Experiment 1

Shashwat Shah
6000422012b
TY BTech Comps B

Aim: To perform data preprocessing in terms of handling missing data, removing outliers, eliminating outliers, eliminating duplicate rows & modifying the datatypes, etc.

Theory: Python an easy to learn programming language which makes it the most popular & preferred language for beginners in data science, data analytics and machine learning. It also has a great community of online learners and excellent data centric libraries. With so much data being generated it becomes important that the data we use for data science explications like machine learning & predictive modelling is clean. Data cleaning refers to the process of cleaning the dirty data by identifying the errors in data and rectifying them. Data cleanup is hence a very important step in ML.

Data cleaning consists of :-

* Missing values

We will start by calculating the percentage of values missing in each column and storing information in the dataset.

* Drop Observation - One way

We can drop observations that contain only null values in them for any of the columns. This works when

the percentage of missing values in each columns is less.

* Remove columns -
Drop columns / features which have significant percentage of missing values.

* Imp missing values -
we can also fill missing value in numerical columns, using mean, median, mode and other such structures.

* Outliers
It's an unusual observation that is random & wrong. They can affect the ML model significantly.

* Duplicate records
Data can sometimes contain duplicate values. Its important duplicate records from dataset before we process to ML model.

* Fixing a datatype
Often values in dataset went stored in the correct data type.

Procedure
Firstly we load the dataset in the dataframe, then we list columns & the no. of null values in that columns. Its observed that age has 177 null values, cabin has 687 null values. Next we fill the null values. We also obeen that the cleared data includes columns age and encoded values of columns embarked & p class as dependant values & survived column as largest.

Conclusion - Here we conclude that data cleaning is very important before applying data on ML models.

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

COURSE NAME: Machine Learning **CLASS:** Third Year BTech

NAME: Shashwat Shah **BATCH:** C22

EXPERIMENT NO. 1

AIM / OBJECTIVE:

To perform data preprocessing in terms of handling, missing data, removing outliers, eliminating duplicate rows and modifying the datatype, etc.

DESCRIPTION OF EXPERIMENT:

Python is an easy-to-learn programming language, which makes it the most preferred choice for beginners in Data Science, Data Analytics, and Machine Learning. It also has a great community of online learners and excellent data-centric libraries. With so much data being generated, it becomes important that the data we use for Data Science applications like Machine Learning and Predictive Modeling is clean. But what do we mean by clean data? And what makes data dirty in the first place? Dirty data simply means data that is erroneous. Duplicacy of records, incomplete or outdated data, and improper parsing can make data dirty. This data needs to be cleaned. Data cleaning (or data cleansing) refers to the process of "cleaning" this dirty data, by identifying errors in the data and then rectifying them. Data cleaning is an important step in and Machine Learning project, and we will cover some basic data cleaning techniques (in Python).

Cleaning Data in Python

We will now separate the numeric columns from the categorical columns.

Missing values

We will start by calculating the percentage of values missing in each column, and then storing this information in a DataFrame.

Drop observations

One way could be to drop those observations that contain any null value in them for any of the columns. This will work when the percentage of missing values in each column is very less.

Remove columns (features)

Another way to tackle missing values in a dataset would be to drop those columns or features that have a significant percentage of values missing.

Impute missing values

There is still missing data left in our dataset. We will now impute the missing values in each numerical column with the median value of that column.

Outliers

An outlier is an unusual observation that lies away from the majority of the data. Outliers can affect the performance of a Machine Learning model significantly.

Duplicate records

Data can sometimes contain duplicate values. It is important to remove duplicate records from your dataset before you proceed with any Machine Learning project. In our data, since the ID column is a unique identifier, we will drop duplicate records by considering all but the ID column.

Fixing data type

Often in the dataset, values are not stored in the correct data type. This can create a problem in later stages, and we may not get the desired output or may get errors while execution.

PROCEDURE:

Describe the procedure that is used to carry out the experiment step-by-step. Describe every line of code with the proper interpretation of the output.

Perform data preprocessing with respect to your case study and discuss results of all the steps.

Code and output:

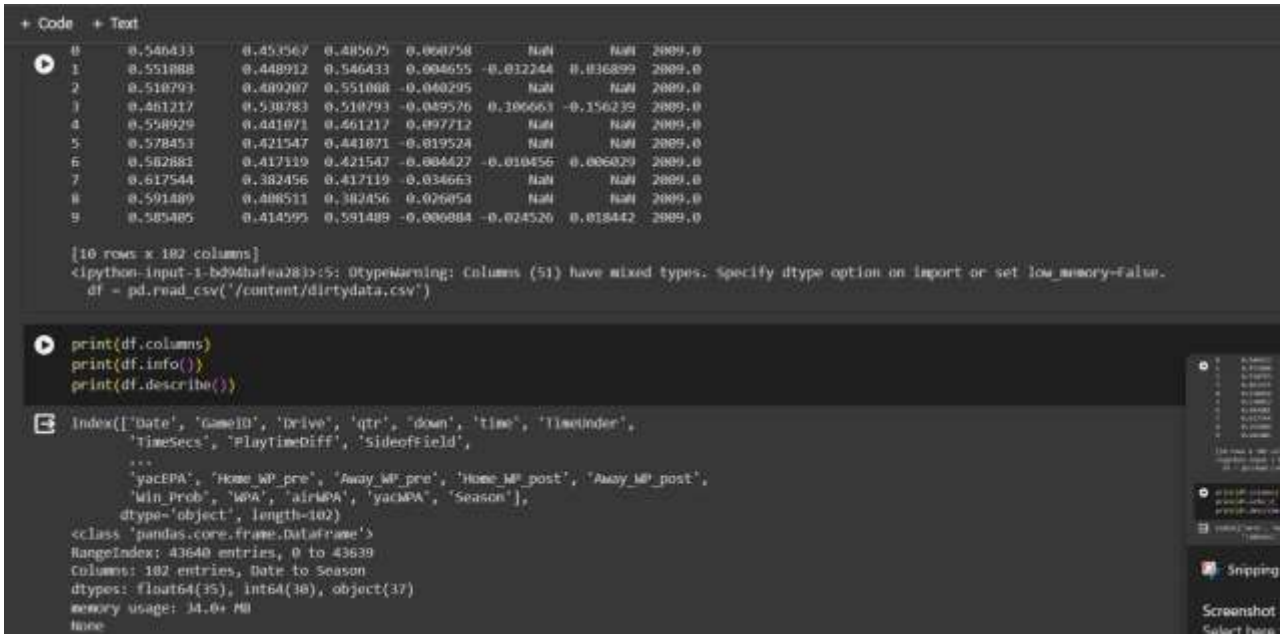
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv('/content/dirtydata.csv')
print(df.head(10))
```

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

	Date	GameID	Drive	qtr	down	time	TimeUnder	TimeSecs	\
0	2009-09-10	2009091000	1	1	NaN	15:00	15	3600.0	
1	2009-09-10	2009091000	1	1	1.0	14:53	15	3593.0	
2	2009-09-10	2009091000	1	1	2.0	14:16	15	3556.0	
3	2009-09-10	2009091000	1	1	3.0	13:35	14	3515.0	
4	2009-09-10	2009091000	1	1	4.0	13:27	14	3507.0	
5	2009-09-10	2009091000	2	1	1.0	13:16	14	3496.0	
6	2009-09-10	2009091000	2	1	2.0	12:40	13	3460.0	
7	2009-09-10	2009091000	2	1	3.0	12:11	13	3431.0	
8	2009-09-10	2009091000	2	1	4.0	11:34	12	3394.0	
9	2009-09-10	2009091000	3	1	1.0	11:24	12	3384.0	

	PlayTimeDiff	SideofField	...	yacEPA	Home_WP_pre	Away_WP_pre	\
0	0.0	TEN	...	NaN	0.485675	0.514325	
1	7.0	PIT	...	1.146076	0.546433	0.453567	
2	37.0	PIT	...	NaN	0.551088	0.448912	
3	41.0	PIT	...	-5.031425	0.510793	0.489207	
4	8.0	PIT	...	NaN	0.461217	0.538783	
5	11.0	TEN	...	NaN	0.558929	0.441071	
6	36.0	TEN	...	0.163935	0.578453	0.421547	
7	29.0	TEN	...	NaN	0.582881	0.417119	
8	37.0	TEN	...	NaN	0.617544	0.382456	
9	10.0	TEN	...	0.541602	0.591489	0.408511	

	Home_WP_post	Away_WP_post	Win_Prob	WPA	airWPA	yacWPA	Season
0	0.546433	0.453567	0.485675	0.060758	NaN	NaN	2009.0
1	0.551088	0.448912	0.546433	0.004655	-0.032244	0.036899	2009.0
2	0.510793	0.489207	0.551088	-0.040295	NaN	NaN	2009.0
3	0.461217	0.538783	0.510793	-0.049576	0.106663	-0.156239	2009.0
4	0.558929	0.441071	0.461217	0.097712	NaN	NaN	2009.0
5	0.578453	0.421547	0.441071	-0.019524	NaN	NaN	2009.0
6	0.582881	0.417119	0.421547	-0.004427	-0.010456	0.006029	2009.0



SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

```
[ ]  count  4.364000e+04  43640.000000  43640.000000  36950.000000  43640.000000
     mean   2.009144e+09     12.335060      2.560151      2.015264      7.334166
     std    1.778114e+05      7.106666      1.126115      1.009782      4.659195
     min    2.009091e+09      1.000000      1.000000      1.000000      0.000000
     25%    2.009101e+09      6.000000      2.000000      1.000000      3.000000
     50%    2.009111e+09     12.000000      2.000000      2.000000      7.000000
     75%    2.009121e+09     18.000000      4.000000      3.000000     11.000000
     max    2.010010e+09     32.000000      5.000000      4.000000     15.000000

              TimeSecs  PlayTimeDiff         yrdln    yrdline100       ydstogo  \
     count  43607.000000  43574.000000  43551.000000  43551.000000  43640.000000
     mean    1708.349554     20.799904     28.381323     47.692292      7.196471
     std     1057.388873     16.910683     13.129457     25.187992      4.796411
     min     -893.000000      0.000000      1.000000      1.000000      0.000000
     25%      803.000000      5.000000     20.000000     30.000000      3.000000
     50%     1800.000000     17.000000     30.000000     49.000000      9.000000
     75%     2597.000000     38.000000     39.000000     69.000000     10.000000
     max     3600.000000    234.000000     50.000000     99.000000     36.000000

           ...         yacEPA   Home_WP_pre   Away_WP_pre  Home_WP_post  \
     count ...  16595.000000  40762.000000  40762.000000  40584.000000
     mean  ...     -0.400900      0.534956      0.465486      0.535202
     std   ...      2.008798      0.289938      0.289991      0.292223
     min   ...    -14.000000      0.000000      0.000000      0.000000
     25%   ...     -0.957404      0.327666      0.221636      0.323830
     50%   ...      0.000000      0.530724      0.469555      0.533272
     75%   ...      0.479230      0.778942      0.672906      0.783487
     max   ...      9.059733      1.000000      1.000000      1.000000

           Away_WP_post      Win_Prob           WPA         airWPA         yacWPA  \
     count  40584.000000  40755.000000  4.296800e+04  16597.000000  16569.000000
     mean       0.465212      0.505817  1.841291e-03      0.014292     -0.010349
```

```
[ ]  #Dropping duplicates
     df.drop_duplicates(inplace = True)
```

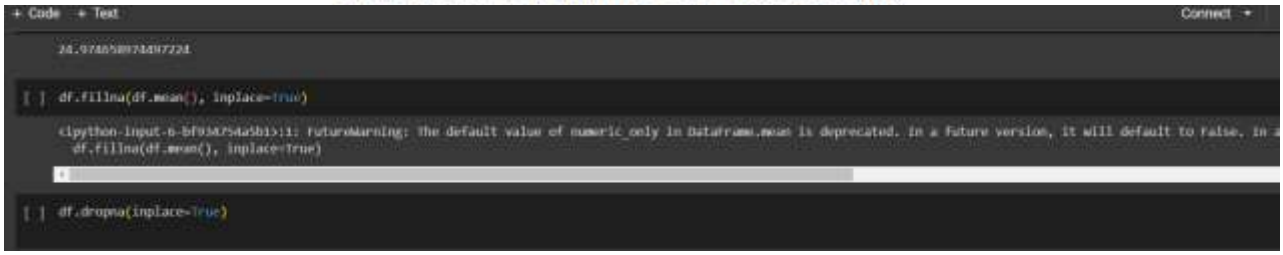
```
[ ]  # get the number of missing data points per column
     missing_values_count = df.isnull().sum()

     # look at the # of missing points in the first ten columns
     missing_values_count[0:10]
```

```
Date            0.0
GameID          0.0
Drive           0.0
qtr             0.0
down            0.0
time            0.0
TimeUnder       0.0
TimeSecs        0.0
PlayTimeDiff    0.0
SideofField     0.0
dtype: float64
```

```
[ ]  total_cells = np.product(df.shape)
     total_missing = missing_values_count.sum()

     # percent of data that is missing
     (total_missing/total_cells) * 100
```

```
24.974658974497224
```

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)



CONCLUSION:

So, we successfully implemented cleaning of data using python

Experiment 2

Snashwat Shah
60004220126
TY Btech Comps B

Aim: To perform linear regression and find errors the model is associated with.

Theory: Linear regression is one of the most popular supervised machine learning algorithms. Its a statistical method of used for predictive analysis. Linear regression makes prediction for continuous /real or numeric variables such as sales, salary, age, etc. Linear regression shows a linear relationship between the variables by giving a sloped straight line.

Mathematically, we can represent a linear regression as $y = b_0 + b_1 x + \varepsilon$

y = Dependant variable

x = Independent variable

b_0 = Intercept of line

b_1 = Linear regression coefficient

ε = random error

The values for x & y variable are training dataset for linear regression model representation.

Procedure:

In the procedure we aim to simply perform Simple Linear regression using the latest least square method without relying on the scikit library. To initiate

the process we start by calculating the mean values X & Y, Next we calculate the deviations of each data point from their respective means. Following this we get the slope (m) of the regression line is determined by diving the sum of squared deviations from the mean of X. Subsequently, the intercept (b) is computed using mean values of X & Y along with the calculated slope. The regression slope equation is formed as $Y = mx + b$, providing for corresponding X values optionally, the results can be varied.

Observed / Discussed result.

The provided python code implemented without external libraries conduct linear regression on a dataset. The linear regression on a dataset. The linear regression calculate slope (m), intercept (b), mean square error (MSE) & R-squared (R^2)

Conclusion:

Hence, linear regression is a very useful predictive machine learning algorithm.

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

Computer Engineering Department

COURSE NAME: Machine Learning **CLASS:** TY Year B.Tech

NAME: Shashwat Shah **BATCH:** C22

EXPERIMENT NO. 2

AIM / OBJECTIVE:

To perform linear regression and find the error associated with the model.

DESCRIPTION OF EXPERIMENT:

Linear regression is one of the easiest and most popular Supervised Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc. Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (x) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.The linear regression model provides a sloped straight line representing the relationship between the variables.Cleaning Data in PythonWe will now separate the numeric columns from the categorical columns.

Mathematically, we can represent a linear regression as: $\mathbf{y= b_0+b_1x+ \varepsilon}$

Here,

y= Dependent Variable (Target Variable)
x= Independent Variable (predictor Variable)
b_0= intercept of the line (Gives an additional degree of freedom)
b_1 = Linear regression coefficient (scale factor to each input value).
ε = random error

The values for x and y variables are training datasets for Linear Regression model representation

The different values for weights or coefficient of lines (b_0, b_1) gives the different line of regression, and the cost function is used to estimate the values of the coefficient for the best fit line. Cost function optimizes the regression coefficients or weights. It measures how a linear regression model is performing. We can use the cost function to find the accuracy of the **mapping function**, which maps the input variable to the output variable. This mapping function is also known as **Hypothesis**

function. For Linear Regression, we use the **Mean Squared Error (MSE)** cost function, which is the average of squared error occurred between the predicted values and actual values. It can be written as:

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i - (b_1 x_i + b_0))^2$$

where,
N=Total number of observation
y_i = Actual value
$(b_1x_i+b_0)$= Predicted value.

Linear regression using Least Square Method

We have linear regression equation as **$y= b_0+b_1x$**

Using least square method,

$$\boldsymbol{b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}}$$

$$\boldsymbol{b_0 = \bar{y} - b_1\bar{x}}$$

PROCEDURE:

1. Describe the procedure that is used to perform Linear regression using Least Square Method carry out the experiment step-by-step for simple linear regression for following dataset without using scikit library. Describe every line of code with the proper interpretation of the output.

X	2	3	4	5	6	7	8	9	10
Y	1	3	6	9	11	13	15	17	20

2. Perform Regression with respect to one dataset of your choice and discuss results of all the steps.

Program

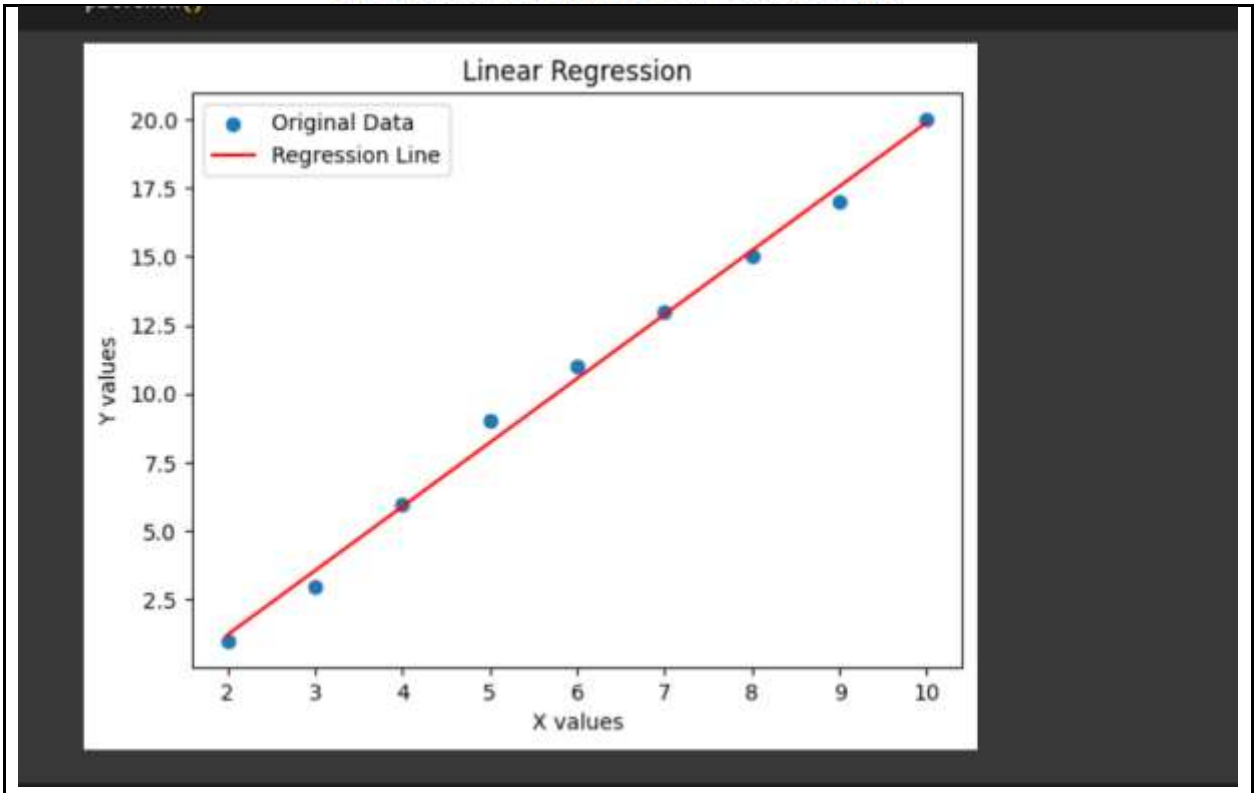
```
import matplotlib.pyplot as plt
```

```
def linear_regression(x_values, y_values):
    n = len(x_values)
    mean_x = sum(x_values) / n
    mean_y = sum(y_values) / n
    numerator = sum((x - mean_x) * (y - mean_y) for x, y in
zip(x_values, y_values))
    denominator = sum((x - mean_x) ** 2 for x in x_values)
    m = numerator / denominator
    b = mean_y - m * mean_x
    y_pred = [m * x + b for x in x_values]
    mse = sum((y - y_pred) ** 2 for y, y_pred in zip(y_values, y_pred))
/ n
    ss_total = sum((y - mean_y) ** 2 for y in y_values)
    ss_residual = sum((y - y_pred) ** 2 for y, y_pred in zip(y_values,
y_pred))
    r_squared = 1 - (ss_residual / ss_total)
    return m, b, mse, r_squared, y_pred


def print_regression_results(slope, intercept, mse, r_squared):
    print(f"Slope (m): {slope}")
    print(f"Intercept (b): {intercept}")
    print(f"Mean Squared Error (MSE): {mse}")
    print(f"R-squared (R²): {r_squared}")


# Example usage:
x_values = [2, 3, 4, 5, 6, 7, 8, 9, 10]
y_values = [1, 3, 6, 9, 11, 13, 15, 17, 20]

slope, intercept, mse, r_squared, y_pred = linear_regression(x_values,
y_values)


print_regression_results(slope, intercept, mse, r_squared)
```

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

```
Slope (m): 2.3333333333333335
Intercept (b): -3.4444444444444446
Mean Squared Error (MSE): 0.17283950617283944
R-squared (R²): 0.995260663507109
```

```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
import matplotlib.pyplot as plt

def linear_regression(x_values, y_values):
    x_values = np.array(x_values).reshape(-1, 1)
    y_values = np.array(y_values)

    model = LinearRegression()
    model.fit(x_values, y_values)
    y_pred = model.predict(x_values)

    mse = mean_squared_error(y_values, y_pred)
    r_squared = r2_score(y_values, y_pred)

    return model.coef_[0], model.intercept_, mse, r_squared, y_pred

x_values = [2,3,4,5,6,7,8,9,10]
y_values = [1,3,6,9,11,13,15,17,20]

slope, intercept, mse, r_squared, y_pred = linear_regression(x_values,
y_values)

print(f"Slope (m): {slope}")
print(f"Intercept (b): {intercept}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R²): {r_squared}")
```




```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


df = pd.read_csv('/content/falguni.csv')

x_values = df["BMI"]
y_values = df["Insurance Cost"]

slope, intercept, mse, r_squared, y_pred = linear_regression(x_values,
y_values)

print(f"Slope (m): {slope}")
print(f"Intercept (b): {intercept}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R²): {r_squared}")
```

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

```
print(f"R-squared (R²): {r_squared}")
```

```
Slope (m): -0.652661473379923
Intercept (b): 0.011702195514829393
Mean Squared Error (MSE): 0.32833188935624363
R-squared (R²): 0.4748942330649337
```

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


df = pd.read_csv('/content/falguni.csv')
x_values = df["BMI"]
y_values = df["Insurance Cost"]

def linear_regression(x_values, y_values):
    n = len(x_values)
    mean_x = sum(x_values) / n
    mean_y = sum(y_values) / n
    numerator = sum((x - mean_x) * (y - mean_y) for x, y in
zip(x_values, y_values))
    denominator = sum((x - mean_x) ** 2 for x in x_values)
    m = numerator / denominator
    b = mean_y - m * mean_x
    y_pred = [m * x + b for x in x_values]
    mse = sum((y - y_pred) ** 2 for y, y_pred in zip(y_values, y_pred))
/ n
    ss_total = sum((y - mean_y) ** 2 for y in y_values)
    ss_residual = sum((y - y_pred) ** 2 for y, y_pred in zip(y_values,
y_pred))
    r_squared = 1 - (ss_residual / ss_total)
    return m, b, mse, r_squared, y_pred
```

```
def print_regression_results(slope, intercept, mse, r_squared):
    print(f"Slope (m): {slope}")
    print(f"Intercept (b): {intercept}")
    print(f"Mean Squared Error (MSE): {mse}")
    print(f"R-squared (R²): {r_squared}")

    slope, intercept, mse, r_squared, y_pred =
linear_regression(x_values, y_values)


print_regression_results(slope, intercept, mse, r_squared)
```

```
# x = 4 predicted with libaries


from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


df = pd.read_csv('/content/datasetcost.csv')

x_values = df["X"]
y_values = df["Y"]
x_to_predict = 4
predicted_y = slope * x_to_predict + intercept

print(f"Predicted value for x = {x_to_predict}: {predicted_y}")

slope, intercept, mse, r_squared, y_pred = linear_regression(x_values,
y_values)

print(f"Slope (m): {slope}")
print(f"Intercept (b): {intercept}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R²): {r_squared}")
```

```
Predicted value for x = 4: 8.448979259357102
Slope (m): 0.6400448894359696
Intercept (b): 5.888799701613223
Mean Squared Error (MSE): 262.2298071449938
R-squared (R²): 0.9213615685311795
```

OBSERVATIONS / DISCUSSION OF RESULT:

1. Find predicted value of y using Linear Regression for one epoch and RMSE for x =4.

X	2	3	4	5	6	7	8	9	10
Y	1	3	6	9	10	13	14	17	21

```
# without libaries pridicted



import pandas as pd

df = pd.read_csv('/content/Linear Regression - Sheet1.csv')
def linear_regression(x_values, y_values):
    n = len(x_values)
    mean_x = sum(x_values) / n
    mean_y = sum(y_values) / n

    numerator = sum((x - mean_x) * (y - mean_y) for x, y in
zip(x_values, y_values))
    denominator = sum((x - mean_x) ** 2 for x in x_values)

    m = numerator / denominator
    b = mean_y - m * mean_x

    y_pred = [m * x + b for x in x_values]
    mse = sum((y - y_pred) ** 2 for y, y_pred in zip(y_values, y_pred))
/ n

    ss_total = sum((y - mean_y) ** 2 for y in y_values)
    ss_residual = sum((y - y_pred) ** 2 for y, y_pred in zip(y_values,
y_pred))
```

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

```
    r_squared = 1 - (ss_residual / ss_total)
    return m, b, mse, r_squared, y_pred

x_values = df["BMI"]
y_values = df["Insurance Cost"]

slope, intercept, mse, r_squared, y_pred = linear_regression(X_values,
Y_values)

print(f"Slope (m): {slope}")
print(f"Intercept (b): {intercept}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R²): {r_squared}")


x_to_predict = 4
predicted_y = slope * x_to_predict + intercept

print(f"Predicted value for x = {x_to_predict}: {predicted_y}")
```

```
Slope (m): 1.4395264828114094
Intercept (b): 3.358000813613444
Mean Squared Error (MSE): 589.7816828135319
R-squared (R²): 0.9213615685311795
Predicted value for x = 4: 9.116106744859081
```

CONCLUSION:

Linear Regression using least squared method was implemented from scratch and using the Libraries

REFERENCES:

(List the references as per format given below and citations to be included the document)

[1] Ponniah P., "Data Warehousing: Fundamentals for IT Professionals", 2nd Edition, Wiley India, 2013.

[2] Ageed, Z. S., Zeebaree, S. R., Sadeeq, M. M., Kak, S. F., Yahia, H. S., Mahmood, M. R., & Ibrahim, I. M. (2021), "Comprehensive survey of big data mining approaches in cloud systems", Qubahan Academic Journal, 1(2), 29-38.

Website References:

Author's Last Name, First Initial. Middle Initial. (Date of Publication or Update). Title of work. Site name. Retrieved Month Day, Year, from URL from Homepage

[3] U.S. Census Bureau. U.S. and world population clock. U.S. Department of Commerce. Retrieved July 3, 2019, from https://www.census.gov/popclock.

Experiment 3

Shashwat Shah
60004220126
TY BTECH C22

Aim: To implement logistic regression

Theory:

Logistic Regression is one of the most popular machine learning algorithms, which comes under the supervised learning technique. It is used for predicting the categorical dependent variable using a given set of independent variable. It predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No. 0 or 1, true or false, etc. but instead of giving the exact value as 0 or 1, it gives the probabilistic values which lies between 0 and 1.

Logistic Regression is like the linear regression except that how they are used. Linear regression is used for solving regression problems. whereas logistic regression is used for solving the classification problems. In logistic regression, instead of fitting a regression line, we fit an 'S' shaped logistic function, which predicts two maximum values (0 or 1). The curve from the logistic function indicates the likelyhood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight.

It is a significant machine learning algorithm because it can provide probabilities and classify new data using continuous and discrete datasets. It can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification.

Conclusion: Hence, we implemented Logistic Regression.

ML EXPERIMENT 3

LOGISTIC REGRESSION

Importing the libraries

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

Importing the dataset

```
[ ] dataset = pd.read_csv('Social_Network_Ads.csv')
    X = dataset.iloc[:,:-1].values
    y = dataset.iloc[:,-1].values
```

Splitting the dataset into the Training set and Test set

```
[ ] 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=0)
```

Feature Scaling

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Training the Logistic Regression model on the Training set

```
[ ] from sklearn.linear_model import LogisticRegression
    classifier = LogisticRegression(random_state=0)
    classifier.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression(random_state=0)
```

Predicting a new result

```
[ ] classifier.predict(sc.transform([[30, 87000]]))
```

```
array([0])
```

Making the Confusion Matrix

```
[ ] from sklearn.metrics import confusion_matrix, accuracy_score
    cm = confusion_matrix(y_test,y_pred)
    print(cm) # Gives us the confusion matrix
    accuracy_score(y_test, y_pred) # Rate of correct prediction
```

```
[[65  3]
 [ 8 24]]
0.89
```

+ Code + Text

Visualising the Training set results

```
[ ] # By this entire code we can see that the curve here is a straight line since logistic regression is a linear classifier
    # We can also see that the demarkation has been made by the color scope showing the results
    from matplotlib.colors import ListedColormap
    X_set, y_set = sc.inverse_transform(X_train), y_train
    X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, step = 0.25),
                         np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step = 0.25))
    plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).reshape(X1.shape),
                 alpha = 0.75, cmap = ListedColormap(('salmon', 'dodgerblue')))
    plt.xlim(X1.min(), X1.max())
    plt.ylim(X2.min(), X2.max())
```

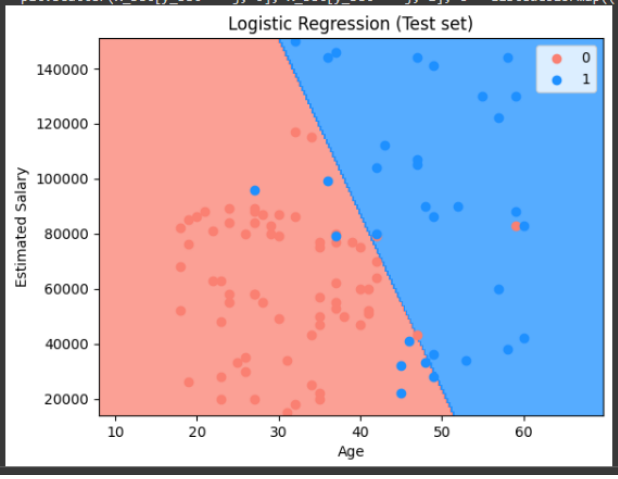
```
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('salmon', 'dodgerblue'))(i), label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
<ipython-input-9-498cf0fd9ad8>:10: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence, which should b
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('salmon', 'dodgerblue'))(i), label = j)
```

Connected to Python 3 Google Compute Engine backend (GPU)

```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_test), y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, step = 0.25),
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step = 0.25))
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('salmon', 'dodgerblue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('salmon', 'dodgerblue'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
<ipython-input-10-96b67a383660>:10: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-map
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('salmon', 'dodgerblue'))(i), label = j)
```



ML EXPERIMENT 3

LOGISTIC REGRESSION

Importing the libraries

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

Importing the dataset

```
[ ] dataset = pd.read_csv('Social_Network_Ads.csv')
    X = dataset.iloc[:,:-1].values
    y = dataset.iloc[:,-1].values
```

Splitting the dataset into the Training set and Test set

```
[ ] 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=0)
```

Feature Scaling

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Training the Logistic Regression model on the Training set

```
[ ] from sklearn.linear_model import LogisticRegression
    classifier = LogisticRegression(random_state=0)
    classifier.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression(random_state=0)
```

Predicting a new result

```
[ ] classifier.predict(sc.transform([[30, 87000]]))
```

```
array([0])
```

Making the Confusion Matrix

```
[ ] from sklearn.metrics import confusion_matrix, accuracy_score
    cm = confusion_matrix(y_test,y_pred)
    print(cm) # Gives us the confusion matrix
    accuracy_score(y_test, y_pred) # Rate of correct prediction
```

```
[[65  3]
 [ 8 24]]
0.89
```

+ Code + Text

Visualising the Training set results

```
[ ] # By this entire code we can see that the curve here is a straight line since logistic regression is a linear classifier
    # We can also see that the demarkation has been made by the color scope showing the results
    from matplotlib.colors import ListedColormap
    X_set, y_set = sc.inverse_transform(X_train), y_train
    X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, step = 0.25),
                         np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step = 0.25))
    plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).reshape(X1.shape),
                 alpha = 0.75, cmap = ListedColormap(('salmon', 'dodgerblue')))
    plt.xlim(X1.min(), X1.max())
    plt.ylim(X2.min(), X2.max())
```

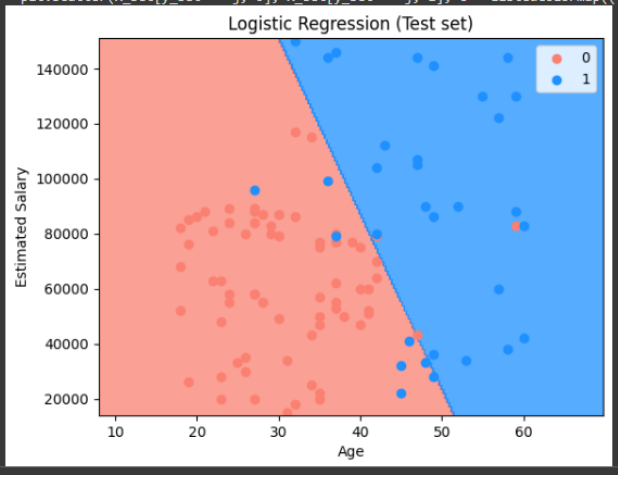
```
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('salmon', 'dodgerblue'))(i), label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
<ipython-input-9-498cf0fd9ad8>:10: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence, which should b
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('salmon', 'dodgerblue'))(i), label = j)
```

Connected to Python 3 Google Compute Engine backend (GPU)

```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_test), y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, step = 0.25),
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step = 0.25))
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('salmon', 'dodgerblue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('salmon', 'dodgerblue'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
<ipython-input-10-96b67a383660>:10: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-map
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('salmon', 'dodgerblue'))(i), label = j)
```



NAME: Shashwat Shah

SAPID:60004220126

C22

B DIV

CASE 1: FROM SCRATCH ON GIVEN DATASET

```python
import numpy as np

class PCA:

    def __init__(self, num_components):
        self.num_components = num_components
        self.components     = None
        self.mean           = None
        self.variance_share = None


    def fit(self, X):

        # 1. Centered Data
        print("Centered Data:")
        self.mean = np.mean(X, axis=0)
        centered_data = X - self.mean
        print(centered_data)

        # data centering
        X = X.astype(np.float64)
        X -= self.mean

        # calculate eigenvalues & vectors
        cov_matrix= np.cov(X.T)

        # 2. Covariance Matrix
        print("\nCovariance Matrix:")
        print(cov_matrix)

        values, vectors = np.linalg.eig(cov_matrix)

        # 3. Eigen Values
        print("\nEigen Values:")
        print(values)

        # 4. Eigen Vectors
        print("\nEigen Vectors:")
        print(vectors)
```

```
        # sort eigenvalues & vectors
        sort_idx = np.argsort(values)[::-1]
        values   = values[sort_idx]
        vectors  = vectors[:, sort_idx]

        # store principal components & variance
        self.components = vectors[:self.num_components]
        self.variance_share = np.sum(values[:self.num_components]) / np.sum(values)

        # 5. New Values (Principal Components)
        print("\nNew Values (Principal Components):")
        print(self.components)


    def transform(self, X):

        # data centering
        X -= self.mean

        # decomposition
        return np.dot(X, self.components.T)

X = np.array([[4, 6],
              [8, 2],
              [13, 3],
              [7, 15]])

# Instantiate PCA with 2 components and fit to data
pca = PCA(num_components=2)
pca.fit(X)
```

OUTPUT:

```
Centered Data:
[[-4.  -0.5]
 [ 0.  -4.5]
 [ 5.  -3.5]
 [-1.   8.5]]

Covariance Matrix:
[[14. -8.]
 [-8. 35.]]

Eigen Values:
[11.29962122 37.70037878]

Eigen Vectors:
[[-0.94747869  0.31981892]
 [-0.31981892 -0.94747869]]

New Values (Principal Components):
[[ 0.31981892 -0.94747869]
 [-0.94747869 -0.31981892]]
```

CASE 2 : DATASET OF YOUR CHOICE USING LIBRARIES

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Load the dataset
df = pd.read_csv("Automobile_data.csv")

# Drop rows with missing values if any
df = df.dropna()

# Select numerical features for PCA
numerical_features = df.select_dtypes(include=['float64', 'int64'])

# Preprocess the data by scaling numerical features
scaler = StandardScaler()
scaled_data = scaler.fit_transform(numerical_features)

# 1. Centered Data
print("Centered Data:")
centered_data = scaled_data - scaled_data.mean(axis=0)
print(centered_data)

# Calculate covariance matrix
cov_matrix = np.cov(centered_data.T)

# 2. Covariance Matrix
print("\nCovariance Matrix:")
print(cov_matrix)

# Compute eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)

# 3. Eigen Values
print("\nEigen Values:")
print(eigenvalues)
```

```
# 3. Eigen Values
print("\nEigen Values:")
print(eigenvalues)

# 4. Eigen Vectors
print("\nEigen Vectors:")
print(eigenvectors)

# Apply PCA
pca = PCA()
pca.fit(scaled_data)

# 5. New Values (Principal Components)
print("\nNew Values (Principal Components):")
print(pca.components_)
```

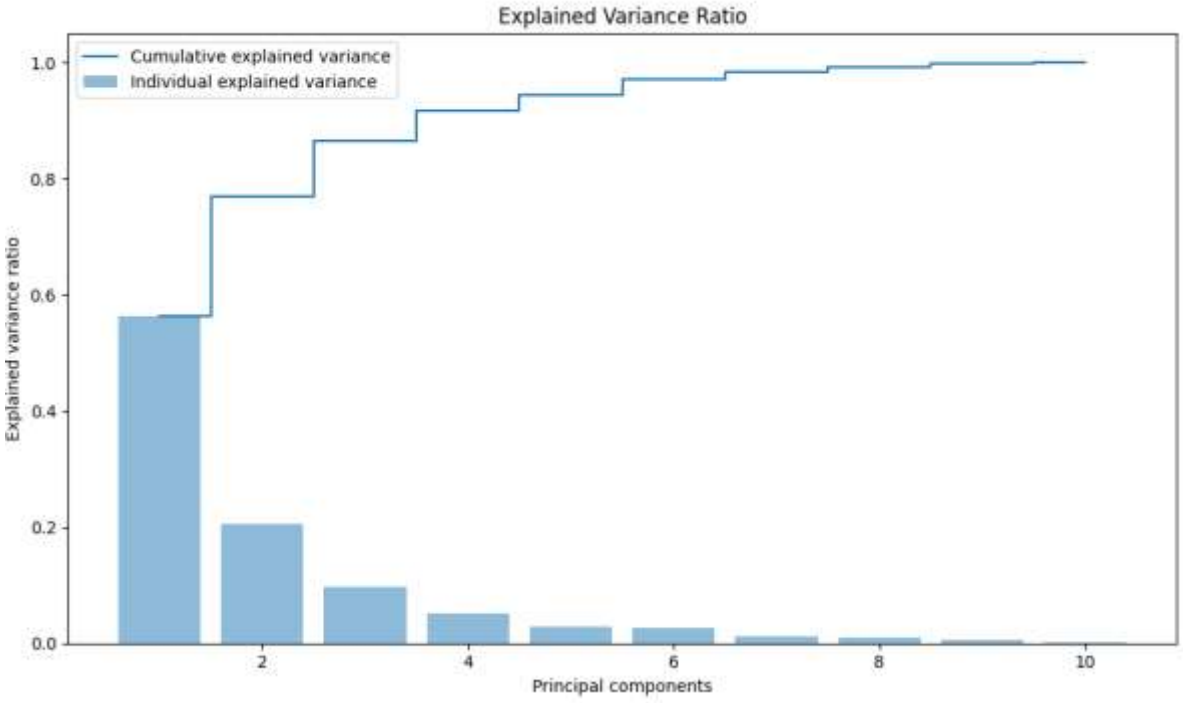
OUTPUT:

```
Centered Data:
[[ 1.74347043 -1.6907718  -0.42652147 ... -0.28834891 -0.64655303
  -0.54605874]
 [ 1.74347043 -1.6907718  -0.42652147 ... -0.28834891 -0.64655303
  -0.54605874]
 [ 0.133509   -0.70859588 -0.23151305 ... -0.28834891 -0.95301169
  -0.69162706]
 ...
 [-1.47645244  1.72187336  1.19854871 ... -0.33882413 -1.10624102
  -1.12833203]
 [-1.47645244  1.72187336  1.19854871 ...  3.24491627  0.11959362
  -0.54605874]
 [-1.47645244  1.72187336  1.19854871 ... -0.16216087 -0.95301169
  -0.83719538]]

Covariance Matrix:
[[ 1.00490196 -0.5345613  -0.35936452 -0.23406082 -0.54369035 -0.22880672
  -0.10630829 -0.17939016 -0.03599823  0.03477564]
 [-0.5345613   1.00490196  0.87887467  0.79904141  0.59232415  0.78019214
   0.57211951  0.25101029 -0.47271956 -0.54674899]
 [-0.35936452  0.87887467  1.00490196  0.8452414   0.49343646  0.88203105
   0.68670968  0.15919024 -0.67419743 -0.70811583]
 [-0.23406082  0.79904141  0.8452414   1.00490196  0.280579    0.87128262
   0.73903847  0.18201651 -0.64585485 -0.68053761]
 [-0.54369035  0.59232415  0.49343646  0.280579    1.00490196  0.29702061
   0.0674779   0.26249469 -0.04887806 -0.10788389]
 [-0.22880672  0.78019214  0.88203105  0.87128262  0.29702061  1.00490196
   0.85476365  0.15210371 -0.7611266  -0.80137393]
 [-0.10630829  0.57211951  0.68670968  0.73903847  0.0674779   0.85476365
   1.00490196  0.02911338 -0.65686212 -0.68079084]
 [-0.17939016  0.25101029  0.15919024  0.18201651  0.26249469  0.15210371
   0.02911338  1.00490196  0.3262931   0.2665014 ]
 [-0.03599823 -0.47271956 -0.67419743 -0.64585485 -0.04887806 -0.7611266
  -0.65686212  0.3262931   1.00490196  0.9760985 ]
 [ 0.03477564 -0.54674899 -0.70811583 -0.68053761 -0.10788389 -0.80137393
  -0.68079084  0.2665014   0.9760985   1.00490196]]
```

```
Eigen Values:
[5.66327865 2.06393304 0.97359585 0.51023591 0.28809477 0.25971235
 0.12508537 0.09029827 0.02196028 0.05282511]

Eigen Vectors:
[[-0.1406803   0.46938248  0.43416281 -0.65753407  0.26084403 -0.20958733
   0.12454723 -0.10570576  0.01452664  0.01332464]
 [ 0.36466184 -0.26203402 -0.0569007  -0.00350329 -0.03796216 -0.41729053
   0.44102192 -0.6343559  -0.13509842 -0.06616736]
 [ 0.3965888  -0.09774234 -0.00607855 -0.13489422 -0.02988667 -0.19022404
   0.47207053  0.71099134  0.14661516 -0.16732211]
 [ 0.38057618 -0.01365742  0.20388087  0.05735174 -0.03937346 -0.55393849
  -0.69811788  0.08754105  0.03470707 -0.07405338]
 [ 0.17202759 -0.47948861 -0.35515743 -0.58511996  0.38055304  0.23974043
  -0.26151004 -0.02579491 -0.00931975 -0.02364729]
 [ 0.40393127  0.0424753   0.16504009  0.0425094   0.06392698  0.17204012
   0.03911469  0.03008734 -0.05454601  0.87572551]
 [ 0.34194007  0.15915237  0.24957784  0.37866195  0.65791046  0.31137929
   0.02860932 -0.08200702 -0.00296087 -0.33870031]
 [ 0.03509208 -0.45916395  0.70716925 -0.11882868 -0.35717351  0.3417246
  -0.01595892 -0.04417378 -0.02988909 -0.16211238]]
```

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
explained_variance_ratio = pca.explained_variance_ratio_
cumulative_variance_ratio = np.cumsum(explained_variance_ratio)
plt.bar(range(1, len(explained_variance_ratio) + 1), explained_variance_ratio, alpha=0.5, align='center', label='Individual explained variance')
plt.step(range(1, len(cumulative_variance_ratio) + 1), cumulative_variance_ratio, where='mid', label='Cumulative explained variance')
plt.xlabel('Principal components')
plt.ylabel('Explained variance ratio')
plt.title('Explained Variance Ratio')
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```

OUTPUT:



NAME : Shashwat Shah

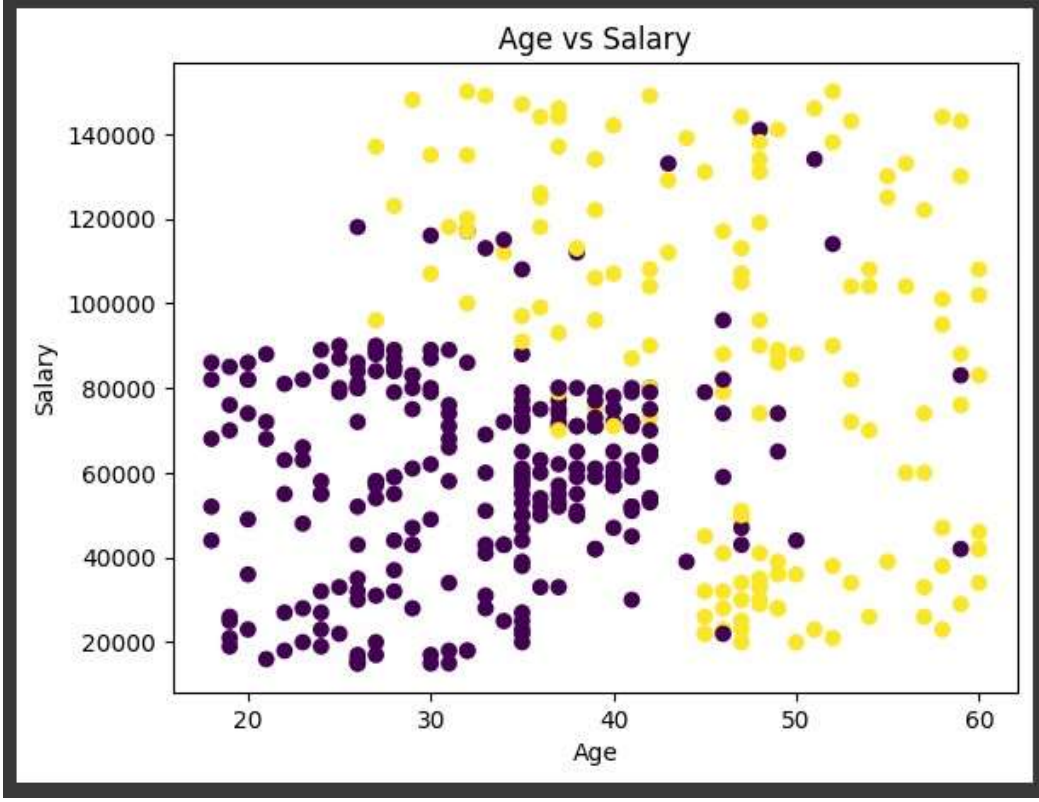
SAPID:60004220126

C22

B DIV

LINEAR DATASET (USER DATA)

```
import pandas as pd
import matplotlib.pyplot as plt

# Read the CSV file
df = pd.read_csv('User_Data.csv')

# Create a scatter plot of estimated age and salary, colored by purchased
plt.scatter(df['Age'], df['EstimatedSalary'], c=df['Purchased'])
plt.xlabel('Age')
plt.ylabel('Salary')
plt.title('Age vs Salary')
plt.show()
```



```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
df = df.dropna()
# Assuming the target variable is in the last column
X = df.iloc[:, :-1]  # Features
y = df.iloc[:, -1]   # Target variable
X = pd.get_dummies(X)

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define kernel types
kernel_types = ['linear', 'poly', 'rbf', 'sigmoid']

for kernel in kernel_types:
    # Train SVM model
    svm_model = SVC(kernel=kernel)
    svm_model.fit(X_train, y_train)

    # Make predictions
    y_pred = svm_model.predict(X_test)

    # Calculate metrics
    accuracy = accuracy_score(y_test, y_pred)
    confusion_mat = confusion_matrix(y_test, y_pred)
    class_report = classification_report(y_test, y_pred)

    # Print results
    print(f"\nKernel: {kernel}")
    print("Accuracy:", accuracy)
    print("Confusion Matrix:")
    print(confusion_mat)
    print("Classification Report:")
    print(class_report)
```

```
Kernel: linear
Accuracy: 0.7375
Confusion Matrix:
[[48  4]
 [17 11]]
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.92      0.82        52
           1       0.73      0.39      0.51        28

    accuracy                           0.74        80
   macro avg       0.74      0.66      0.67        80
weighted avg       0.74      0.74      0.71        80
```

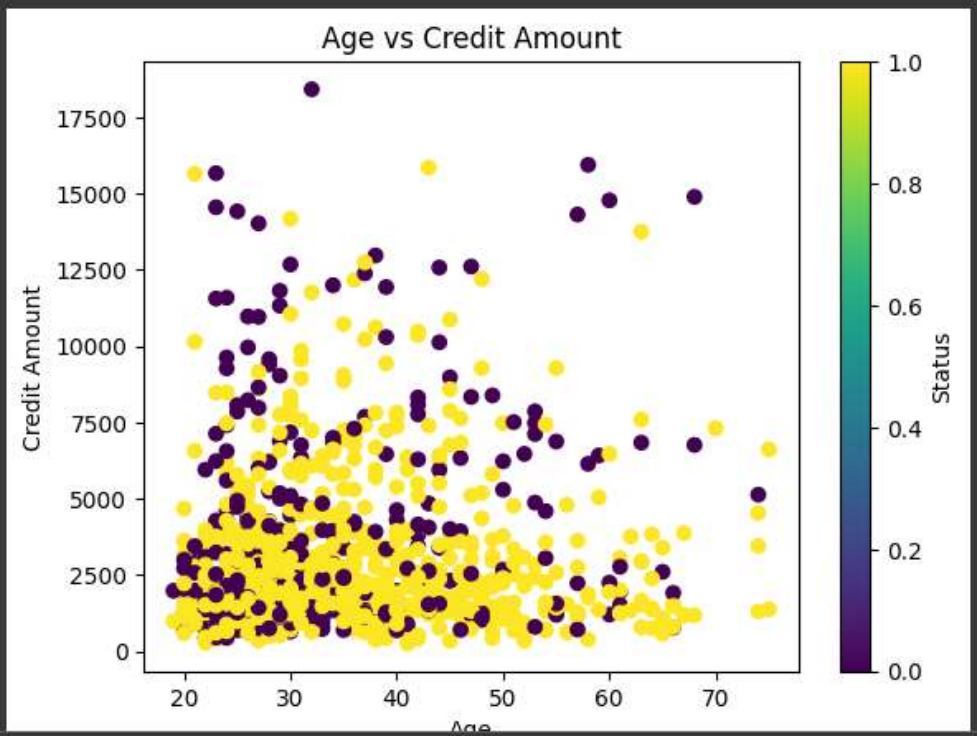
```
Kernel: poly
Accuracy: 0.65
Confusion Matrix:
[[52  0]
 [28  0]]
Classification Report:
              precision    recall  f1-score   support

           0       0.65      1.00      0.79        52
           1       0.00      0.00      0.00        28

    accuracy                           0.65        80
   macro avg       0.33      0.50      0.39        80
weighted avg       0.42      0.65      0.51        80
```

```
Kernel: rbf
Accuracy: 0.65
Confusion Matrix:
[[52  0]
 [28  0]]
Classification Report:
              precision    recall  f1-score   support

           0       0.65      1.00      0.79        52
           1       0.00      0.00      0.00        28

    accuracy                           0.65        80
   macro avg       0.33      0.50      0.39        80
weighted avg       0.42      0.65      0.51        80
```

```
Kernel: sigmoid
Accuracy: 0.65
Confusion Matrix:
[[52  0]
 [28  0]]
Classification Report:
              precision    recall  f1-score   support

           0       0.65      1.00      0.79        52
           1       0.00      0.00      0.00        28

    accuracy                           0.65        80
   macro avg       0.33      0.50      0.39        80
weighted avg       0.42      0.65      0.51        80
```

Non linear dataset(german_csv_file)

```
[9] import pandas as pd
    import matplotlib.pyplot as plt

    # Read the CSV file
    df = pd.read_csv('/content/GermanCredit.csv')

    # Create a scatter plot of age and credit amount, colored by status
    plt.scatter(df['age'], df['amount'], c=df['credit_risk'])
    plt.xlabel('Age')
    plt.ylabel('Credit Amount')
    plt.title('Age vs Credit Amount')
    plt.colorbar(label='Status')
    plt.show()
```



```python
import pandas as pd
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Read the CSV file
df = pd.read_csv('/content/GermanCredit.csv')

# Splitting the data into features (X) and target variable (y)
X = df[['age', 'amount']]
y = df['credit_risk']

# Splitting the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# List of kernel types
kernels = ['linear', 'poly', 'rbf', 'sigmoid']

# Loop through each kernel type
for kernel in kernels:
    print(f"Kernel: {kernel}")

    # Create a support vector machine classifier
    clf = SVC(kernel=kernel)

    # Fit the classifier to the training data
    clf.fit(X_train, y_train)

    # Predict the labels of the test data
    y_pred = clf.predict(X_test)

    # Print accuracy
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy: {accuracy:.2f}")
```

```
# Splitting the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# List of kernel types
kernels = ['linear', 'poly', 'rbf', 'sigmoid']

# Loop through each kernel type
for kernel in kernels:
    print(f"Kernel: {kernel}")

    # Create a support vector machine classifier
    clf = SVC(kernel=kernel)

    # Fit the classifier to the training data
    clf.fit(X_train, y_train)

    # Predict the labels of the test data
    y_pred = clf.predict(X_test)

    # Print accuracy
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy: {accuracy:.2f}")

    # Print confusion matrix
    print("Confusion Matrix:")
    print(confusion_matrix(y_test, y_pred))

    # Print classification report
    print("Classification Report:")
    print(classification_report(y_test, y_pred))
    print()
```

```
Kernel: linear
Accuracy: 0.67
Confusion Matrix:
[[  3  56]
 [ 10 131]]
Classification Report:
              precision    recall  f1-score   support

           0       0.23      0.05      0.08        59
           1       0.70      0.93      0.80       141

    accuracy                           0.67       200
   macro avg       0.47      0.49      0.44       200
weighted avg       0.56      0.67      0.59       200
```

```
Kernel: poly
Accuracy: 0.72
Confusion Matrix:
[[  3  56]
 [  0 141]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.05      0.10        59
           1       0.72      1.00      0.83       141

    accuracy                           0.72       200
   macro avg       0.86      0.53      0.47       200
weighted avg       0.80      0.72      0.62       200
```

```
Kernel: rbf
Accuracy: 0.71
Confusion Matrix:
[[  3  56]
 [  1 140]]
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.05      0.10        59
           1       0.71      0.99      0.83       141

    accuracy                           0.71       200
   macro avg       0.73      0.52      0.46       200
weighted avg       0.72      0.71      0.61       200
```

```
Kernel: sigmoid
Accuracy: 0.66
Confusion Matrix:
[[ 22  37]
 [ 32 109]]
Classification Report:
              precision    recall  f1-score   support

           0       0.41      0.37      0.39        59
           1       0.75      0.77      0.76       141

    accuracy                           0.66       200
   macro avg       0.58      0.57      0.57       200
weighted avg       0.65      0.66      0.65       200
```

A34 M2

Assignment 1

Shashwat Shah
60004220126
TYBtech Comps B

1 No, k means and gaussian mixture models (GMM) will generally not produce same cluster centers (means) for a given dataset.

Reasoning →

k-means: This is a centroid-based clustering algorithm. It works by initially placing a fixed no of centroids at random locations. Then it iteratively assigns each data point to the closest centroid and recomputes the centroid locations as the means of the point assigned to it.

Gaussian Mixture Model- This is a probabilistic clustering method that assumes the data is generated by a mixture of gaussian distributions. It uses expectation maximization (EM) algorithm to find the optimal parameters for the model, including the means and variances of the gaussian distributed that best fit the dates

Due to this fundamental differences, k-means and GMM will often produce different clustering results even with the same no. of cluster specified.

In conclusion, while both k-means and GMM are clustering algorithms, they make different assumptions about the data and produce different cluster representations. The choice of algorithms depends on the characteristics of your data and the desired outcome of your clustering tasks.

Hidden markov model are versatile tool with numerous real-time applications due to their ability to model sequential data with hidden states.

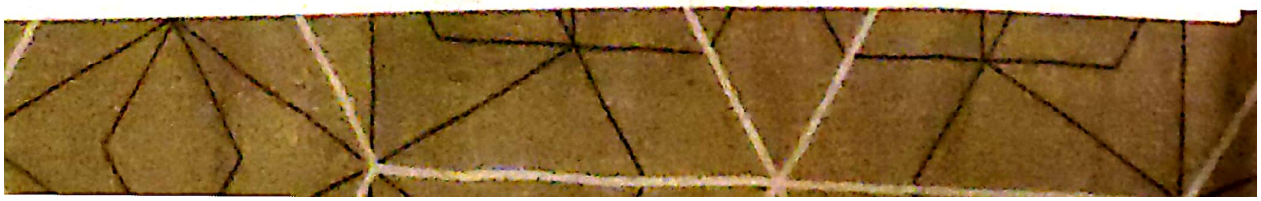
Speech recognition - They can model the statistical properties of phonemes, words or sentences, allowing them to recognise spoken language patterns in real time.

Gesture Recognition - In ~~add~~ applications where gestures are used as input (eg. Sign language, interpretation, human computer interaction), HMM, can be used to model and recognise different gestures based on observed movements.

Bioinformatics: HMMs are used for tasks such as gene prediction protein structure prediction and sequence alignment they can model biological sequences and their hidden structures, aiding in the understanding of genetic information.

Financial time series Analysis - They can be used to model and predict trends in financial data, such as stock prices or market indices by capturing hidden states. that represent market conditions or investor behavior

Robotics and autonomous systems: They are used in robotics for localization and mapping tasks. They can model the robots environment and its movements allowing for real-time navigation and path planning.



Independent component analysis is a method used to separate mixed signals into their original independent component. It assumes that the observed data is combined of three independent sources each with its own unique distribution. By finding their linear transformation that maximizes the statistical independence of the components, ICA can effectively extract useful information from complex data.

The fast ICA algorithm is commonly in various fields including image and signal processing, where the ability to isolate individual sources from mixed signals is crucial.

Deep neural networks are a class of ANN that are composed of multiple layers of nodes which attempt to model high level abstraction in data.

In the context of unsupervised learning, DNN are used to learn the underlying structure of the input data without explicit supervision.

DNN consist of an input layer, multiple hidden layer and an output layer

Each layer contain a set of nodes that perform computations on input data.

Application

- Image recognition
- Speech recognition
- Natural Language Processing
- Anomaly Detection.

Assignment 2

Shashwat Shah
60004220126
TY BTech Comps B

1) a) Video Surveillance :- Support Vector Machine
Beyond anomaly detection, SVMs can also be used for object recognition in video. SVM's learn a hyperplane decision boundary in a high dimensional feature space extracted from index frames. This hyperplane effectively seperate normal activities, patterns from anomalies. Kernal functions are used to map the original features. Potentially low-dimensional into a higher dimensional space where better seperation between normal & anomalous activities by achieved.

When a new frame is captured, the SVM classifies object or tracks specific items, thus enabling alert for unauthorised objects

Challenges - Fine tuning the SVM kernel function for optimal performance and handling variations in lighting and camera angles can be complex.

Applications - Traffic surveillance
Crowd Management
Security monitoring.

b) Sentiment analysis : Naive Bayes.
It calculates the conditional probability of a text review belonging to a specific sentiment class (positive, negative or neutral) given the presence of individual words, Bayes theorem is used to combine these probabilities for

words in the review to find the most likely sentiment class.
This technique can be enhanced by incorporating features beyond just word presence techniques like n-gram or sentiment (word list with associated sentiment scores) can improve accuracy.
Challenges.
Naive Bayes assumes independence of features which may not always hold true in language. Additionally handling sarcasms and complex emotions can be different.
Application
Brand Reputation
Customer Support
Market Research.

c) Image Recognition:- K-Nearest Neighbours (KNN)
KNN relies on the similarity of new images features to those of labelled images in the training set. A distance metrics like Euclidean distance is used to measure this similarity.
Common features extraction techniques for images including color histograms, edge detection and texture analysis which capture the visual properties of the image content.
KNN can be computationally expensive for large datasets and the choice of K significantly improves performance.
Applications:
Medical Imagery

Autonomous Vehicles
Quantify Context

d) Recommender Systems - Collaborative filtering

CF algorithms analyze users - items interaction data to classify patterns & relationships. Thus data is technically represented in a user item matrix, where rows represent users & columns represent items. The matrix entries contain values & represent the level of interaction.

This technique is a popular model based CF approach. The model learn these factors from the user - item interactions data and user. Then to predict user ratings for unseen items. This method can handle space data while users haven't interacted with many items.

Challenges:

Cold starts problems (recommending to new user or items) can be challenging additionally CF techniques are susceptable to biases present in the data.

Applications:

Movie Recommendation
E - commerce
Music Streaming

In conclusion, all these applications are considered as powerful application of ML despite they challenges they play vital role in security communication & personalization in various industries.

ML MINI PROJECT

NAME: Priyal Kamdar (60004210111)
Shashwat Shah (60004220126)
Falguni Parmar (60004220130)

TITLE :

A data-driven approach to predict the success of bank telemarketing.

KEYWORDS :

Bank telemarketing, Predictive analytics, Random Forest, Features selection, SVM, Classification , KNN.

ABSTRACT :

This research paper presents the development and evaluation of a machine learning model designed to predict the effectiveness of bank marketing campaigns using a dataset obtained from a Portuguese banking institution's direct marketing efforts. The model incorporates various customer demographic and behavioral attributes to make predictions, employing multiple algorithms and evaluating its performance through metrics including accuracy, F1-score, precision, recall, and roc_auc_score. The study provides insights into the efficacy of different machine learning techniques in optimizing marketing strategies for financial institutions, contributing to the advancement of data-driven decision-making in the banking sector.

INTRODUCTION :

Classification and prediction algorithms serve as the cornerstone of predictive analytics, offering a systematic approach to uncovering hidden patterns and trends in data. Decision Trees, for instance, employ a hierarchical structure to recursively partition the data based on different attributes, making them well-suited for both classification and regression tasks. Similarly, k-Nearest Neighbors (kNN) relies on the principle of proximity, where data points are classified based on the majority vote of their nearest neighbors. This algorithm is particularly effective for handling noisy data and does not require explicit model training, making it suitable for real-time applications.

On the other hand, Support Vector Machines (SVM) excel in finding the optimal hyperplane that separates data points into different classes with the maximum margin. SVMs are highly versatile and can handle both linear and nonlinear relationships, making them a preferred choice for complex classification tasks. Additionally, Random Forest (RF) algorithms utilize an ensemble of decision trees to improve prediction accuracy and mitigate overfitting. By aggregating the

predictions of multiple trees, Random Forests offer robust performance and can handle high-dimensional datasets efficiently. Overall, the diverse array of classification and prediction algorithms in machine learning empowers data scientists and analysts to extract valuable insights and drive informed decision-making in a wide range of applications.

This project focuses on utilizing machine learning techniques to predict the effectiveness of bank marketing campaigns. The data used in the project is provided by a Portuguese banking institution and includes input variables such as age, job, marital status, education, and balance etc. The goal of this project is to develop a classification model that can accurately predict the effectiveness of bank marketing campaigns. Through the use of machine learning algorithms and techniques, the model will be able to classify a client's response to a campaign as either positive or negative.

This project will provide insight into how different input variables can affect the effectiveness of bank marketing campaigns and help banks better target their customers. The data set contained details about bank marketing campaigns. Descriptive statistics were computed for each variable as part of the analysis, and visualizations were made to investigate the relationships between the various variables. We created a number of graphs, such as a distplot, count plot, bar plot, pair plot, heatmap, and boxplot, to gain insight from the dataset. There are 45211 observations in this dataset and 17 columns with the following names: age, job, marital, education, default, balance, housing, loan, contact, day, month, duration, campaign, pdays, previous, poutcome, and y (target variable).

There are no duplicate values in this dataset. The 10 categorical variables in this dataset are: employment, marital, education, default, housing, loan, contact, month, poutcome, and y. This dataset contains seven numerical variables: age, balance, day, duration, campaign, pdays, and prior. For the variables job, education, contact, and poutcome, the number of unknown tagged values are 288; 1857; 13020; and 36959, respectively. Unknown tagged values can be treated as null since they are not defined and can be taken out of features by treatment. Replaced null values with their respective modes for features like contact, education, and job. Moreover, features with more than 50% null values were eliminated because they were useless and negatively impacted model performance. Outliers are treated using the interquartile range for the variables age, balance, duration, campaign, p-days, and previous.

This project utilizes machine learning techniques to predict the effectiveness of bank marketing campaigns using data from a Portuguese banking institution. With input variables such as age, job, marital status, and education, the goal is to develop a classification model to classify clients' responses as positive or negative. Descriptive statistics and visualizations, including distplot, count plot, and pair plot, were employed to analyze relationships between variables. The dataset

comprises 45211 observations and 17 columns, with categorical variables like employment, marital status, and education, as well as numerical variables such as age and balance. Treatment of unknown tagged values as null and handling of outliers using interquartile range were essential steps in data preprocessing. Features with over 50% null values were eliminated to enhance model performance.

LITERATURE REVIEW :

In Selecting critical features for data classification based on machine learning methods by Rung-Ching Chen, Christine Dewi, Su-Wen Huang and Rezzy Eko Caraka [1] The significance of feature selection in high-dimensional datasets is the main topic of this study, especially with regard to machine learning applications like classification. The study uses three different datasets for its feature selection: Bank Marketing, Car Evaluation Database, and Human Activity Recognition Using Smartphones. Its primary algorithm is Random Forest. It assesses the effectiveness of several machine learning models, highlighting the influence of feature selection on classification performance and accuracy. These models include Random Forest, Support Vector Machines, K-Nearest Neighbors, and Linear Discriminant Analysis. Through an analysis of several feature selection techniques, including varImp(), Boruta, and Recursive Feature Elimination (RFE), the study emphasizes how important critical feature selection is to achieving better classification results. By evaluating feature relevance, contrasting machine learning models, and offering suggestions for further research, the study makes a contribution.

In Predicting the Success of Bank Telemarketing using various Classification Algorithms by Muneeb Asif [2] In conjunction with feature selection strategies like best-subset Logistic Regression, LASSO, and Random Forest, this thesis investigates the effectiveness of several classification techniques, such as Support Vector Machine, Decision Trees, Random Forest, and Artificial Neural Network. The goal is to minimize dimensionality without sacrificing prediction performance and accuracy. The application context is telemarketing in the banking industry, which highlights the significance of efficient decision-making aided by data mining and Decision Support Systems (DSS). In supervised learning for outcome prediction, in particular, machine learning (ML) is essential. Authentic data is taken from the University of California, Irvine database, and statistical methods are applied to feature selection. The findings show that the accuracy of a smaller set of variables chosen by Random Forest is similar to the accuracy of the entire model across classification techniques, with Random Forest outperforming the others. Notably, Random Forest found that age, balance, and duration are the most influential characteristics.

In Application of Selected Supervised Classification methods to Bank Marketing Campaign by Danieal Grzonka, Grazyna Suchaka and Barbara Borowik [3] This paper explores the use of

classification models, particularly decision trees and ensemble methods like bagging, boosting, and random forests, to predict the effectiveness of marketing campaigns, focusing on a telemarketing campaign by a Portuguese bank. By analyzing real campaign data, the study identifies key attributes influencing client decisions. Despite omitting certain parameters known only post-campaign, the models demonstrate significant predictive capability, with random forests yielding the best results. However, the randomness inherent in the modeling process highlights the need for careful interpretation. Nonetheless, the findings underscore the potential of decision tree-based methods in optimizing bank marketing strategies.

In Research Paper Classification using Supervised Machine Learning Techniques by Shovan Chowdhury and Marco P. Schoen [4] The problem of effectively classifying research papers into different fields, like business, social science, and science, is the focus of this study. The study attempts to automate this process, thereby relieving the laborious task of manual classification by utilizing supervised machine learning techniques and text classification. Support Vector Machines (SVM), Naïve Bayes, k-Nearest Neighbor, and Decision Tree are the four classification algorithms that are used after a balanced dataset consisting of abstracts from different research domains is created. Along with vector representation techniques such as Term Frequency Inverse Document Frequency (TF-IDF) and Bag of Words, text preprocessing techniques like tokenization and stemming are applied. SVM performs better than the other methods, according to the results, though KNN and Naïve Bayes also do fairly well. Overall, the research highlights machine learning's potential.

Machine Learning: Algorithms, Real-World Applications and Research Directions by Iqbal H. Sarker [5] In the context of the Fourth Industrial Revolution (4IR), large and diverse datasets are frequently analyzed. This paper investigates the importance of machine learning (ML) techniques in this regard. It covers a range of machine learning algorithms, such as supervised, unsupervised, semi-supervised, and reinforcement learning, emphasizing how they can be used in real-world settings like e-commerce, cybersecurity, smart cities, healthcare, and agriculture. Emphasis is placed on classification, a crucial component of supervised learning, covering binary, multiclass, and multi-label classification scenarios. The function of popular classification algorithms like Random Forests, Decision Trees, Naive Bayes, and Linear Discriminant Analysis in creating data-driven systems is explained. The study also emphasizes how crucial it is to comprehend the fundamentals and subtleties of various machine learning algorithms in order to effectively handle a range of application requirements in the context of Industry 4.0.

In A novel ensemble approach for estimating the competency of bank telemarketing by WeiGuo, YaoYao, Lihua Liu & Tong Shen [6] In order to forecast bank customers' long-term deposit subscription status, this study explores the field of bank telemarketing. For this purpose, it assesses how well four metaheuristic algorithms—social ski-driver (SSD), harmony search algorithm (HSA), future search algorithm (FSA), and electromagnetic field optimization

(EFO)—train artificial neural networks (ANNs). In terms of prediction accuracy, the hybrid EFO-ANN model proves to be the best, outperforming traditional methods such as logistic regression, decision trees, and support vector machines (SVM). Furthermore, prior studies that utilized machine learning techniques, like ANN and naive Bayes, for comparable predictive tasks in bank telemarketing are referenced, emphasizing the importance of classification algorithms in enhancing marketing tactics and outreach to customers.

In Machine Learning Algorithm for Classification by Haoyuan Tan [7] This study emphasizes supervised learning while examining the performance of several machine learning classifiers in classifying tasks. Popular methods like Random Forest, XGBoost, Naive Bayes, Support Vector Machines (SVM), Gaussian Mixture Model (GMM), and Random Forest are covered, along with examples of how they are used in various datasets. The results show that while overall performance of all classifiers is good, the accuracy is highly dependent on the task's complexity. For example, SVM performs poorly in text classification while Random Forest achieves the best accuracy in remote sensing. Overall, the study emphasizes how important it is to select classifiers based on the particular classification tasks in order to achieve the best results.

In Bank Marketing Data Classification Using Machine Learning by Dr. Smitha Shekar B and Pooja A [8] This paper investigates the application of machine learning concepts, particularly supervised learning algorithms, in analyzing bank marketing data for term deposit subscription prediction. Utilizing Python as the programming language, the study focuses on building a predictive model using the Naïve Bayes classifier algorithm. The dataset, sourced from the UCI Machine Learning Repository and Kaggle, pertains to bank marketing campaigns conducted via phone calls. By assessing metrics like accuracy, precision, recall, and F1 score, the study aims to enhance the bank's targeting strategy towards potential customers who are more likely to respond positively to marketing calls, ultimately improving campaign effectiveness and customer response rates. Additionally, it discusses various machine learning techniques, including Random Forest and Naïve Bayes, emphasizing their roles in data analysis and decision-making in the banking sector.

In Enhancing bank marketing strategies with ensemble learning: Empirical analysis by Xing Tang and Yusi Zhu [9] This paper introduces a customer demand learning model for bank marketing, aiming to enhance market segmentation and competitiveness in the e-commerce banking sector. Leveraging ensemble learning techniques, the study compares the predictive performance of random forest and support vector machine (SVM) models. Results indicate that the ensemble learning model achieves higher accuracy (92%) compared to the SVM model (87%), leading to improved targeted marketing, customer relationship maintenance, and product marketing success. The research contributes valuable insights for bank marketing decision-making, emphasizing the significance of predictive modeling in optimizing marketing strategies and enhancing overall marketing capabilities in the banking industry. Additionally, the

study suggests avenues for future research to further refine the predictive model's accuracy and applicability.

In Classification and prediction‑based machine learning algorithms to predict students' low and high programming performance by Aykut Durak and Vahide Bulut [10] This study explores various estimation and classification techniques, including Decision Trees, k-Nearest Neighbors (kNN), Support Vector Machines (SVM), Naive Bayes, Logistic Regression (LR), and Random Forest (RF), to analyze programming performance factors among students. Decision trees demonstrate superior performance in predicting programming performance, while other classifiers exhibit differences in handling the dependent variable. Factors such as gender and educational level influence variables like computational identity, computational thinking perspective, and programming empowerment scores. Notably, these variables remain consistent across different levels of academic success. By extending existing models and incorporating programming-related variables, the study provides valuable insights into understanding and predicting programming performance among students.

RESEARCH OBJECTIVE :

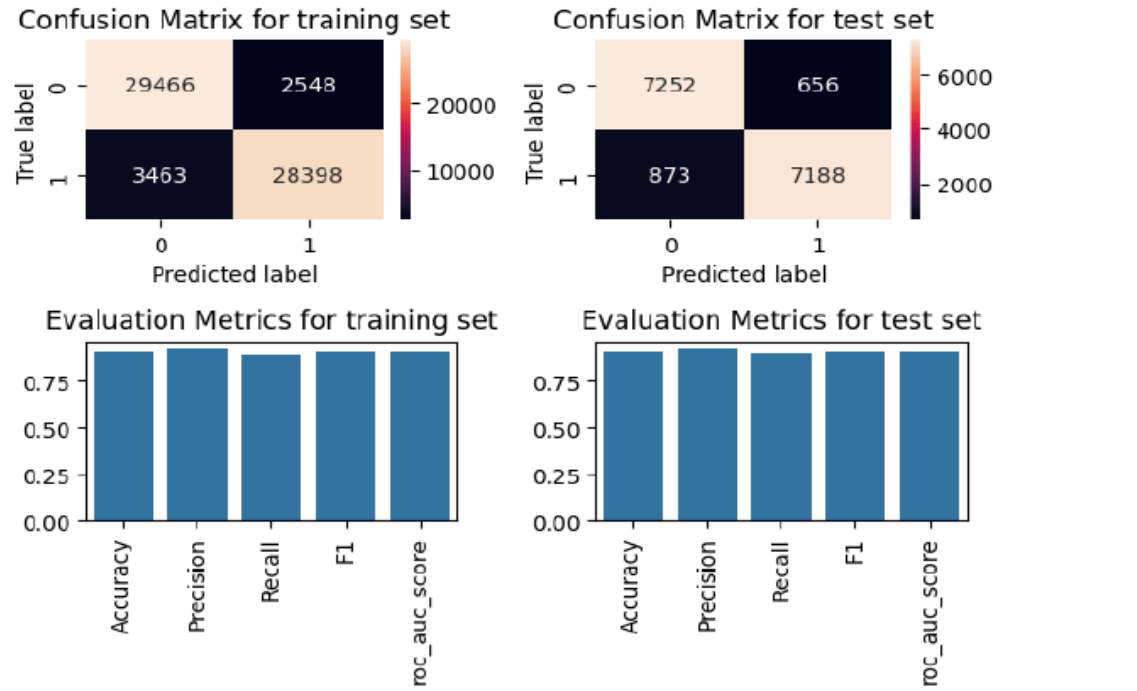
The aim of this study is to create and assess a machine learning-based method for estimating bank marketing campaigns' efficacy. By utilizing a dataset that was acquired through direct marketing initiatives of a Portuguese banking institution, the research endeavors to build predictive models that encompass various customer demographic and behavioral characteristics. Through the use of a variety of machine learning algorithms and performance metrics, including accuracy, precision, recall, F1-score, and roc_auc_score, the study aims to shed light on how best to optimize marketing tactics for the banking industry. The ultimate objective is to use data-driven decision-making processes to increase campaign effectiveness and customer engagement.

RESEARCH METHODOLOGY :

1. LOGISTIC REGRESSION

```
# Import Logistic Regression algorithm in envirnoment
from sklearn.linear_model import LogisticRegression
# Fitting Logistic Regression model to training set
Logistic_regression=LogisticRegression(fit_intercept=True, max_iter=10000,random_state=0)
lr=classification_model(X_train, X_test, y_train, y_test, Logistic_regression)
```

```
================================================================================================
Training set evaluation result :

Confusion Matrix:
 [[29466  2548]
 [ 3463 28398]]
Accuracy:  0.905894324853229
Precision:  0.9176630259161119
Recall:  0.8913091240074071
F1 Score:  0.9042941073447226
roc_auc_score:  0.9058594723554246

--------------------------------

Test set evaluation result :

Confusion Matrix:
 [[7252  656]
 [ 873 7188]]
Accuracy:  0.9042519882271902
Precision:  0.9163691993880673
Recall:  0.8917007815407517
F1 Score:  0.9038667085822069
roc_auc_score:  0.9043734054390659
================================================================================================
```

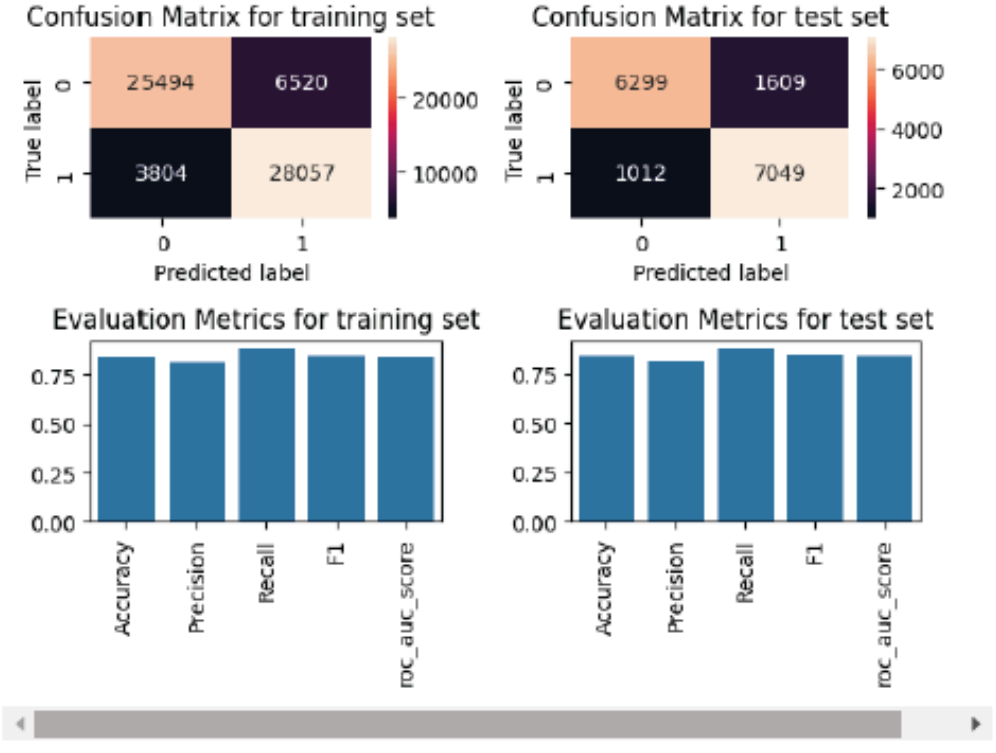



```
================================================================================================
```

2. DECISION TREE:

```
# Import Decision Tree algoritham in envirnoment
from sklearn.tree import DecisionTreeClassifier
# Fitting Decision Tree model to training set
classifier_dt = DecisionTreeClassifier(criterion='entropy', max_leaf_nodes=10, random_state=0)
dt=classification_model(X_train, X_test, y_train, y_test, classifier_dt)
```

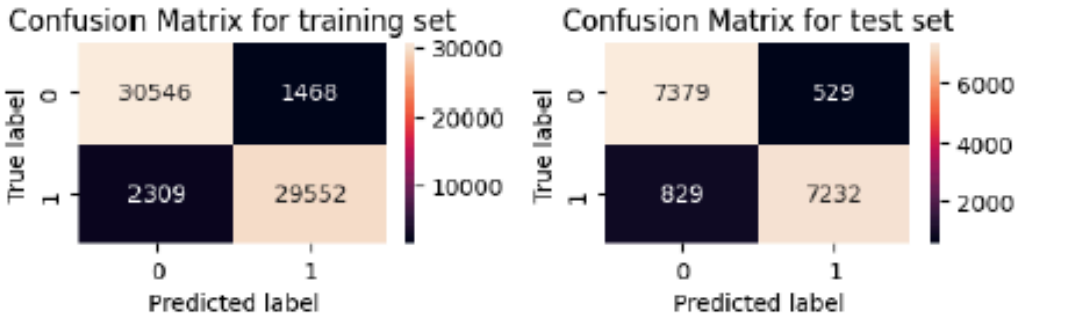
```
DecisionTreeClassifier(criterion='entropy', max_leaf_nodes=10, random_state=0)
================================================================================
Training set evaluation result :

Confusion Matrix:
 [[25494  6520]
 [ 3804 28057]]
Accuracy:  0.8383718199608611
Precision:  0.811435347196113
Recall:  0.8806063839804149
F1 Score:  0.8446070020169181
roc_auc_score:  0.838472742811723

------------------------------

Test set evaluation result :

Confusion Matrix:
 [[6299 1609]
 [1012 7049]]
Accuracy:  0.8358694971507296
Precision:  0.8141603141603142
Recall:  0.8744572633668279
F1 Score:  0.8432322507326994
roc_auc_score:  0.8354962088204904
================================================================================
```

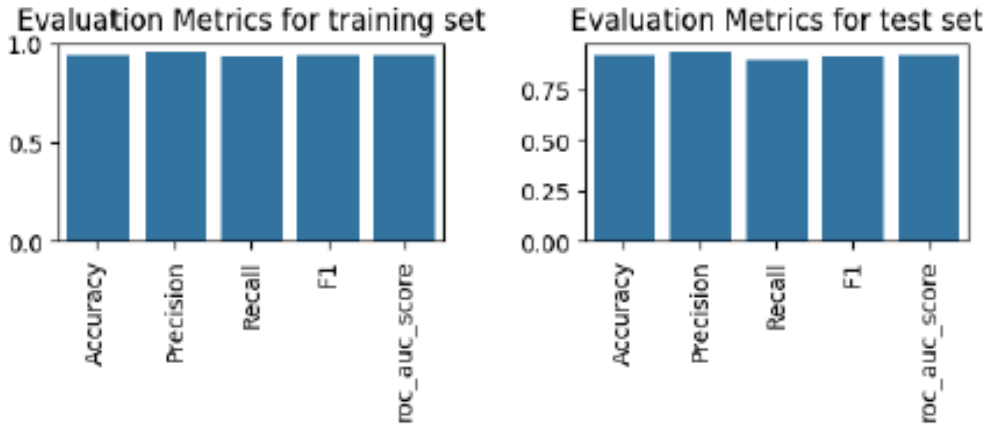


3. K-NEAREST NEIGHBOUR

```
# Import KNN algoritham in envirnoment
from sklearn.neighbors import KNeighborsClassifier
# Fitting model to training set
classifier_knn = KNeighborsClassifier(n_neighbors=5)
knn=classification_model(X_train, X_test, y_train, y_test, classifier_knn)
```

```
================================================================================
Training set evaluation result :

Confusion Matrix:
 [[30546  1468]
 [ 2309 29552]]
Accuracy:  0.9408688845401174
Precision:  0.952675693101225
Recall:  0.9275289538934748
F1 Score:  0.9399341613523958
roc_auc_score:  0.9408370077145266

--------------------------------

Test set evaluation result :

Confusion Matrix:
 [[7379  529]
 [ 829 7232]]
Accuracy:  0.9149602354561964
Precision:  0.9318386805823992
Recall:  0.8971591613943679
F1 Score:  0.9141701428390847
roc_auc_score:  0.9151324385626367
================================================================================
```





RESULT:

Based on the performance metrics provided for the three classification algorithms (Logistic Regression, Decision Tree, and K-Nearest Neighbors), we can make several observations to determine the most suitable model for the bank dataset:
Accuracy: KNN has the highest accuracy of approximately 91.50%, followed by Logistic Regression with around 90.43%, and Decision Tree with about 83.59%.
Precision: KNN has the highest precision of approximately 93.18%, followed by Logistic Regression with around 91.64%, and Decision Tree with about 81.42%.
Recall (Sensitivity): KNN has a recall of approximately 89.72%, followed by Logistic Regression with around 89.17%, and Decision Tree with about 87.45%.
F1 Score: KNN has the highest F1 score of approximately 91.42%, followed by Logistic Regression with around 90.39%, and Decision Tree with about 84.32%.
ROC AUC Score: KNN has the highest ROC AUC score of approximately 91.51%, followed by Logistic Regression with around 90.44%, and Decision Tree with about 83.55%.
Based on these observations, the K-Nearest Neighbors (KNN) model appears to be the most suitable for the bank dataset among the three algorithms. It consistently outperforms Logistic Regression and Decision Tree across all key performance metrics such as accuracy, precision, recall, F1 score, and ROC AUC score. The higher values in these metrics indicate better overall performance and predictive capability of the model.

RESEARCH GAP :
While the presented research provides valuable insights into the development and evaluation of a machine learning model for predicting the effectiveness of bank marketing campaigns, there exist several potential avenues for further exploration and research. Specifically, the following research gaps could be addressed:

1. Exploration of Additional Features: The study mentions incorporating various customer demographic and behavioral attributes into the predictive model. However, there might be other relevant features that could enhance the model's predictive performance. Future research could focus on exploring additional features, such as socio-economic indicators, customer preferences, or external economic factors, to further refine the predictive capabilities of the model.

2. Model Comparison and Selection: The research mentions employing multiple algorithms for prediction but does not delve deeply into the comparative analysis of these algorithms. Future studies could focus on systematically comparing different machine learning algorithms, including ensemble methods and deep learning approaches, to identify the most effective model for predicting marketing campaign effectiveness in the banking sector.

4. Generalizability and External Validation: The research focuses on a specific dataset from a Portuguese banking institution, raising questions about the generalizability of the findings to other banking contexts or geographic regions. Future studies could address this limitation by validating the predictive model using data from multiple banking institutions or conducting cross-validation across diverse datasets to assess its external validity and applicability in different settings.

5. Ethical and Privacy Considerations: The study does not explicitly address ethical or privacy considerations related to the use of customer data for predictive modeling in the banking sector. Future research could explore the ethical implications of deploying machine learning models in marketing campaigns, including issues related to data privacy, fairness, transparency, and the potential impact on consumer trust and autonomy.

Addressing these research gaps could contribute to a deeper understanding of the challenges and opportunities associated with leveraging machine learning for optimizing marketing strategies in the banking sector, ultimately advancing the field of data-driven decision-making in financial institutions.

CONCLUSION :

The K-Nearest Neighbors (KNN) algorithm performs the best among the three models, consistently demonstrating higher accuracy, precision, recall, F1 score, and ROC AUC score.
KNN achieves an accuracy of approximately 91.50%, precision of 93.18%, recall of 89.72%, F1 score of 91.42%, and ROC AUC score of 91.51%.
Logistic Regression and Decision Tree algorithms also show competitive performance but are slightly outperformed by KNN across all metrics.
Therefore, based on the evaluation results, KNN is deemed the most suitable model for the bank dataset, showcasing superior predictive capability and overall effectiveness in classification tasks.

REFERENCES :

1) Selecting critical features for data classification based on machine learning methods by Rung-Ching Chen, Christine Dewi, Su-Wen Huang and Rezzy Eko Caraka.

2) Predicting the Success of Bank Telemarketing using various Classification Algorithms by Muneeb Asif.

3) Application of Selected Supervised Classification methods to Bank Marketing Campaign by Danieal Grzonka, Grazyna Suchaka and Barbara Borowik.

4) Research Paper Classification using Supervised Machine Learning Techniques by Shovan Chowdhury and Marco P. Schoen**.**

5)Machine Learning: Algorithms, Real-World Applications and Research Directions by Iqbal H. Sarker.

6) A novel ensemble approach for estimating the competency of bank telemarketing by WeiGuo, YaoYao, Lihua Liu & Tong Shen.

7) Machine Learning Algorithm for Classification by Haoyuan Tan.

8) Bank Marketing Data Classification Using Machine Learning by Dr. Smitha Shekar B and Pooja A.

9) Enhancing bank marketing strategies with ensemble learning: Empirical analysis by Xing Tang and Yusi Zhu.

10) Classification and prediction-based machine learning algorithms to predict students' low and high programming performance by Aykut Durak and Vahide Bulut.