# Suraj Sawant TEB-38

## Practical A5:-Logistic Regression (Data Analytics II)

- 1. Implement logistic regression using Python/R to perform classification on Social\_Network\_Ads.csv dataset.
- 2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

```
# https://www.kaggle.com/datasets/akram24/social-network-ads/data
```

# Dataset on Social media ads describing users, whether users have purchased a product by

import pandas as pd

 $\rightarrow$ 

df = pd.read\_csv("/content/Social\_Network\_Ads - Social\_Network\_Ads.csv")
df.head(10)

		User ID	Gender	Age	EstimatedSalary	Purchased
	0	15624510	Male	19	19000	0
	1	15810944	Male	35	20000	0
	2	15668575	Female	26	43000	0
	3	15603246	Female	27	57000	0
	4	15804002	Male	19	76000	0
	5	15728773	Male	27	58000	0
	6	15598044	Female	27	84000	0
	7	15694829	Female	32	150000	1
	8	15600575	Male	25	33000	0
sha	<b>9</b> ape	15727311	Female	35	65000	0

**(400, 5)** 

df.s

df.info()

0	User ID	400 non-null	int64	
1	Gender	400 non-null	object	
2	Age	400 non-null	int64	
3	EstimatedSalary	400 non-null	int64 4 Purchased	400 non-
	null int64 dt	ypes: int64(4),	<pre>object(1) memory usage:</pre>	15.8+ KB

## df.describe()

-		
_	_	_
_	7	~
•		_

<b>→</b> ▼		User ID	Age	EstimatedSalary	Purchased	
	count	4.000000e+02	400.000000	400.000000	400.000000	
	mean	1.569154e+07	37.655000	69742.500000	0.357500	
	std	7.165832e+04	10.482877	34096.960282	0.479864	
	min	1.556669e+07	18.000000	15000.000000	0.000000	
	25%	1.562676e+07	29.750000	43000.000000	0.000000	
	50%	1.569434e+07	37.000000	70000.000000	0.000000	
	75%	1.575036e+07	46.000000	88000.000000	1.000000	
df.dr	max op(['Us	1.581524e+07 er ID'],axis=3	60.000000 1,inplace=Tr	150000.000000 rue)# Drop the unw	1.000000 wanted column	Use

### df.Purchased.value\_counts()



#### count

#### Purchased

0	257
1	143

dtype: int64

from sklearn.preprocessing import LabelEncoder # Import LabelEncoder label\_encoder = LabelEncoder() # Create an instance of LabelEncoder df['Gender'] = label\_encoder.fit\_transform(df['Gender']) # Male=1, Female=0

df.head()

<b>→</b>	Gender		Age Est	imatedSalary	Purchased
	0	1	19	19000	0
	1	1	35	20000	0
	2	0	26	43000	0
	3	0	27	57000	0
	4	1	19	76000	0
x=df[	['Ger	ider','Age	','Estim	atedSalary']]	

x.head()

# x contains the independent features (Gender, Age, EstimatedSalary)

<b>→</b>		Gender A	Age	EstimatedSalary
	0	1	19	19000
	1	1	35	20000
	2	0	26	43000
	3	0	27	57000
165	4		19	76000

y=df['Purchased']

y.head() # y contains the dependent variable
"Purchased")

<b>→</b> ▼	Pur	chased
	0	0
	1	0
	2	0
	3	0
	4	0

dtype: int64

from sklearn.model\_selection import train\_test\_split xtrain,xtest, ytrain,
ytest=train\_test\_split(x,y,test\_size=0.2,random\_state=42)
#20% of the data is reserved for testing, while 80% is used for training.
# y contains the dependent variable "Purchased")

xtrain.shape

ytrain.shape

ytest.shape

from sklearn.linear\_model import LogisticRegression #import
model=LogisticRegression() #Create model.fit(xtrain,ytrain)
#Train

```
2/28/25 → LogisticRegression i LogisticRegression()
```

ypred=model.predict(xtest) # model is generated for the test dataset

ypred #Predicted Values

ytest #Actual Values

<b>→</b>		Purchased
	209	0
	280	1
	33	0
	210	1
	93	0
	246	0
	227	1
	369	1
	176	0
	289	1

80 rows × 1 columns

dtype: int64

# Measure the Performance of the Model
from sklearn.metrics import accuracy\_score
accuracy=accuracy\_score(ytest, ypred)
accuracy

**₹** 0.8875

from sklearn.metrics import confusion\_matrix
cm=confusion\_matrix(ytest,ypred) cm

from sklearn.metrics import classification\_report

report=classification report(ytest, ypred) print(report)

precision recall f1-score support

0 0.88 0.96 0.92 52 1 0.91 0.75 0.82 28

accuracy 0.89 80 macro avg 0.90 0.86 0.87

80 weighted avg 0.89 0.89 0.88

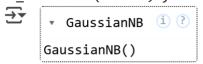
80 # With this Classification Report, Practical No. A-5 is Over

# The Program is extended further just to understand the implementaion of Naive Bayes Cla # with respect to the same example, not to be included for practical exam

#Using Naive Bayes Classifier

from sklearn.naive\_bayes import GaussianNB #algorithm based on Bayes' Theorem.

nb\_model = GaussianNB()# GaussianNB() initializes the Naïve Bayes classifier.
nb\_model.fit(xtrain, ytrain) #it trains model using x train and y train



nb\_ypred = nb\_model.predict(xtest)

nb report = classification report(ytest, nb ypred)

0 0.94 0.96 0.95 52 1 0.93 0.89 0.91 28 0.94 accuracy 80 0.93 0.93 0.93 80 macro avg