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:

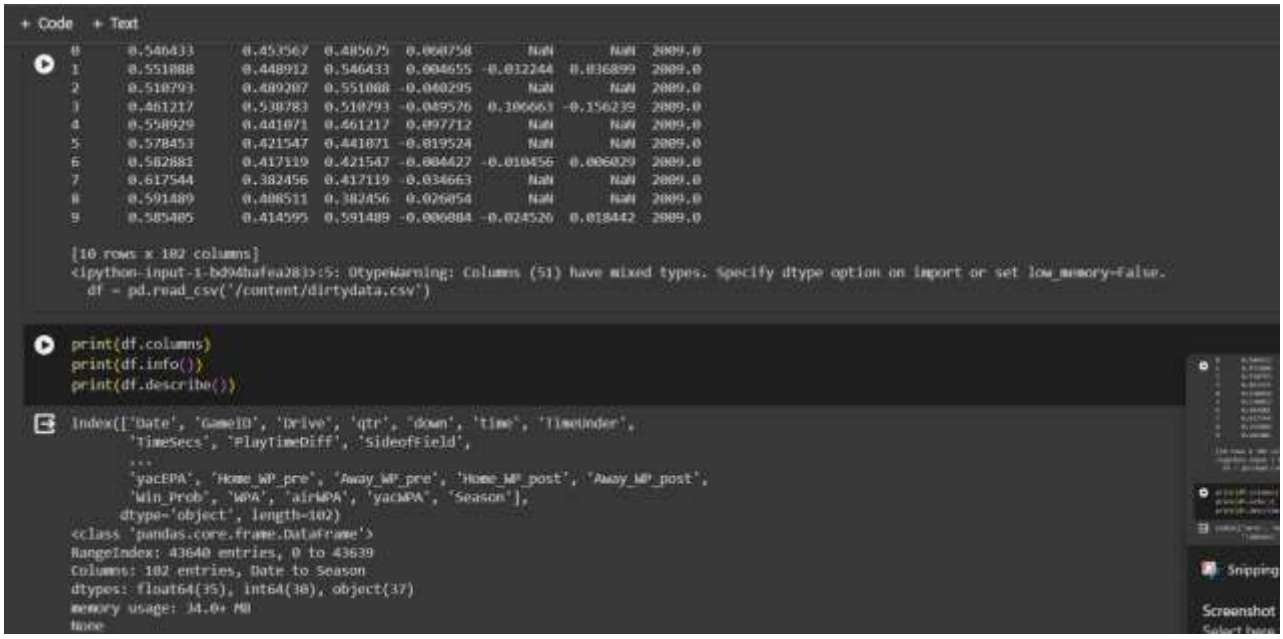
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

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)
```

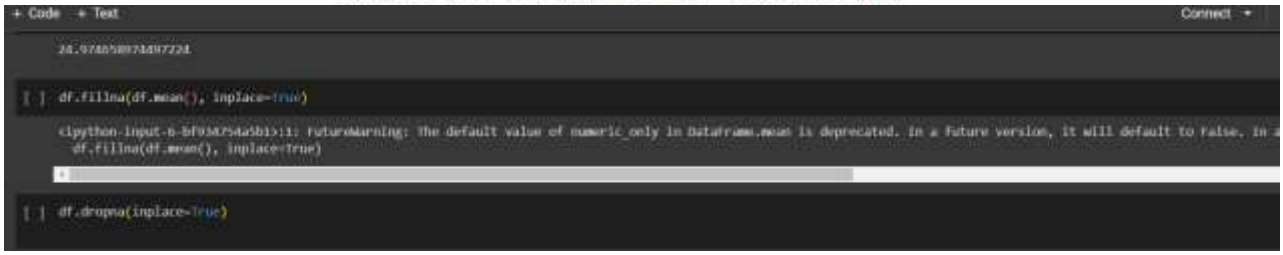
```
[ ]  # get the number of missing data points per column
     missing_values_count = df.isnull().sum()

     # 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

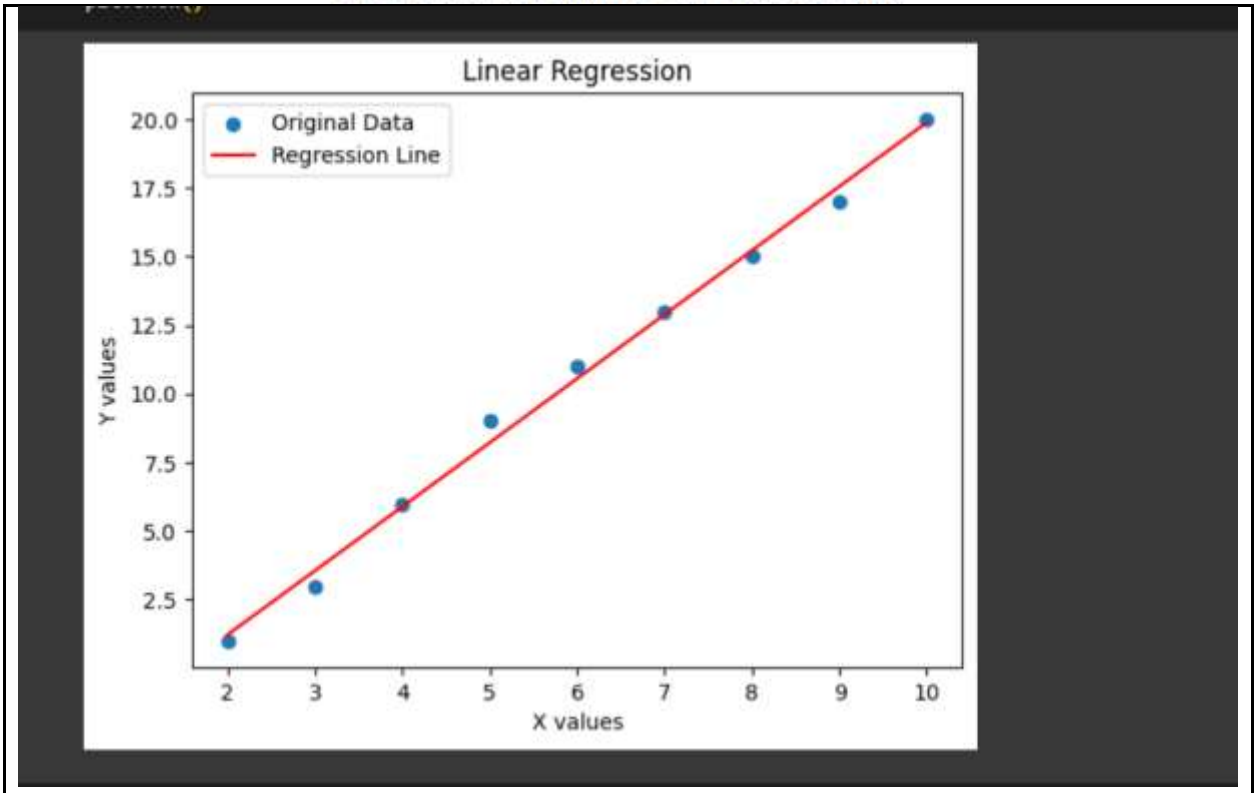
```
import matplotlib.pyplot as plt
```

```
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.