Linear Regression

Linear Regression related terms

Note: Linear regression is a foundational algorithm in machine learning, providing a solid understanding of the core concepts. It's essential to grasp these terms to delve deeper into more complex models and techniques.

Linear Regression

- **Definition:** A statistical method used to model the relationship between a dependent variable (target) and one or more independent variables (predictors) by fitting a linear equation to the observed data.
- **Goal:** To find the best-fitting line that minimizes the difference between the predicted values and the actual values.

Gradient Descent

- **Definition:** An optimization algorithm used to find the minimum of a function. In machine learning, it's used to find the optimal parameters (weights and biases) of a model.
- **Process:** It iteratively adjusts the parameters in the direction of steepest descent (negative gradient) of the error function until a minimum is reached.

Gradient Descent Optimizer

- **Definition:** An algorithm that implements the gradient descent process.
- **Role:** It determines how the parameters are updated at each iteration.
- **Examples:** Stochastic Gradient Descent (SGD), Adam, RMSprop.

Best Fit Line: y = mx + c

- **Definition:** The straight line that best represents the relationship between two variables on a scatter plot.
- **Equation:** y = mx + c, where:
  - y is the dependent variable
  - x is the independent variable
  - m is the slope of the line
  - c is the y-intercept (the value of y when x is 0)

Slope (m)

- **Definition:** The rate of change of the dependent variable with respect to the independent variable. It represents the steepness of the line.

- **Interpretation:** A positive slope indicates a positive relationship between the variables, while a negative slope indicates a negative relationship.

Intercept (c)

- **Definition:** The value of the dependent variable when the independent variable is zero. It's the point where the line crosses the y-axis.

Error (Residual)

- **Definition:** The difference between the actual value of the dependent variable and the predicted value from the regression line.
- **Role:** The goal of linear regression is to minimize the sum of squared errors.

Global Minima

- **Definition:** The lowest point of a function over its entire domain.
- **Goal:** In gradient descent, the aim is to find the global minimum of the error function to achieve the best model performance.

Mathematical Intuition of Linear Regression

- **Objective:** To find the values of m and c that minimize the sum of squared errors between the observed data points and the predicted values on the line.
- **Method:**
  1. Initialize random values for m and c.
  2. Calculate the error for each data point.
  3. Calculate the gradient of the error function with respect to m and c.
  4. Update m and c using gradient descent.
  5. Repeat steps 2-4 until the error converges to a minimum.
- **Underlying principle:** The line that minimizes the sum of squared errors is the best fit line.

EDA (Exploratory Data Analysis)

- **Definition**: EDA is the process of analyzing and summarizing the main characteristics of a dataset, often using visual methods. It helps in understanding the underlying structure, detecting anomalies or outliers, identifying patterns, and making decisions on how to handle data before applying machine learning models.
- **Purpose**: EDA helps in making informed decisions about feature selection, transformation, and model selection, ultimately improving the performance of machine learning models.

Mean Squared Error (MSE)

- **Definition**: MSE is a common loss function used in regression models to measure the average of the squares of the errors (i.e., the differences between the actual and predicted values). It gives a higher weight to larger errors.
  - Formula:

  `MSE = (1/n) * Σ(y_true - y_pred)^2`
  where:
  - n is the number of data points
  - y_true is the true value
  - y_pred is the predicted value
- **Purpose**: MSE is used to evaluate the performance of regression models, with lower values indicating better model accuracy.

Mean Absolute Error (MAE)

- **Definition**: MAE is another loss function used in regression, which measures the average of the absolute differences between actual and predicted values. Unlike MSE, it treats all errors equally without giving extra weight to larger errors.

  - Formula:

  `MAE = (1/n) * Σ|y_true - y_pred|`
- **Purpose**: MAE provides a straightforward measure of model accuracy, making it easier to interpret than MSE.

Outliers

- **Definition**: Outliers are data points that significantly differ from the other observations in a dataset. They can result from measurement errors, data entry errors, or they might represent a real anomaly in the data.
- **Purpose**: Identifying and understanding outliers is crucial as they can heavily influence the results of a model, potentially leading to misleading conclusions.

BoxPlot

- **Definition**: A BoxPlot is a graphical representation used in EDA to display the distribution of a dataset based on five summary statistics: minimum, first quartile (Q1), median, third quartile (Q3), and maximum. It also identifies outliers by showing points outside the whiskers of the box.
- **Purpose**: BoxPlots help in visualizing the spread, skewness, and potential outliers in the data, aiding in better data understanding during EDA.

How MAE is Robust to Outliers

- **Explanation**: MAE is considered robust to outliers because it calculates the average absolute error without squaring the differences. This means that large errors (from outliers) do not disproportionately affect the overall error measure, unlike MSE, where outliers can have a significant impact due to the squaring of errors.

- **Purpose**: Using MAE can provide a more balanced view of model performance, especially in datasets with outliers, making it a useful metric in scenarios where outliers are expected.

How They Help in EDA

- **EDA**: By identifying outliers using tools like BoxPlots and understanding error metrics like MAE and MSE, you can make more informed decisions on how to preprocess your data. For example, you might decide to remove or transform outliers or choose a specific model evaluation metric based on the nature of your data. This process improves the quality of data fed into machine learning models, leading to better performance and more reliable results.

These explanations should help you take clear and concise notes on these important machine learning concepts.

Learning Rate

What is Learning Rate?

- A hyperparameter in machine learning algorithms, specifically optimization algorithms like gradient descent.
- Determines the step size taken in the direction of the negative gradient during each iteration.
- Controls how quickly the model's parameters are updated.

Where is it Used?

- Primarily used in gradient-based optimization algorithms, including:
  - Gradient Descent
  - Stochastic Gradient Descent (SGD)
  - Adam
  - RMSprop

Impact of Learning Rate

- **Too Low Learning Rate:**
  - Slow convergence: Model takes a long time to reach the optimal solution.
  - Risk of getting stuck in local minima.
  - Inefficient training process.
- **Just Right Learning Rate:**
  - Optimal convergence: Model reaches the optimal solution efficiently.
  - Good balance between speed and accuracy.
- **Too High Learning Rate:**
  - Divergence: Model's parameters oscillate and fail to converge.

    - Overfitting: Model becomes too sensitive to training data.
    - Instability in the training process.

Note: Finding the optimal learning rate is crucial for model performance. Techniques like learning rate scheduling and adaptive learning rate methods can help in fine-tuning the learning rate during training.

Assumptions in Linear Regression

Linear regression is a powerful statistical tool, but it relies on certain assumptions about the data. These assumptions are crucial for the validity of the model and its inferences. Let's explore them:

1. Linearity

- **Assumption:** There is a linear relationship between the dependent variable and the independent variables.
- **Implications:** If the relationship is not linear, the model will be inaccurate.
- **Checking:** Scatter plots can help visualize the relationship.

2. Independence

- **Assumption:** The observations are independent of each other.
- **Implications:** If observations are correlated, the model's estimates might be biased.
- **Checking:** Time series data often violates this assumption. It can be checked using autocorrelation plots.

3. Homoscedasticity

- **Assumption:** The variance of the error terms is constant across all levels of the independent variable.
- **Implications:** If the variance is not constant (heteroscedasticity), the model's estimates might be inefficient.
- **Checking:** Residual plots can help identify non-constant variance.

4. Normality

- **Assumption:** The error terms follow a normal distribution.
- **Implications:** This assumption is important for inference and hypothesis testing.
- **Checking:** Histograms and Q-Q plots of residuals can assess normality.

5. No Multicollinearity

- **Assumption:** There is no perfect linear relationship between the independent variables.

- **Implications:** Multicollinearity can inflate standard errors and make it difficult to interpret the model.
- **Checking:** Correlation matrix, Variance Inflation Factor (VIF), and Tolerance can be used.

6. No Autocorrelation

- **Assumption:** The error terms are uncorrelated with each other.
- **Implications:** Autocorrelation often occurs in time series data and can lead to inefficient estimates.
- **Checking:** Durbin-Watson test can be used.

Note: While these assumptions are important, it's essential to remember that no real-world data perfectly meets all assumptions. The key is to identify potential violations and address them appropriately, either through data transformations, using robust methods, or considering alternative models.

Linear Regression

Definition: Linear Regression is a statistical method used in machine learning to model the relationship between a dependent variable (target) and one or more independent variables (features). The goal is to find the best-fitting linear equation that predicts the target variable from the features.

Mathematical Equation:

- For a single feature (Simple Linear Regression): (y = mx + c)
- For multiple features (Multiple Linear Regression): (y = b_0 + b_1x_1 + b_2x_2 + \dots + b_nx_n)

Where:

- (y) is the predicted value.
- (x) is the input feature.
- (m) or (b_1, b_2, \dots, b_n) are the coefficients (slopes).
- (c) or (b_0) is the intercept (constant term).

Types of Linear Regression

1. **Simple Linear Regression**

- **Definition**: Simple linear regression models the relationship between a single independent variable (feature) and a dependent variable (target) using a straight line.
- **Equation**: (y = mx + c)
- **Example**:
  - Predicting house prices based on the size of the house.

In [1]:
```
from sklearn.linear_model import LinearRegression
import numpy as np

# Sample data
X = np.array([[1400], [1600], [1700], [1875], [1100]])  # Size of the hou
y = np.array([245000, 312000, 279000, 308000, 199000])  # Price of the ho

# Create and train the model
model = LinearRegression()
model.fit(X, y)

# Predict the price of a new house
new_house_size = np.array([[1500]])
price_prediction = model.predict(new_house_size)
print(f"Predicted price: {price_prediction[0]:.2f}")
```

```
Predicted price: 263525.74
```

2. Multiple Linear Regression

- **Definition**: Multiple linear regression models the relationship between two or more independent variables (features) and a dependent variable (target). The goal is to find the best-fitting linear equation that describes the relationship.
- **Equation**: y = b0 + b1x1 + b2x2 + ... + bnxn
- **Example**:
  - Predicting house prices based on multiple factors like size, number of bedrooms, and location.
  - **Code Example using scikit-learn**:

In [2]:
```
from sklearn.linear_model import LinearRegression
import numpy as np
# Sample data
X = np.array([[1400, 3], [1600, 4], [1700, 3], [1875, 4], [1100, 2]])  #
y = np.array([245000, 312000, 279000, 308000, 199000])  # Price of the ho
# Create and train the model
model = LinearRegression()
model.fit(X, y)
# Predict the price of a new house
new_house = np.array([[1500, 3]])
price_prediction = model.predict(new_house)
print(f"Predicted price: {price_prediction[0]:.2f}")
```

```
Predicted price: 259098.43
```

3. Polynomial Regression

- **Definition**: Polynomial regression is an extension of linear regression where the relationship between the independent variable and the dependent variable is modeled as an nth degree polynomial. It can capture non-linear relationships between the features and the target.
- **Equation**: y = b0 + b1x1 + b2x1^2 + ... + bnx1^n
- **Example**:
  - Predicting the trajectory of a projectile where the relationship between time and distance is quadratic.

```
In [3]: from sklearn.linear_model import LinearRegression
        from sklearn.preprocessing import PolynomialFeatures
        import numpy as np
        # Sample data
        X = np.array([[1], [2], [3], [4], [5]])  # Time
        y = np.array([1, 4, 9, 16, 25])
        # Distance (quadratic relationship)
        # Transform the data to include polynomial features
        poly = PolynomialFeatures(degree=2)
        X_poly = poly.fit_transform(X)
        # Create and train the model
        model = LinearRegression()
        model.fit(X_poly, y)
        # Predict the distance at a new time
        new_time = np.array([[6]])
        new_time_poly = poly.transform(new_time)
        distance_prediction = model.predict(new_time_poly)
        print(f"Predicted distance: {distance_prediction[0]:.2f}")
```

```
Predicted distance: 36.00
```

Summary for Notes

- **Linear Regression**: A method to model the relationship between a dependent variable and one or more independent variables using a linear equation.
  - **Simple Linear Regression**: Models a single feature's impact on the target (e.g., house price prediction based on size).
  - **Multiple Linear Regression**: Models multiple features' impacts on the target (e.g., house price prediction based on size, bedrooms, and location).
  - **Polynomial Regression**: Captures non-linear relationships by modeling the data with an nth degree polynomial (e.g., predicting projectile trajectory).

R-Squared (R^2)

Definition: R-Squared, also known as the coefficient of determination, is a statistical measure that indicates the proportion of the variance in the dependent variable (target) that is predictable from the independent variables (features). It represents how well the regression model fits the observed data.

Mathematical Formula: R^2 = 1 - (SSres / SStot)

Where:

- SSres (Residual Sum of Squares): Sum of the squared differences between the observed values and the predicted values.
- SStot (Total Sum of Squares): Sum of the squared differences between the observed values and the mean of the observed values.

Interpretation:

- (R^2 = 1): The model perfectly predicts the target variable, meaning all the variance in the target is explained by the features.
- (R^2 = 0): The model does not explain any of the variance in the target variable, meaning the model's predictions are as good as the mean of the target variable.
- **Value Range**: (R^2) ranges from 0 to 1, where higher values indicate a better fit.

Use in Performance Metric:

- R-Squared helps determine how well the independent variables explain the variability of the dependent variable. It provides a quick assessment of model performance, especially in linear regression models.

Adjusted R-Squared

Definition: Adjusted R-Squared is a modified version of R-Squared that adjusts for the number of independent variables in the model. Unlike R-Squared, which can increase as more variables are added to the model (even if they are not significant), Adjusted R-Squared accounts for the model's complexity and only increases if the added variables improve the model.

Mathematical Formula: Adjusted R^2 = 1 - ((1-R^2)(n-1)/(n - p - 1))

Where:

- (n) is the number of observations.
- (p) is the number of predictors (independent variables).
- (R^2) is the R-Squared value.

Interpretation:

- **Penalizes Complexity**: Adjusted R-Squared decreases if the added variables do not improve the model, helping to avoid overfitting.
- **Better Comparison**: It allows for a more accurate comparison between models with a different number of predictors, as it accounts for the model's complexity.

Use in Performance Metric:

- Adjusted R-Squared is particularly useful in multiple regression models where the number of predictors may vary. It helps in selecting the right model by balancing model fit and complexity.

Summary for Notes

- **R-Squared (R²)**: Measures the proportion of variance in the target variable explained by the independent variables. `Formula: R^2 = 1 - (SSres / SStot)` A higher (R^2) indicates a better model fit.
- **Adjusted R-Squared**: Adjusts the R-Squared value based on the number of predictors, penalizing the addition of non-significant variables. `Formula:`

`Adjusted R^2 = 1 – ((1–R^2)(n–1)/(n – p – 1))` It provides a more accurate performance metric, especially for multiple regression models.