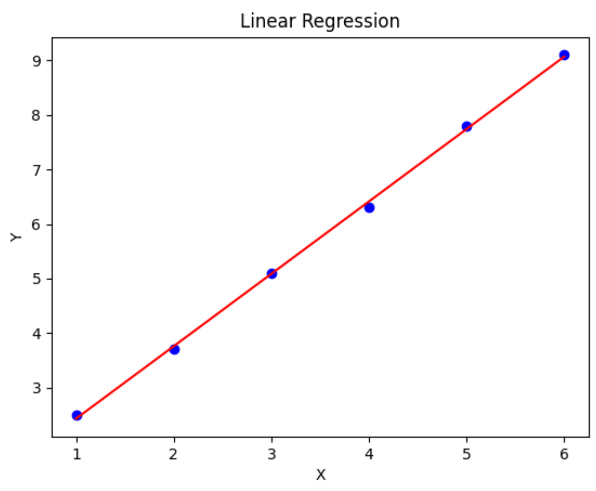
Machine Learning #17

▼ 1. Using a graph to illustrate slope and intercept, define basic linear regression.

Linear regression is a method used to find the best linear relationship between two continuous variables. One variable is considered the dependent variable (Y), while the other is the independent variable (X). The goal of linear regression is to find the slope (β1) and the intercept (β0) that minimize the distance between the line of best fit and the observed data points.

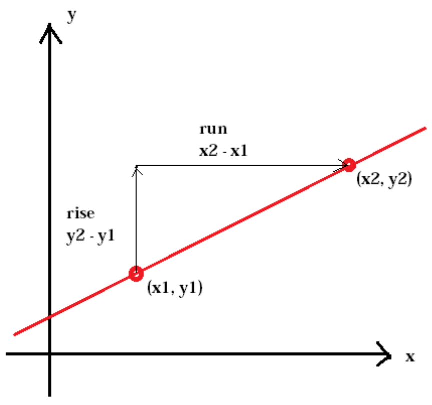
```
import matplotlib.pyplot as plt
import numpy as np

# Generate data
x = np.array([1, 2, 3, 4, 5, 6])
y = np.array([2.5, 3.7, 5.1, 6.3, 7.8, 9.1])

# Calculate slope and intercept
beta1, beta0 = np.polyfit(x, y, 1)

# Plot data and line of best fit
plt.scatter(x, y, color='blue')
plt.plot(x, beta1*x + beta0, color='red')

# Add labels and title
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Linear Regression')

plt.show()
```



In this example, the slope (β1) of the line of best fit is positive, indicating a positive relationship between X and Y. The intercept (β0) is the point where the line crosses the Y-axis.

The line of best fit is determined by minimizing the sum of the squared differences between the observed data points and the predicted values of Y on the line. This method is known as the method of least squares.

▼ 2. In a graph, explain the terms rise, run, and slope.

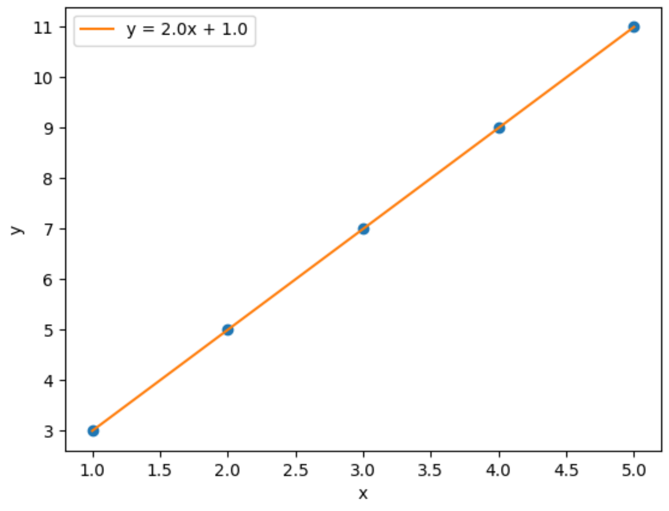


- ***Rise***: ***The rise is the vertical distance between two points on a line***. It is calculated as the difference between the y-coordinates of the two points.
- ***Run***: ***The run is the horizontal distance between two points on a line***. It is calculated as the difference between the x-coordinates of the two points.
- ***Slope***: ***The slope of a line is the ratio of the rise to the run.*** It is calculated as the change in y divided by the change in x. The slope gives the direction and steepness of the line.

For example, if we have two points on a line, (1, 2) and (4, 8), the rise is 6 (8 - 2), the run is 3 (4 - 1), and the slope is 2 (6 / 3).

▼ 3. Use a graph to demonstrate slope, linear positive slope, and linear negative slope, as well as the different conditions that contribute to the slope.

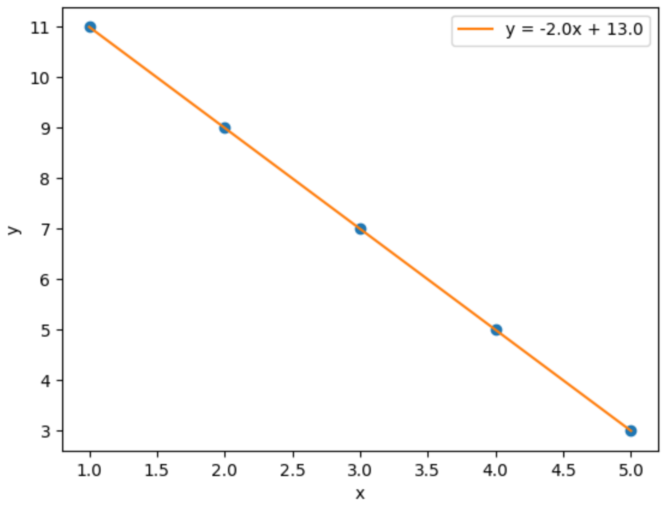
```
import matplotlib.pyplot as plt
import numpy as np

# Generate some sample data
x = np.array([1, 2, 3, 4, 5])
y = np.array([3, 5, 7, 9, 11])

# Calculate the slope of the line
slope, intercept = np.polyfit(x, y, 1)

# Create a plot
plt.plot(x, y, 'o')
plt.plot(x, slope*x + intercept, label='y = {}x + {}'.format(round(slope,2), round(intercept,2)))
plt.legend()

# Set the axes labels
plt.xlabel('x')
plt.ylabel('y')

# Show the plot
plt.show()
```



In this example, the data points create a line with a positive slope, meaning that as the x values increase, the y values also increase. The slope of the line is calculated using the `np.polyfit()` function, which fits a line to the data points and returns the slope and intercept. The slope is then used to plot the line using the equation `y = mx + b`, where `m` is the slope and `b` is the y-intercept.

To create a linear negative slope, we can use the same code but modify the `y` values so that they decrease as the `x` values increase. Alternatively, we can simply flip the sign of the slope:

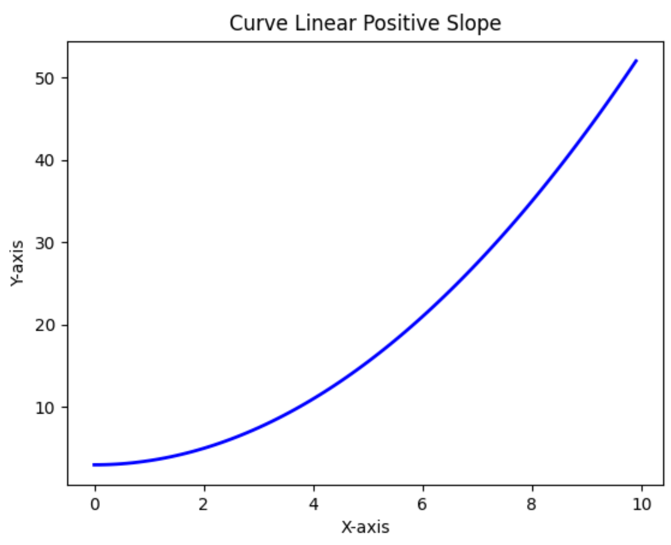
```
# Generate some sample data with a negative slope
x = np.array([1, 2, 3, 4, 5])
y = np.array([11, 9, 7, 5, 3])

# Calculate the slope of the line (note the negative sign)
slope, intercept = np.polyfit(x, y, 1)

# Create a plot
plt.plot(x, y, 'o')
plt.plot(x, slope*x + intercept, label='y = {}x + {}'.format(round(slope,2), round(intercept,2)))
plt.legend()

# Set the axes labels
plt.xlabel('x')
plt.ylabel('y')

# Show the plot
plt.show()
```



In this case, the line has a negative slope, meaning that as the `x` values increase, the `y` values decrease. The slope is negative because the `y` values decrease by `2` for every `1` increase in `x`.

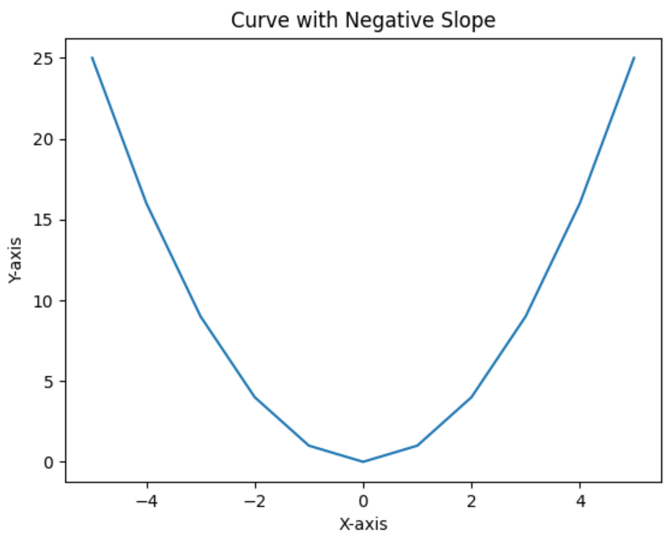
▼ 4. Use a graph to demonstrate curve linear negative slope and curve linear positive slope.

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

# Generate data points for x and y variables
x = np.arange(0, 10, 0.1)
y = 3 + 0.5*x**2

# Plot the curve linear positive slope graph
plt.plot(x, y, color='blue', linewidth=2)

# Add labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Curve Linear Positive Slope')

# Display the plot
plt.show()
```



A curve linear positive slope, on the other hand, occurs when a curve line slopes upwards as it progresses from left to right. This indicates that as the value of the independent variable increases, the value of the dependent variable also increases. An example of a curve linear positive slope can be seen in the relationship between the price of a product and the demand for it.

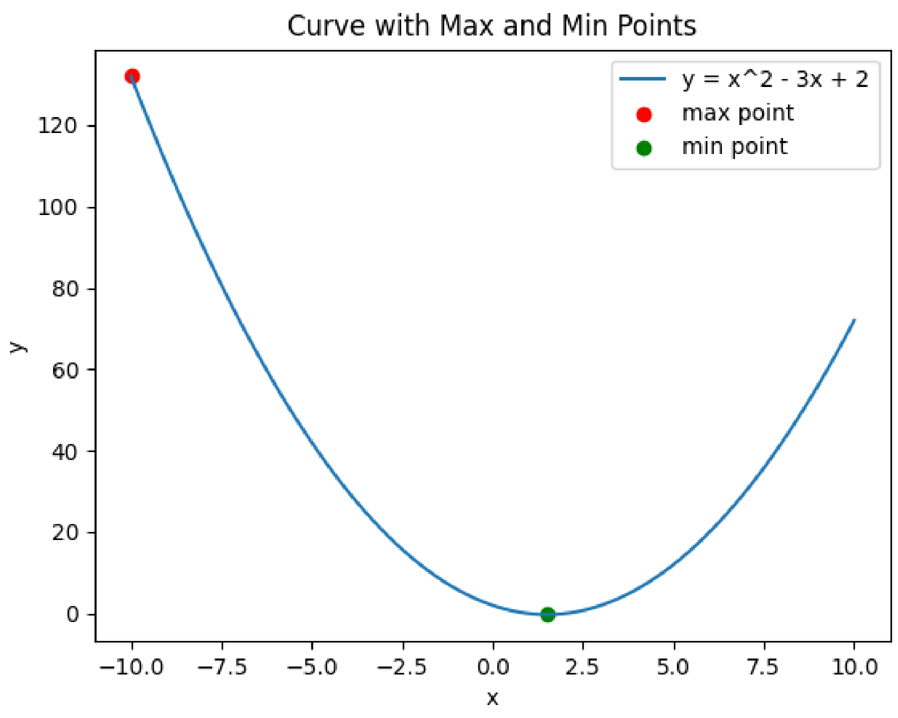
```
import numpy as np
import matplotlib.pyplot as plt

# Generate data points
x = np.arange(-5, 6, 1)
y = np.array([25, 16, 9, 4, 1, 0, 1, 4, 9, 16, 25])

# Plot the curve
plt.plot(x, y)

# Add labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Curve with Negative Slope')

# Show the plot
plt.show()
```



A curve linear negative slope occurs when a curve line slopes downwards as it progresses from left to right. This indicates that as the value of the independent variable increases, the value of the dependent variable decreases. An example of a curve linear negative slope can be seen in the relationship between the age of a car and its resale value.

▼ 5. Use a graph to show the maximum and low points of curves.

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

# create x and y data
x = np.linspace(-10, 10, 100)
y = x ** 2 - 3 * x + 2

# find the maximum and minimum points
x_max = x[np.argmax(y)]
x_min = x[np.argmin(y)]
y_max = np.max(y)
y_min = np.min(y)

# plot the curve
plt.plot(x, y, label='y = x^2 - 3x + 2')

# plot the maximum and minimum points
plt.scatter(x_max, y_max, marker='o', color='r', label='max point')
plt.scatter(x_min, y_min, marker='o', color='g', label='min point')

# add legend and labels
plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.title('Curve with Max and Min Points')

# show the plot
plt.show()
```



▼ 6. Use the formulas for a and b to explain ordinary least squares.

Ordinary least squares (OLS) is a technique used to find the best-fitting line through a set of points. In simple linear regression, the OLS method involves finding the values of the intercept (a) and slope (b) that minimize the sum of squared residuals, which is the sum of the squared differences between the observed values and the predicted values.

The formula for the slope (b) in OLS is:

$$b = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

where xi and yi are the observed values, $\bar{x}$ and $\bar{y}$ are the means of the xi and yi values, respectively.

The formula for the intercept (a) in OLS is:

$$a = \bar{y} - b\bar{x}$$

where $\bar{y}$ is the mean of the yi values and $\bar{x}$ is the mean of the xi values.

Together, these formulas allow us to calculate the values of a and b that minimize the sum of squared residuals and provide the best-fitting line through a set of points.

▼ 7. Provide a step-by-step explanation of the OLS algorithm.

Here is a step-by-step explanation of the OLS algorithm:

1. ***Collect data***: Collect a dataset that contains the predictor variable (x) and the response variable (y).

2. ***Calculate the means:*** Calculate the mean of the predictor variable $(\bar{x})$ and the response variable $(\bar{y})$.
3. ***Calculate the deviations***: Calculate the deviations of each observation from their respective means. That is, for each observation i, calculate $(x_i - \bar{x})$ and $(y_i - \bar{y})$.
4. ***Calculate the sum of squares for x:*** Calculate the sum of squares of the deviations of the predictor variable. That is, calculate $\Sigma(xi - \bar{x})^2$.
5. ***Calculate the sum of products of x and y:*** Calculate the sum of the products of the deviations of x and y. That is, calculate $\Sigma(xi - \bar{x})(yi - \bar{y})$.
6. ***Calculate the slope:*** Calculate the slope (b) of the regression line using the formula:

   $b = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$
7. ***Calculate the intercept:*** Calculate the intercept (a) of the regression line using the formula:

   $a = \bar{y} - b * \bar{x}$
8. ***Build the regression equation:*** The regression equation is of the form $y = a + bx$. Using the values of a and b calculated in steps 6 and 7, the regression equation can be built.
9. ***Evaluate the model:*** The model can be evaluated by checking the residuals, R-squared value, and other statistical measures.
10. ***Make predictions:*** Using the regression equation, predictions can be made for new values of x.

That's a brief explanation of the OLS algorithm. It is widely used in linear regression analysis to find the best-fitting line to a given set of data points.

▼ 8. What is the regression's standard error? To represent the same, make a graph.

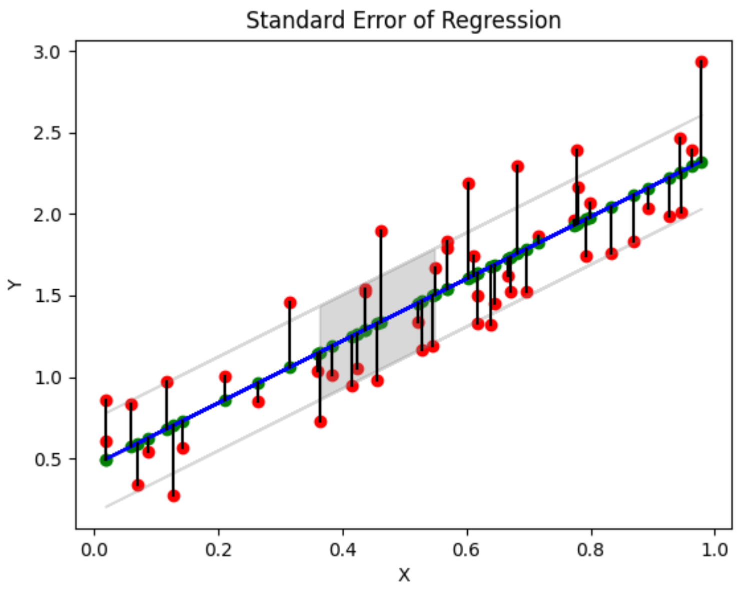
The regression's standard error is a measure of the variation or scatter of the data points around the regression line. It is an estimate of the average distance that the observed data points fall from the regression line.

It is calculated using the following formula:

$$SE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n-2}}$$

where y_i is the observed value of the dependent variable, $\hat{y}_i$ is the predicted value of the dependent variable from the regression equation, and n is the sample size.

Here's an example of a graph that shows a regression line with some scatter around it:



In this graph, the blue line represents the regression line, which is the line of best fit for the data. The red dots represent the actual data points, and the green dots represent the predicted values from the regression line. The vertical lines between the red and green dots show the distances between the actual and predicted values for each data point. The standard error of the regression is represented by the gray shaded area, which shows the typical distance between the data points and the regression line. The wider the shaded area, the higher the standard error, indicating that the data points are more scattered around the regression line.

▼ Reference Code for graph

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

# Generate some sample data
np.random.seed(0)
x = np.random.rand(50)
y = 2 * x + np.random.rand(50)

# Perform linear regression
coef = np.polyfit(x, y, deg=1)
line = np.polyval(coef, x)

# Calculate residuals
residuals = y - line

# Calculate standard error of the regression
std_error = np.std(residuals)

# Plot the data and regression line
plt.scatter(x, y, color='red')
plt.plot(x, line, color='blue')

# Plot the predicted values from the regression line
predicted = coef[0] * x + coef[1]
plt.scatter(x, predicted, color='green')

# Plot the vertical lines between actual and predicted values
for i in range(len(x)):
    plt.plot([x[i], x[i]], [y[i], predicted[i]], color='black')

# Plot the shaded area representing standard error of regression
```

```
plt.fill_between(x, line - std_error, line + std_error, color='gray', alpha=0.3)

# Add labels and title
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Standard Error of Regression')

# Show the plot
plt.show()
```

▼ 9. Provide an example of multiple linear regression.

Suppose we want to predict the house prices based on various factors like the size of the house, the number of bedrooms, and the age of the house. We collect data on 100 houses in a certain area and record the following information:

- Size of the house (in square feet): X1
- Number of bedrooms: X2
- Age of the house (in years): X3
- Selling price (in dollars): Y

We can use multiple linear regression to model the relationship between the predictors (X1, X2, X3) and the response variable (Y) using the following equation:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \varepsilon$$

where β_0 is the intercept, β_1 is the coefficient for X_1, β_2 is the coefficient for X_2, β_3 is the coefficient for X_3, and ε is the error term.

We can estimate the coefficients $\beta_0, \beta_1, \beta_2$, and β_3 using least squares regression, which involves finding the values of these coefficients that minimize the sum of the squared differences between the observed values of Y and the predicted values of Y based on the predictors.

Once we have estimated the coefficients, we can use the equation to predict the selling price of a house based on its size, number of bedrooms, and age. For example, if a house has a size of 2000 square feet, 3 bedrooms, and is 10 years old, we can predict its selling price using the equation:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \varepsilon$$

$= (estimatedvalueof \beta_0) + (estimatedvalueof \beta_1) * 2000 + (estimatedvalueof \beta_2) * 3 + (estimatedvalueof \beta_3) * 10$

The above equation would give us the predicted value of the selling price of the house.

▼ 10. Describe the regression analysis assumptions and the BLUE principle.

Regression analysis assumptions are a set of assumptions that should be met to ensure the accuracy and reliability of regression results. These assumptions include:

1. ***Linearity:*** The relationship between the independent and dependent variables should be linear.
2. ***Independence:*** The observations should be independent of each other.
3. ***Homoscedasticity:*** The variance of the residuals should be constant across all levels of the independent variables.
4. ***Normality:*** The residuals should be normally distributed.

5. ***No multicollinearity:*** There should be no high correlation between independent variables.

The BLUE principle stands for Best Linear Unbiased Estimator. ***It is a principle that states that the regression coefficient estimates should be both unbiased and have the smallest variance among all unbiased linear estimators.*** In other words, the regression coefficients should be estimated using the linear combination of independent variables that gives the most accurate and precise results.

▼ 11. Describe two major issues with regression analysis.

1. ***Overfitting:*** ***Overfitting occurs when a model is too complex and is fit to the noise in the data rather than the underlying relationship between the variables.*** This results in a model that performs well on the training data but poorly on new data. Overfitting can be addressed by simplifying the model, using regularization techniques, or increasing the amount of data used to train the model.

2. ***Multicollinearity:*** ***Multicollinearity occurs when two or more predictor variables in a regression model are highly correlated with each other.*** This can cause problems with interpreting the coefficients and standard errors, as well as reducing the overall predictive power of the model. Multicollinearity can be addressed by removing one of the correlated variables, combining them into a single variable, or using regularization techniques.

▼ 12. How can the linear regression model's accuracy be improved?

There are several ways to improve the accuracy of a linear regression model:

1. ***Feature selection:*** Choosing the most relevant features (independent variables) that have the most impact on the dependent variable can improve the accuracy of the model.

2. ***Data cleaning and preprocessing:*** Outliers, missing data, and irrelevant information can affect the accuracy of a regression model. Removing or imputing missing data, dealing with outliers, and performing feature scaling can help improve the accuracy of the model.

3. ***Regularization:*** Regularization techniques like Ridge and Lasso regression can help reduce overfitting and improve the accuracy of the model.

4. ***Cross-validation:*** Cross-validation techniques can help evaluate the performance of the model and improve its accuracy by tuning the hyperparameters.

5. ***Model selection:*** Choosing the appropriate model for the data can improve the accuracy of the model. For example, using a polynomial regression model instead of a linear regression model for data with a nonlinear relationship can improve accuracy.

6. ***Increasing sample size:*** Increasing the sample size can reduce the impact of random variation and improve the accuracy of the model.

7. ***Ensemble techniques:*** Using ensemble techniques like bagging, boosting, and stacking can improve the accuracy of the model by combining the predictions of multiple models.

▼ 13. Using an example, describe the polynomial regression model in detail.

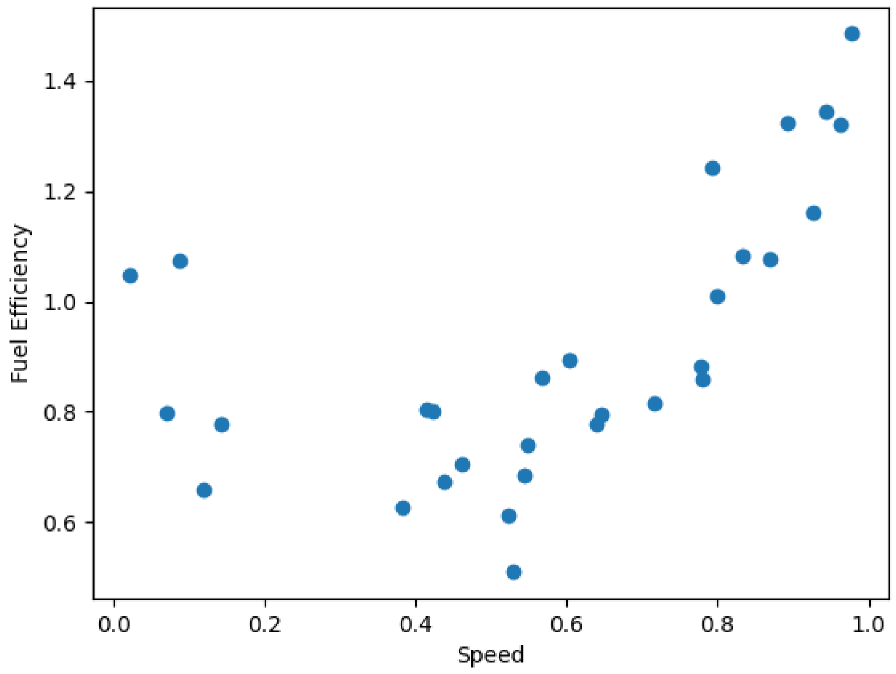
Polynomial regression is a type of regression analysis that models the relationship between a dependent variable and one or more independent variables as an nth-degree polynomial. It can be used to model non-linear relationships between the variables.

Let's consider an example where we have a dataset of a car's fuel efficiency (dependent variable) as a function of its speed (independent variable). A simple linear regression model may not be a good fit, as the relationship between speed and fuel efficiency may not be linear. In this case, we can use a polynomial regression model to better capture the relationship.

To start, we can plot the data points to get an idea of the relationship between the two variables.

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

# Generate some random data
np.random.seed(0)
x = np.sort(np.random.rand(30))
y = 1 - 2*x + 3*x**2 - 0.5*x**3 + np.random.randn(30)/10

# Plot the data points
plt.scatter(x, y)
plt.xlabel('Speed')
plt.ylabel('Fuel Efficiency')
plt.show()
```

The plot shows that there is a non-linear relationship between speed and fuel efficiency.

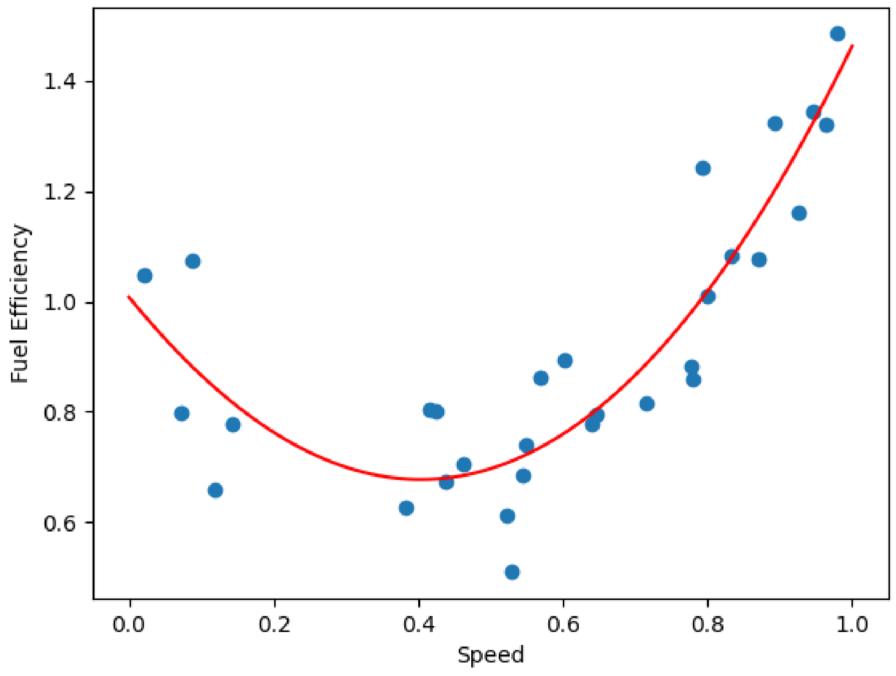


Next, we can create a polynomial regression model by fitting a polynomial equation to the data. The degree of the polynomial can be chosen based on the complexity of the relationship. In this example, we can use a cubic polynomial equation.

```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

# Create polynomial features
poly = PolynomialFeatures(degree=3)
X_poly = poly.fit_transform(x.reshape(-1, 1))
```

```
# Fit a linear regression model
lin_reg = LinearRegression()
lin_reg.fit(X_poly, y)

# Plot the polynomial regression line
X_plot = np.linspace(0, 1, 100)
X_plot_poly = poly.transform(X_plot.reshape(-1, 1))
y_plot = lin_reg.predict(X_plot_poly)

plt.scatter(x, y)
plt.plot(X_plot, y_plot, color='red')
plt.xlabel('Speed')
plt.ylabel('Fuel Efficiency')
plt.show()
```

The `PolynomialFeatures` transformer is used to create polynomial features up to the specified degree. The `LinearRegression` model is then fit to the transformed data. Finally, we can plot the polynomial regression line along with the data points.



The polynomial regression model provides a better fit to the data than a simple linear regression model would. However, it is important to note that increasing the degree of the polynomial can lead to overfitting and poor performance on new data. Therefore, the degree of the polynomial should be chosen carefully based on the trade-off between bias and variance.

▼ 14. Provide a detailed explanation of logistic regression.

Logistic regression is a type of regression analysis used to model the relationship between a binary dependent variable and one or more independent variables. It is commonly used in predicting the probability of an event or outcome.

The dependent variable in logistic regression is a binary variable that takes on two values, typically represented by 0 and 1. The independent variables can be continuous, binary, or categorical. The goal of logistic regression is to estimate the probability of the dependent variable taking on the value of 1, given the values of the independent variables.

The logistic regression model uses the logistic function, also known as the sigmoid function, to estimate the probability of the dependent variable taking on the value of 1. The logistic function is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

where z is the linear combination of the independent variables, weighted by their respective coefficients:

$$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p$$

The coefficients $\beta_0, \beta_1, \beta_2, \ldots, \beta_p$ are estimated using maximum likelihood estimation (MLE), which is a statistical method used to find the values of the coefficients that maximize the likelihood of the observed data.

Once the coefficients are estimated, the logistic function can be used to predict the probability of the dependent variable taking on the value of 1, given the values of the independent variables.

▼ 15. What are the logistic regression assumptions?

Logistic regression has several assumptions, including:

1. ***Binary dependent variable:*** The dependent variable should be binary or dichotomous, with only two possible outcomes.
2. ***Linearity of independent variables:*** The relationship between the independent variables and the log odds of the dependent variable should be linear.
3. ***Independence of observations:*** Each observation in the data should be independent of the others.
4. ***No multicollinearity:*** The independent variables should not be highly correlated with each other.
5. ***Large sample size:*** A minimum sample size is required to ensure the statistical power of the analysis.
6. ***Non-zero variance of independent variables:*** The independent variables should have some variability, or else the model will not be able to estimate their effects.
7. ***Absence of outliers:*** Extreme values or outliers in the data can affect the estimated coefficients and reduce the accuracy of the model.

▼ 16. Go through the details of maximum likelihood estimation.

Maximum likelihood estimation (MLE) is a statistical method used to estimate the parameters of a model by maximizing the likelihood function. The likelihood function represents the probability of observing the given data, given a specific set of model parameters.

The basic idea behind MLE is to find the values of the parameters that maximize the likelihood function. This is achieved by finding the derivative of the likelihood function with respect to each of the parameters, and setting them equal to zero. The resulting set of equations can then be solved for the parameter values that maximize the likelihood function.

For example, let's consider a simple linear regression model:

$$y = \beta_0 + \beta_1 x + \varepsilon$$

where y is the dependent variable, x is the independent variable, β_0 and β_1 are the intercept and slope parameters, and ε is the error term.

To estimate the parameters β_0 and β_1, we need to find the values that maximize the likelihood function. Assuming that the errors ε are normally distributed with mean zero and constant variance σ^2, the likelihood function can be written as:

$L(\beta_0, \beta_1, \sigma^2) = \left(2\pi\sigma^2\right)^{-n/2} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y_i - \beta_0 - \beta_1 x_i\right)^2\right)$

where n is the sample size and Σ denotes the sum over all observations.

Taking the derivative of the likelihood function with respect to β0 and β1, we obtain:

$$\frac{\partial L}{\partial \beta_0} = \sum \frac{y_i - \beta_0 - \beta_1 x_i}{\sigma^2}$$

$$\frac{\partial L}{\partial \beta_1} = \sum \frac{(y_i - \beta_0 - \beta_1 x_i) x_i}{\sigma^2}$$

Setting these equations equal to zero and solving for β0 and β1, we obtain the maximum likelihood estimates:

$$\beta_1 = \frac{\sum\limits_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum\limits_{i=1}^{n} (x_i - \bar{x})^2}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

where x̄ and ȳ are the sample means of x and y, respectively.

The estimation of σ^2 can also be obtained by taking the derivative of the likelihood function with respect to σ^2 and setting it equal to zero. The resulting equation gives the maximum likelihood estimate of σ^2:

$$\sigma^2 = \frac{\sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2}{n}$$

MLE has several desirable properties, including consistency (i.e., the estimates converge to the true values as the sample size increases), asymptotic normality (i.e., the estimates are normally distributed as the sample size increases), and efficiency (i.e., the estimates are the most efficient among all unbiased estimators). However, MLE can be sensitive to the initial values of the parameters and may require numerical optimization techniques to find the maximum of the likelihood function.