected to Python 3 Google Compute Engine backend".

A screenshot of a Jupyter Notebook interface. The title bar says "Program 4_FML_Prac_File.ipynb". The code cell [8] contains:

```
print(x_test)
```

The output shows a large list of numerical values:

```
[[ -0.11147471 -1.06666624]  
[ 0.18274086 -0.23672956]  
[ -0.11147471 1.4527844 ]  
[ 0.18274086 1.57134678]  
[-0.09219061 1.4527844 ]  
[ 0.08466847 -0.11186717]  
[-1.68662015 -0.97774445]  
[ 1.16345596 0.59320712]  
[-0.00183266 -1.51127517]  
[ 1.06538437 2.1345181 ]  
[-0.38761789 0.089317 ]  
[ 0.18274086 0.11895759]  
[-0.209546 0.089317 ]  
[ 1.06538437 1.83811214]  
[ 1.55574232 0.38572295]  
[ 1.84095709 1.89739333]  
[ 2.24224345 0.41536355]  
[-0.99411982 -0.32565134]  
[-0.89604743 0.53392593]  
[ 2.24224345 0.97853487]  
[ 0.20801165 0.17823878]  
[ 1.26152755 0.56356653]  
[ 0.67309901 2.07523691]  
[ 0.96731278 -0.77026028]  
[ 0.77116901 -1.27415041]  
[-1.09219061 -1.60019606]  
[ 2.04610027 -0.6516979 ]  
[ 1.16345596 0.14859819]  
[ 1.55574232 -1.03702564]  
[ 1.45767073 1.33422281]  
[ 0.18274086 -0.79990088]  
[-1.77869174 0.28787938]  
[-1.87676333 0.50428533]  
[ 1.94802868 0.14859819]  
[-0.99411982 0.44580414]  
[-0.69909425 0.32644176]  
[-1.48447697 -1.51127517]  
[-0.50376107 0.59428533]  
[ 1.26152755 -1.45193998]  
[-1.77869174 -1.303791 ]  
[ 0.96731278 2.22343989]  
[ 2.04610027 0.94889427]  
[-0.30761789 1.36386261]  
[-1.38640538 -0.08852658]
```

At the bottom, it says "Connected to Python 3 Google Compute Engine backend".

```
[12]: from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(x_train, y_train)

[13]: purchased=classifier.predict(sc.transform([[30,87000]]))
print(purchased)

[14]: y_pred = classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))

[[0 0]
 [1 1]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [1 1]
 [1 1]
 [1 1]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [1 1]
 [1 1]
 [0 0]
 [0 1]
 [0 0]
 [1 1]]
```

Connected to Python 3 Google Compute Engine backend

```
[15]: from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print("Accuracy: - ",100*accuracy_score(y_test, y_pred),"%)"

[[52  6]
 [12 30]]
Accuracy: -  82.0 %
```

[] Start coding or generate with AI.

Learning Outcome:

EXPERIMENT 5

Problem statement: Study & Implement Bagging using Random Forests.

Theory:

Source Code:

Program 5_FML_Prac_File.ipynb

```

File Edit View Insert Runtime Tools Help
Commands + Code + Text

[1] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

[2] df=pd.read_csv("Social_Network_Ads (2).csv")

[3] x=df.iloc[:, :-1].values
y=df.iloc[:, -1].values

[4] from sklearn.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 = 1)

[5] print(x_train)
[[ 36 126000]
 [ 47 20000]
 [ 30 79000]
 [ 59 83000]
 [ 29 80000]
 [ 43 112000]
 [ 38 80000]
 [ 58 144000]
 [ 36 125000]
 [ 49 20000]
 [ 18 52000]
 [ 30 15000]
 [ 59 88000]
 [ 27 57000]
 [ 38 71000]
 [ 31 89000]
 [ 47 34000]
 [ 31 74000]
 [ 37 72000]
 [ 48 57000]
 [ 59 130000]
 [ 49 65000]
 [ 48 90000]
 [ 46 22000]
 [ 54 104000]
 [ 35 20000]
 [ 49 141000]]

```

✓ 0s completed at 2:26PM

Program 5_FML_Prac_File.ipynb

```

File Edit View Insert Runtime Tools Help
Commands + Code + Text

[6] print(y_train)
[[ 0 1 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 0 1 1 1 0 0
 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 0 1 1 0 0 0 1 1 0 0 1 1 0 0 1
 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1
 1 1 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 0 1 1 0 0 0 0 1 0 0 1 0 0 0 0 1
 1 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 1 0 1 0 0 1 0
 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 0 1
 1 0 1 0 1 0 0 0 0 1 0 1 0 0 0 1 0 0 0 1 1 0 0 0 1 0 0 1 1 0 0 0 1 0 0 0 1
 0 0 0 0 1 0 1 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1
 0 1 1 0]

[7] print(x_test)
[[ 34 115000]
 [ 23 66000]
 [ 56 60000]
 [ 31 118000]
 [ 48 35000]
 [ 47 113000]
 [ 39 79000]
 [ 52 38000]
 [ 24 58000]
 [ 37 53000]
 [ 42 80000]
 [ 46 28000]
 [ 42 73000]
 [ 37 62000]
 [ 68 42000]
 [ 36 52000]
 [ 58 95000]
 [ 43 129000]
 [ 27 89000]
 [ 23 82000]
 [ 38 112000]
 [ 35 50000]
 [ 36 99000]
 [ 37 144000]
 [ 26 35000]
 [ 42 70000]
 [ 43 133000]
 [ 38 50000]
 [ 46 96000]
 [ 35 44000]
 [ 39 113000]]
```

✓ 0s completed at 2:26PM

KUNSH SABHARWAL

Program 5_FML_Prac_File.ipynb

```
File Edit View Insert Runtime Tools Help
q Commands + Code + Text
print(y_test)
[0] [9] from sklearn.preprocessing import StandardScaler
[X] sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

[10] print(x_train)
[0] [-1.11474787e-01 1.68990016e+00]
[-9.67312783e-01 -1.45199398e+00]
[-6.99904247e-01 2.96801165e-01]
[2.14417186e+00 4.15363548e-01]
[-7.97975837e-01 3.26441761e-01]
[5.75026423e-01 1.27494082e+00]
[8.46684727e-02 3.26441761e-01]
[2.04619827e+00 2.22343980e+00]
[-1.11474787e-01 1.68990016e+00]
[1.16345596e+00 -1.21486922e+00]
[-1.87676333e+00 -5.03494919e-01]
[-6.99904247e-01 -1.60019696e+00]
[2.14417186e+00 5.63566526e-01]
[-9.94119817e-01 -3.55291941e-01]
[8.46684727e-02 5.06763904e-02]
[-6.01832657e-01 5.93207122e-01]
[9.67312783e-01 -1.03702564e+00]
[-6.01832657e-01 1.48598186e-01]
[-1.34031173e-02 8.93169951e-02]
[2.80811653e-01 -3.55291941e-01]
[2.14417186e+00 1.88847155e+00]
[1.16345596e+00 -1.18167175e-01]
[1.06538437e+00 6.22847718e-01]
[8.69241193e-01 -1.39271279e+00]
[1.65381391e+00 1.03781606e+00]
[-2.09546297e-01 -4.15199398e+00]
[1.16345596e+00 2.13451810e+00]
[1.06538437e+00 -8.29541472e-01]
[-2.09546297e-01 -4.14573132e-01]
[-1.11474787e-01 -2.66370153e-01]
[-5.03761067e-01 2.40128346e+00]
```

✓ 0s completed at 2:26PM

Program 5_FML_Prac_File.ipynb

```
File Edit View Insert Runtime Tools Help
q Commands + Code + Text
print(x_test)
[0] [-0.30761789 1.36386261]
[-1.38640538 -0.08852658]
[1.84995789 -0.26637015]
[-0.60183266 1.4527844]
[1.06538437 -1.00738585]
[0.96731278 1.30458142]
[0.18274082 0.29688117]
[1.45767073 -0.91846326]
[-1.28833379 -0.32565134]
[-0.01340312 -0.47385432]
[0.47695484 0.32644176]
[0.86924119 -1.21486922]
[0.47695483 0.11895759]
[-0.01340312 -0.26708896]
[2.24224345 -0.79990888]
[-0.11147471 -0.50349492]
[2.04619827 0.77185807]
[0.57502642 1.7783805]
[0.99411982 0.59320712]
[-1.38640538 0.38572295]
[0.08466847 1.27494082]
[-0.2095463 -0.56277611]
[-0.11147471 0.88961308]
[-0.01340312 2.22343989]
[-0.09219861 -1.00738585]
[0.47695484 0.0300358]
[0.57502642 1.89739333]
[0.08466847 -0.56277611]
[0.86924119 0.00069129]
[-0.2095463 -0.74061968]
[0.08466847 1.30458142]
[0.18274082 0.8596764]
[-1.09219861 -0.50349492]
[1.65381391 1.15637844]
[-0.40568944 -0.53313551]
[-1.09219861 -1.57055636]
[-0.69990425 0.53392593]
[-0.2095463 -0.26637015]
[-0.79797584 -1.21486922]
[0.77110986 -1.39271279]
[0.86924119 1.4231438]
[-0.50376107 -1.51127517]
[-1.48447697 0.35608236]
[-1.1902622 0.53392593]
```

✓ 0s completed at 2:26PM

Program 5_FML_Prac_File.ipynb

```
[16]: from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
classifier.fit(x_train, y_train)

[17]: purchased=classifier.predict(sc.transform([[30,87000]]))
print(purchased)
```

[0]

```
y_pred = classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

[1 0]
[0 0]
[1 1]
[1 1]
[1 1]
[1 1]
[0 0]
[1 1]
[0 0]
[0 1]
[1 1]
[0 0]
[1 1]
[0 0]
[0 0]
[1 1]
[0 0]
[1 1]
[0 0]
[1 0]
[0 0]

0s completed at 2:26PM

Program 5_FML_Prac_File.ipynb

```
[0 0]  
[0 0]  
[1 1]  
[1 1]  
[0 1]  
[0 0]  
[0 0]]
```

[x] [18]: from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print("Accuracy: - ",100*accuracy_score(y_test, y_pred), "%")

[[49 9]
 [3 39]]
Accuracy: - 88.0 %

Start coding or generate with AI.

Learning Outcome:

EXPERIMENT 6

Problem statement: Study & Implement Naïve-Bayes.

Theory:

Source Code with Outputs:

Program 6_FML_Prac_File.ipynb

```

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

df=pd.read_csv("Social_Network_Ads (2).csv")
x=df.iloc[:, :-1].values
y=df.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 = 0.25, random_state = 1)

print(x_train)

```

[[[28 49000]
[46 88000]
[31 34000]
[47 30000]
[35 50000]
[39 96000]
[33 113000]
[49 86000]
[45 79000]
[44 39000]
[41 59000]
[42 53000]
[35 73000]
[41 72000]
[27 96000]
[30 116000]
[41 52000]
[41 52000]
[20 82000]
[46 41000]
[27 31000]
[35 71000]
[49 28000]
[35 91000]
[37 75000]
[32 117000]
[36 75000]]

Connected to Python 3 Google Compute Engine backend

Program 6_FML_Prac_File.ipynb

```

print(y_train)

```

[0 1 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 0 1 1 1 0 0
0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1 0 0 1 1 0 0 0 1
1 1 0 0 0 0 0 1 1 0 0 0 1 0 1 0 1 1 0 0 0 0 1 0 0 1 0 0 0 0 1
1 1 0 0 0 0 0 1 0 0 1 1 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 1 0 0 1 0
0 0 0 1 0 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 0 1
1 0 1 0 1 0 0 0 0 0 1 0 1 0 1 1 0 0 1 1 0 0 1 0 1 1 0 0 0 1 0 0 1
0 0 0 1 0 1 0 1 0 1 1 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1
0 1 1 0]

```

print(x_test)

```

[[[36 33000]
[39 61000]
[36 118000]
[39 122000]
[26 118000]
[38 65000]
[28 36000]
[49 89000]
[31 18000]
[48 141000]
[34 72000]
[39 73000]
[35 72000]
[48 131000]
[53 82000]
[56 133000]
[68 83000]
[27 58000]
[28 87000]
[66 102000]
[49 75000]
[50 88000]
[44 139000]
[47 43000]
[45 26000]
[26 15000]
[58 47000]
[49 74000]
[53 34000]
[52 114000]
[39 42000]
[19 76000]]

Connected to Python 3 Google Compute Engine backend

KUNSH SABHARWAL

Program 6_FML_Prac_File.ipynb

```
[8]: print(y_test)
[0 0 1 0 0 0 1 0 0 0 0 1 1 1 0 0 1 0 1 1 0 1 0 1 0 1 0 0 0 0 1 0 0
0 1 0 1 1 0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 1 1 0 0 0 0 1 0 0 0 0 0 1 0
0 1 0 0 0 0 1 1 0 0 0 1 0 1 0 1 1 0 0 1 1 0 0 0 1 0 0 0 0 0 1 0

[9]: from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

[10]: print(x_train)
[[ -1.6862015e+00 -5.92416706e-01
  8.69241193e-01 5.63566526e-01
  -6.0182657e-01 -1.93702564e+00
  9.67312783e-01 -1.15558082e+00
  -2.09546297e-01 -5.62776118e-01
  1.82740863e-01 8.00691292e-01
  4.0569477e-01 1.38458142e+00
  1.16345596e+00 5.04285335e-01
  7.71169683e-01 2.96801165e-01
  6.73998013e-01 -8.88822663e-01
  3.78883243e-01 -2.96010749e-01
  4.76954833e-01 -4.73854323e-01
  -2.09546297e-01 1.18957591e-01
  3.78883243e-01 8.93169951e-02
  -9.94119017e-01 8.00691292e-01
  -6.9994247e-01 1.39350321e+00
  3.78883243e-01 -5.03494919e-01
  3.78883243e-01 -5.03494919e-01
  -1.6862015e+00 3.85722052e-01
  8.69241193e-01 -8.29541472e-01
  -9.94119017e-01 -1.12594743e+00
  -2.09546297e-01 5.96763994e-02
  1.16345596e+00 -1.21869922e+00
  -2.09546297e-01 6.52488313e-01
  -1.34031173e-01 1.78238782e-01
  -5.03761867e-01 1.42314380e+00
  -1.11474787e-01 1.78238782e-01
  -1.6862015e+00 5.04285335e-01
  8.46684727e-02 5.62776118e-01
  1.16345596e+00 -9.77744456e-01
  ...]
```

Connected to Python 3 Google Compute Engine backend

Program 6_FML_Prac_File.ipynb

```
[0]: print(x_test)
[[ -0.11147471 -1.06666624
  0.18274006 -0.23672956
  -0.11147471 1.452784
  0.18274006 1.57134678
  -1.09219061 1.4527844
  0.88466847 -0.11816717
  -1.6862015 -0.97774445
  1.16345596 0.59320712
  -0.60183266 -1.51127517
  0.06538437 2.1345181
  -0.307611789 0.089317
  0.18274006 0.11895759
  -0.2095463 0.089317
  1.06538437 1.83811214
  1.55574232 0.38572295
  1.84995709 1.89739333
  2.24224345 0.41536355
  -0.99411982 -0.32565134
  -0.89604743 0.53392593
  2.24224345 0.97853487
  0.28081165 0.17823878
  1.26152755 0.56356653
  0.67399881 2.07523691
  0.96731278 -0.77026928
  0.7711696 -1.27415841
  -1.09219061 -1.08019696
  2.04618027 -0.6516979
  1.16345596 0.14859819
  1.55574232 -1.03702564
  1.45767873 1.33422281
  0.18274006 -0.79990888
  -1.77869174 0.287897938
  -1.87676333 0.50428533
  1.94882886 0.14859819
  -0.99411982 0.445980414
  -0.69998425 0.32644176
  -1.48447697 -1.51127517
  -0.50376107 0.50428533
  1.26152755 -1.45199398
  -1.77869174 -1.303791
  0.96731278 2.22343989
  2.04618027 0.94889427
  -0.307611789 1.36386261
  -1.38648538 -0.08852658
  1.84995709 -0.266378151
  ...]
```

Connected to Python 3 Google Compute Engine backend

KUNSH SABHARWAL

A screenshot of a Jupyter Notebook interface. The title bar says "Program 6_FML_Prac_File.ipynb". The code cell contains the following Python code:

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train, y_train)

[15] purchased=classifier.predict(sc.transform([[30,87000]]))
print(purchased)

[16] y_pred = classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))

[[0 0]
 [0 0]
 [1 1]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [1 1]
 [1 1]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [1 1]
 [1 1]
 [1 1]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [1 1]
 [1 1]
 [0 0]
 [0 0]
 [1 1]
 [1 1]
 [0 0]
 [0 1]
 [0 1]
 [0 1]
 [0 0]
 [1 1]
 [0 0]]]
```

The output cell shows the predicted values for the test data. A status bar at the bottom right indicates "Connected to Python 3 Google Compute Engine backend".

A screenshot of a Jupyter Notebook interface. The title bar says "Program 6_FML_Prac_File.ipynb". The code cell contains the following Python code:

```
+ Code + Text
```

```
[17] from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print("Accuracy: - ",100*accuracy_score(y_test, y_pred),"%")


[[51  7]
 [ 7 35]]
Accuracy: -  86.0 %
```

The output cell shows the confusion matrix and the calculated accuracy. Two buttons "+ Code" and "+ Text" are visible at the bottom right of the output area.

Learning Outcome:

EXPERIMENT 7

Problem statement: Study & Implement Decision Trees.

Theory:

Source Code with Outputs:

Program 7_FML_Prac_File.ipynb

```

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

df=pd.read_csv("Social_Network_Ads (2).csv")
x=df.iloc[:, :-1].values
y=df.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 = 0.25, random_state = 1)

print(x_train)

```

[5] print(x_train)

```

[[ 20 49000]
 [ 46 88000]
 [ 31 34000]
 [ 47 30000]
 [ 35 50000]
 [ 39 96000]
 [ 33 113000]
 [ 49 66000]
 [ 45 79000]
 [ 44 39000]
 [ 41 59000]
 [ 42 53000]
 [ 35 73000]
 [ 41 72000]
 [ 27 96000]
 [ 38 116000]
 [ 41 52000]
 [ 41 52000]
 [ 28 82000]
 [ 46 41000]
 [ 27 31000]
 [ 35 71000]
 [ 49 28000]
 [ 35 91000]
 [ 37 75000]
 [ 32 117000]
 [ 36 75000]]

```

Connected to Python 3 Google Compute Engine backend

Program 7_FML_Prac_File.ipynb

```

print(y_train)

```

[6] print(y_train)

```

[0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 0 1 1 1 0 0
 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 1 0 1 1 0 0 0 1 1 0 0 1 1 0 1
 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0
 1 1 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 0 1 1 0 1 1 0 0 0 1 0 0 1 0 0 0 0 1
 1 1 0 0 0 0 0 1 0 0 1 1 0 0 0 1 0 1 0 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0 1
 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 1 0 1 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 0 1
 1 0 1 0 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 0 1 1 1 0 0 0 1 0 0 0 1
 0 0 1 0 1 0 1 0 1 1 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1
 0 1 1 0]

```

print(x_test)

```

[[ 36 33000]
 [ 39 61000]
 [ 36 118000]
 [ 39 122000]
 [ 26 118000]
 [ 38 65000]
 [ 28 36000]
 [ 49 89000]
 [ 31 18000]
 [ 48 141000]
 [ 34 72000]
 [ 39 73000]
 [ 35 72000]
 [ 48 131000]
 [ 53 82000]
 [ 56 133000]
 [ 68 83000]
 [ 27 58000]
 [ 28 87000]
 [ 68 102000]
 [ 48 75000]
 [ 58 88000]
 [ 44 139000]
 [ 47 43000]
 [ 45 26000]
 [ 26 15000]
 [ 58 47000]
 [ 49 74000]
 [ 53 34000]
 [ 52 114000]
 [ 39 42000]
 [ 19 76000]]

```

Connected to Python 3 Google Compute Engine backend

KUNSH SABHARWAL

Program 7_FML_Prac_File.ipynb

```
[8]: print(y_test)
[8] [0 1 1 0 0 1 0 0 0 0 1 1 1 0 0 1 0 1 1 0 1 0 1 0 0 0 0 1 0 0
0 1 0 1 1 0 0 1 1 1 1 0 1 0 0 1 1 1 0 1 0 1 1 0 0 0 0 1 1 0 0 0 0 0 1 0
0 1 0 0 0 0 1 1 0 0 0 1 0 1 1 1 0 0 1 0 0]

<>
[X] [9]: from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

[10]: print(x_train)
[10] [[-1.6862815e+00 -5.92416706e-01
       8.69241193e-01 5.63566526e-01
      -6.01832657e-01 -1.03702564e+00
       9.67312783e-01 -1.15558802e+00
      -2.09546297e-01 -5.62776110e-01
       1.82740065e-01 8.06691292e-01
      -4.06589477e-01 1.30458142e+00
       1.16345596e+00 5.04285353e-01
       7.7116963e-01 2.96881165e-01
      -6.73098013e-01 -8.88822663e-01
       3.78883243e-01 -2.96810749e-01
      -4.76954833e-01 -4.73854323e-01
      -2.09546297e-01 1.18957591e-01
       3.78883243e-01 8.93169591e-02
       9.94119017e-01 8.06691292e-01
      -6.99964247e-01 1.39358321e+00
       3.78883243e-01 -5.03494919e-01
       3.78883243e-01 [-5.03494919e-01]
      -1.6862815e+00 3.85722552e-01
       8.69241193e-01 -8.29541472e-01
      -9.94119017e-01 -1.12594743e+00
      -2.09546297e-01 5.06763994e-02
       1.16345596e+00 -1.21486922e+00
      -2.09546297e-01 6.52488313e-01
      -1.34831173e-02 1.78238782e-01
      -5.03761067e-01 1.42314380e+00
      -1.11474707e-01 1.78238782e-01
      -1.6862815e+00 5.04285353e-01
      -8.46684727e-02 -5.62776110e-01
       1.16345596e+00 -9.77744450e-01
       2.80811653e-01 -1.18167175e-01]
```

Connected to Python 3 Google Compute Engine backend

Program 7_FML_Prac_File.ipynb

```
[8]: print(x_test)
[8] [[-0.11147471 -1.06666624]
     [ 0.18274086 -0.23672956]
     [-0.11147471 1.4527844]
     [ 0.18274086 1.57134678]
     [-1.09219061 1.4527844]
     [ 0.08466847 -0.11816717]
     [-1.6862815 -0.97774445]
     [ 1.16345596 0.59328712]
     [-0.68183266 -1.51127517]
     [ 1.06538437 2.1345181]
     [-0.30761789 0.089317]
     [ 0.18274086 0.11895759]
     [-0.2895463 0.089317]
     [ 1.06538437 1.83811214]
     [ 1.55574238 0.38572295]
     [ 1.84995709 1.89739333]
     [ 2.24224345 0.41536355]
     [-0.99411902 -0.32565134]
     [-0.89604743 0.53392393]
     [ 2.24224345 0.97853487]
     [ 0.28981165 0.17823878]
     [ 1.26152755 0.56356653]
     [ 0.67309801 2.07523691]
     [ 0.96731278 -0.7702628]
     [ 0.7711696 -1.27415841]
     [-1.09219061 -1.60819696]
     [ 2.04610827 -0.6516979]
     [ 1.16345596 0.14859819]
     [ 1.55574232 -1.03702564]
     [ 1.45767078 1.33422281]
     [ 0.18274086 -0.79990888]
     [-1.77869174 0.20787938]
     [-1.87676333 0.50428533]
     [ 1.94882868 0.14859819]
     [-0.99411902 0.44580414]
     [-0.69999425 0.32644176]
     [-1.48447697 -1.51127517]
     [-0.58376167 0.50428533]
     [ 1.26152755 -1.45199398]
     [-1.77869174 -1.383791]
     [ 0.96731278 2.22343898]
     [ 2.04610827 0.94889427]
     [-0.30761789 1.36386261]
     [-1.38640538 -0.88852658]]
```

Connected to Python 3 Google Compute Engine backend

```
[12] from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(x_train, y_train)

[13] purchased=classifier.predict(sc.transform([[30,87000]]))
print(purchased)

[14] y_pred = classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))

[[0 0]
 [0 0]
 [1 1]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [1 1]
 [0 1]
 [0 1]
 [0 1]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [1 1]
 [0 1]
 [0 1]
 [1 1]
 [1 0]
 [1 1]
 [0 0]
 [1 1]
 [0 0]
 [1 1]]]
```

Connected to Python 3 Google Compute Engine backend

```
[15] from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print("Accuracy: - ",100*accuracy_score(y_test, y_pred),"%")

[[48 10]
 [ 7 35]]
Accuracy: -  83.0 %
```

[] Start coding or generate with AI.

Learning Outcome:

EXPERIMENT 8

Problem statement: Study & Implement K- Means Clustering to find natural patterns in Data.

Theory:

Dataset:

A	B	C	D	E	F	G
Customer	Genre	Age	Annual Income	Spending Score (1-100)		
1	Male	19	15	39		
2	Male	21	15	81		
3	Female	20	16	6		
4	Female	23	16	77		
5	Female	31	17	40		
6	Female	22	17	76		
7	Female	35	18	6		
8	Female	23	18	94		
9	Male	64	19	3		
10	Female	30	19	72		
11	Male	67	19	14		
12	Female	35	19	99		
13	Female	58	20	15		
14	Female	24	20	77		
15	Male	37	20	13		
16	Male	22	20	79		
17	Female	35	21	35		
18	Male	20	21	66		
19	Male	52	23	29		
20	Female	35	23	98		
21	Male	35	24	35		
22	Male	25	24	73		
23	Female	46	25	5		
24	Male	31	25	73		
25	Female	54	28	14		
26	Male	29	28	82		
27	Female	45	28	32		
28	Male	35	28	61		
29	Female	40	29	31		
30	Female	23	29	87		
31	Male	60	30	4		
32	Male					

Source Code and Output:

Program 8_FML_Prac_File.ipynb

```

File Edit View Insert Runtime Tools Help
Q Commands + Code + Text
RAM Disk
[1] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

[2] df=pd.read_csv("Mall_Customers.csv")
x=df.iloc[:,[3,4]].values

[3] from sklearn.cluster import KMeans
wcss = []
for i in range(1,11):
    kmeans=KMeans(n_clusters=i,init='k-means++',random_state=42)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)

[4] plt.plot(range(1,11),wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

```

The Elbow Method

250000
200000
150000
100000
50000

1 2 3 4 5 6 7 8 9 10

✓ 0s completed at 9:28PM

The Elbow Method

```

[5] kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(x)

[6] plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')

```

✓ 0s completed at 9:28PM

Clusters of customers

```

[6] plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 100, c = 'green', label='Cluster 3')
plt.scatter(x[y_kmeans == 3, 0], x[y_kmeans == 3, 1], s = 100, c = 'cyan', label='Cluster 4')
plt.scatter(x[y_kmeans == 4, 0], x[y_kmeans == 4, 1], s = 100, c = 'magenta', label='Cluster 5')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = '(centroids)')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()

```

Start coding or generate with AI.

✓ 0s completed at 9:28PM

Learning Outcome: