In [11]:

```
import pandas as pd
```

In [12]:

```
df=pd.read_csv("/Users/myyntiimac/Desktop/Salary_Data.csv")
df.head()
```

Out[12]:

	YearsExperience	Salary
0	1.1	39343.0
1	0.0	21111.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

In [13]:

```
df.isnull().any()
```

Out[13]:

```
YearsExperience    False
Salary              True
dtype: bool
```

In [14]:

```
1 df
```

Out[14]:

	YearsExperience	Salary
0	1.1	39343.0
1	0.0	21111.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60105.0
7	3.2	NaN
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0
11	4.0	55794.0
12	4.0	56957.0
13	4.1	57081.0
14	4.5	61111.0
15	0.0	21111.0
16	5.1	NaN
17	5.3	83088.0
18	5.9	81363.0
19	6.0	9394.0
20	0.0	21111.0
21	7.1	98273.0
22	7.9	101302.0
23	8.2	113812.0
24	8.7	109431.0
25	9.0	105582.0
26	9.5	116969.0
27	9.6	NaN
28	10.5	121872.0
29	5.0	589.0

In [15]:

```
# Impute NaN values with column means
df_filled = df.fillna(df.mean())

print("Original DataFrame:")
print(df)

print("\nDataFrame with NaNs Imputed by Column Means:")
print(df_filled)
```

```
Original DataFrame:
    YearsExperience    Salary
0               1.1   39343.0
1               0.0   21111.0
2               1.5   37731.0
3               2.0   43525.0
4               2.2   39891.0
5               2.9   56642.0
6               3.0   60105.0
7               3.2       NaN
8               3.2   64445.0
9               3.7   57189.0
10              3.9   63218.0
11              4.0   55794.0
12              4.0   56957.0
13              4.1   57081.0
14              4.5   61111.0
15              0.0   21111.0
16              5.1       NaN
17              5.3   83088.0
18              5.9   81363.0
19              6.0    9394.0
20              0.0   21111.0
21              7.1   98273.0
22              7.9  101302.0
23              8.2  113812.0
24              8.7  109431.0
25              9.0  105582.0
26              9.5  116969.0
27              9.6       NaN
28             10.5  121872.0
29              5.0     589.0

DataFrame with NaNs Imputed by Column Means:
    YearsExperience        Salary
0               1.1   39343.00000
1               0.0   21111.00000
2               1.5   37731.00000
3               2.0   43525.00000
4               2.2   39891.00000
5               2.9   56642.00000
6               3.0   60105.00000
7               3.2   62890.37037
8               3.2   64445.00000
9               3.7   57189.00000
10              3.9   63218.00000
11              4.0   55794.00000
12              4.0   56957.00000
13              4.1   57081.00000
14              4.5   61111.00000
15              0.0   21111.00000
16              5.1   62890.37037
17              5.3   83088.00000
18              5.9   81363.00000
19              6.0    9394.00000
20              0.0   21111.00000
21              7.1   98273.00000
22              7.9  101302.00000
23              8.2  113812.00000
24              8.7  109431.00000
25              9.0  105582.00000
```

```
26              9.5  116969.00000
27              9.6   62890.37037
28             10.5  121872.00000
29              5.0     589.00000
```

In [18]:

```
1 df_filled.head()
```

Out[18]:

	YearsExperience	Salary
0	1.1	39343.0
1	0.0	21111.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

In [19]:

```
from sklearn.preprocessing import StandardScaler
# Initialize the StandardScaler
scaler = StandardScaler()

# Fit the scaler on the data and transform the DataFrame
scaled_data = scaler.fit_transform(df_filled)

# Create a new DataFrame with scaled values
scaled_df = pd.DataFrame(scaled_data, columns=df_filled.columns)

print("Original DataFrame:")
print(df)

print("\nScaled DataFrame:")
print(scaled_df)
```

```
Original DataFrame:
    YearsExperience    Salary
0               1.1   39343.0
1               0.0   21111.0
2               1.5   37731.0
3               2.0   43525.0
4               2.2   39891.0
5               2.9   56642.0
6               3.0   60105.0
7               3.2       NaN
8               3.2   64445.0
9               3.7   57189.0
10              3.9   63218.0
11              4.0   55794.0
12              4.0   56957.0
13              4.1   57081.0
14              4.5   61111.0
15              0.0   21111.0
16              5.1       NaN
17              5.3   83088.0
18              5.9   81363.0
19              6.0    9394.0
20              0.0   21111.0
21              7.1   98273.0
22              7.9  101302.0
23              8.2  113812.0
24              8.7  109431.0
25              9.0  105582.0
26              9.5  116969.0
27              9.6       NaN
28             10.5  121872.0
29              5.0     589.0

Scaled DataFrame:
    YearsExperience        Salary
0         -1.215075 -7.356500e-01
1         -1.586005 -1.305241e+00
2         -1.080192 -7.860109e-01
3         -0.911588 -6.049989e-01
4         -0.844146 -7.185297e-01
5         -0.608100 -1.952071e-01
6         -0.574379 -8.701853e-02
7         -0.506937  2.273102e-16
8         -0.506937  4.856862e-02
9         -0.338333 -1.781181e-01
10        -0.270891  1.023557e-02
11        -0.237170 -2.216997e-01
12        -0.237170 -1.853661e-01
13        -0.203449 -1.814922e-01
14        -0.068566 -5.558981e-02
15        -1.586005 -1.305241e+00
16         0.133759  2.273102e-16
17         0.201201  6.309998e-01
18         0.403526  5.771086e-01
19         0.437247 -1.671295e+00
20        -1.586005 -1.305241e+00
21         0.808177  1.105399e+00
22         1.077944  1.200028e+00
23         1.179107  1.590857e+00
24         1.347711  1.453989e+00
25         1.448874  1.333741e+00
```

```
26          1.617478  1.689486e+00
27          1.651199  2.273102e-16
28          1.954687  1.842662e+00
29          0.100039 -1.946374e+00
```

In [20]:

```
# Split the DataFrame into features (X) and target (y)
X = scaled_df.drop(columns=['Salary'])
y = scaled_df['Salary']
```

In [21]:

```
from sklearn.model_selection import train_test_split
```

In [22]:

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

In [23]:

```
from sklearn.linear_model import LinearRegression
```

In [24]:

```
# Create a Linear Regression model
model = LinearRegression()
```

In [25]:

```
# Fit the model on the training data
model.fit(X_train, y_train)
```

Out[25]:

```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [26]:

```
# Make predictions on the testing data
y_pred = model.predict(X_test)
```

In [31]:

```
import numpy as np
from sklearn.metrics import mean_squared_error
```

In [32]:

```
1
2 # Calculate RMSE
3 rmse = np.sqrt(mean_squared_error(y_test, y_pred))
4 print("Root Mean Squared Error (RMSE):", rmse)
```

```
Root Mean Squared Error (RMSE): 1.0933466614839007
```

In [ ]:

```
1
```