Response variable (dependant variable): The variable you want to predict

Explanatory variable (independant variable): The variable that explains how the response variable will change

In [43]:
```
import numpy as np
import pandas as pd

# Create a DataFrame with X and Y columns
data = pd.DataFrame({'Hours_Studied': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                     'Exam_Score': [45, 50, 55, 60, 65, 70, 75, 80, 85, 90]
```

In [44]:
```
# Import the ols function
from statsmodels.formula.api import ols

# Create the model object
#ols('dependant ~ independant', data=data)
lin_reg = ols('Exam_Score ~ Hours_Studied ', data=data)

# This step estimates the coefficients of the linear regression equation (t
#best describe the relationship between 'X' and 'Y' based on the data.
lin_reg = lin_reg.fit()

# this line prints the parameters of the fitted linear regression model. li
#of the model, including the intercept (the coefficient for the constant te
print(lin_reg.params)
```

```
Intercept        40.0
Hours_Studied     5.0
dtype: float64
```

Predictions

explanatory_data = pd.DataFrame({"explanatory_var": list_of_values})
predictions = model.predict(explanatory_data)
prediction_data = explanatory_data.assign(response_var=predictions)

To make predictions, you need a DataFrame with the same column names and structure as your original dataset but with the values for which you want to make predictions. Create a new DataFrame or modify your existing one with the values for prediction.

In [22]:

```
new_data = pd.DataFrame({'Exam_Score': [58, 60, 72]})
predictions = lin_reg.predict(new_data)
prediction_data = new_data.assign(predictions=predictions)
print(prediction_data)
```

```
   Exam_Score  predictions
0          58          3.6
1          60          4.0
2          72          6.4
```

In [35]:

```
print(lin_reg.fittedvalues.head(n=10))
print()

print(' Residuals represent the errors or deviations of the actual data poi
print(lin_reg.resid.head(n=10))
```

```
0     1.0
1     2.0
2     3.0
3     4.0
4     5.0
5     6.0
6     7.0
7     8.0
8     9.0
9    10.0
dtype: float64

 Residuals represent the errors or deviations of the actual data points f
rom the regression line or model.
0    2.664535e-15
1    2.664535e-15
2    2.664535e-15
3    2.664535e-15
4    4.440892e-15
5    4.440892e-15
6    4.440892e-15
7    4.440892e-15
8    3.552714e-15
9    7.105427e-15
dtype: float64
```

Summary

In [26]:
```python
print(lin_reg.summary())
```

```
                            OLS Regression Results
==========================================================================
=====
Dep. Variable:          Hours_Studied   R-squared:
1.000
Model:                            OLS   Adj. R-squared:
1.000
Method:                 Least Squares   F-statistic:                  3.87
3e+30
Date:                Tue, 05 Sep 2023   Prob (F-statistic):           4.98
e-120
Time:                        14:31:22   Log-Likelihood:                  3
17.02
No. Observations:                  10   AIC:                             -
630.0
Df Residuals:                       8   BIC:                             -
629.4
Df Model:                           1
Covariance Type:            nonrobust
==========================================================================
=====
                 coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------
-----
Intercept     -8.0000   7.01e-15  -1.14e+15      0.000      -8.000       -
8.000
Exam_Score     0.2000   1.02e-16   1.97e+15      0.000       0.200
0.200
==========================================================================
=====
Omnibus:                        5.713   Durbin-Watson:
0.097
Prob(Omnibus):                  0.057   Jarque-Bera (JB):
2.157
Skew:                           1.088   Prob(JB):
0.340
Kurtosis:                       3.665   Cond. No.
332.
==========================================================================
=====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is co
rrectly specified.

C:\Users\Rubab\AppData\Local\Programs\Python\Python310\lib\site-packages
\scipy\stats\_stats_py.py:1806: UserWarning: kurtosistest only valid for
n>=20 ... continuing anyway, n=10
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
```

Transforming Variables

In [36]:

```
import pandas as pd
import numpy as np

# Create a sample DataFrame
data = pd.DataFrame({
    'Product': ['A', 'B', 'C', 'D', 'E'],
    'Price': [100, 250, 75, 300, 150]
})

# Display the original DataFrame
print("Original DataFrame:")
print(data)

# Perform a natural logarithm transformation on the 'Price' column
data['Log_Price'] = np.log(data['Price'])

# Display the DataFrame with the transformed variable
print("\nDataFrame with Log-Transformed Price:")
print(data)
```

```
Original DataFrame:
  Product  Price
0       A    100
1       B    250
2       C     75
3       D    300
4       E    150

DataFrame with Log-Transformed Price:
  Product  Price  Log_Price
0       A    100   4.605170
1       B    250   5.521461
2       C     75   4.317488
3       D    300   5.703782
4       E    150   5.010635
```

Quantifying Model Fit: This refers to the process of evaluating how well a statistical model's predictions align with the actual observed data. It helps us understand and assess the accuracy of the model's predictions.

Coefficient of Determination (R-squared):

R-squared is a metric used to assess model fit. It ranges from 0 to 1, where 1 indicates a perfect fit (the model explains all variance in the response variable), and 0 suggests that the model's predictions are no better than randomness. R-squared is accessible inside the .summary() or as .rsquared when working with models. Residual Standard Error (RSE):

RSE Residuals represent how much the model's predictions deviate from the actual data. It can be calculated from the Mean Squared Error (MSE), which is the square of RSE. RSE is accessible using .mse_resid() when working with models. To calculate RSE manually, you square each residual, sum them up, calculate degrees of freedom, and then take the square root of the sum of squares divided by degrees of freedom.

Root Mean Square Error (RMSE):

RMSE is similar to MSE (Mean Squared Error), but it does not remove degrees of freedom; it divides by the number of observations only. RMSE is another way to assess the model's prediction accuracy, particularly when degrees of freedom are not considered

Logistic Regression

Used when the response variable is binary/logical

In [41]:

```
# Import logit
from statsmodels.formula.api import logit

# Create a DataFrame for logistic regression
data = pd.DataFrame({
    'Age': [25, 30, 35, 40, 45, 20, 55, 60, 28, 50],
    'Purchase': [1,1,1,0,1,0,1,1,0,1]  # 1 for Yes, 0 for No
})

purchased = logit('Purchase~Age', data=data).fit()

# Print the parameters of the fitted model
print(purchased.params)
```

```
Optimization terminated successfully.
         Current function value: 0.474479
         Iterations 7
Intercept   -3.111943
Age          0.112027
dtype: float64
```

Predicitions

In [48]:

```
explanatory_data = pd.DataFrame({'Age': np.arange(30, 55)})

# Create prediction_data
prediction_data = explanatory_data.assign(has_purchased = purchased.predict

# Print the head
print(prediction_data.head())
```

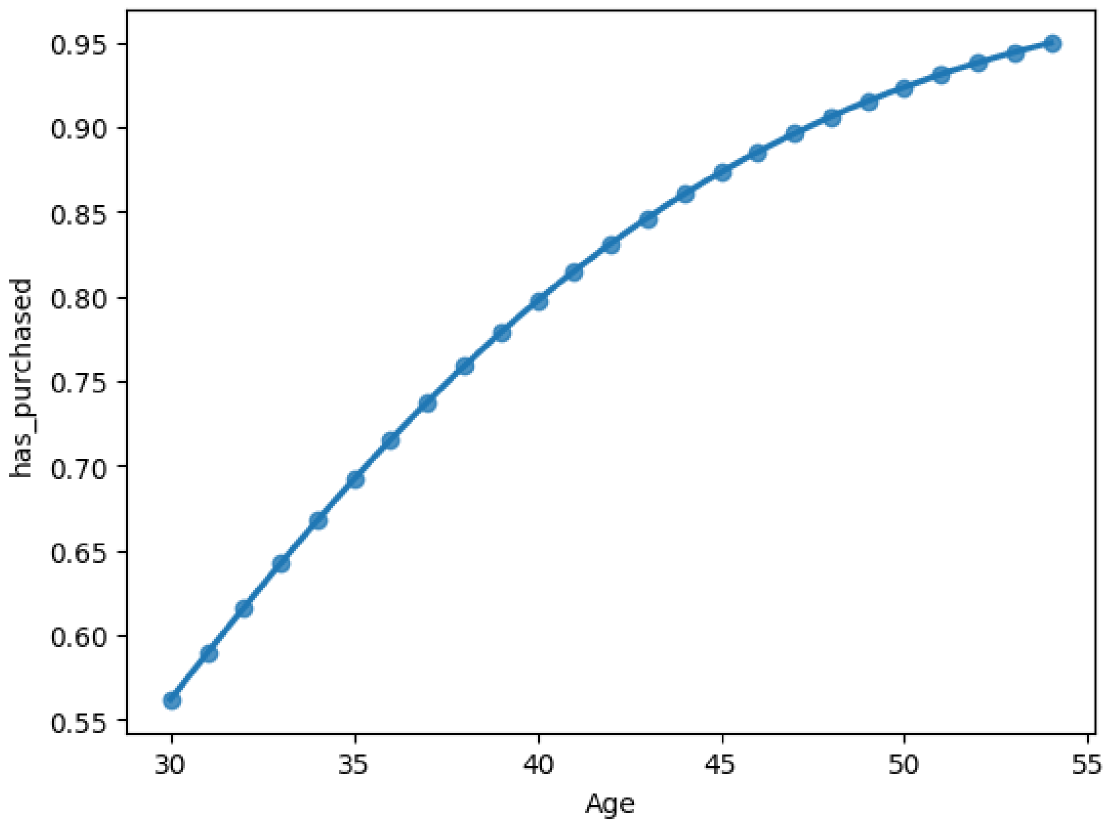
```
   Age  has_purchased
0   30       0.561900
1   31       0.589259
2   32       0.616077
3   33       0.642207
4   34       0.667519
```

In [52]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
# Create a new figure
fig = plt.figure()

# Create a scatter plot with logistic trend line
sns.regplot(x="Age",
            y="has_purchased",
            data=prediction_data,
            ci=None,
            logistic=True)
```

```
C:\Users\Rubab\AppData\Local\Programs\Python\Python310\lib\site-packages
\statsmodels\genmod\generalized_linear_model.py:1257: PerfectSeparationWa
rning: Perfect separation or prediction detected, parameter may not be id
entified
  warnings.warn(msg, category=PerfectSeparationWarning)
```

Out[52]: <Axes: xlabel='Age', ylabel='has_purchased'>



In [54]:

```
# Import mosaic from statsmodels.graphics.mosaicplot
from statsmodels.graphics.mosaicplot import mosaic

# Calculate the confusion matrix conf_matrix
conf_matrix = purchased.pred_table()

# Draw a mosaic plot of conf_matrix
mosaic(conf_matrix)
plt.show()
```