Detailed Explanation of Regression

Regression is a fundamental statistical and machine learning technique used to model and analyze the relationship between a dependent variable (target) and one or more independent variables (features). The goal is to predict or estimate the dependent variable based on the values of the independent variables.

Types of Regression

1. Linear Regression

Linear regression assumes a linear relationship between the dependent variable y and the independent variables x.

Equation:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon$$

Where:

- β_0: Intercept (constant term),
- $\beta_1, \beta_2, \ldots, \beta_p$: Coefficients,
- ϵ: Error term.

Objective: Minimize the residual sum of squares (RSS):

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

Where $\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}$.

Example: Predicting house prices based on features like area, number of bedrooms, etc.

2. Logistic Regression

Logistic regression is used for binary classification problems. It models the probability that the dependent variable belongs to one of the two classes.

Equation:

$$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p)}}$$

The model predicts $y = 1$ if $P(y = 1|x) \geq 0.5$, and $y = 0$ otherwise.

Example: Predicting whether a customer will buy a product based on their age and income.

3. Polynomial Regression

Polynomial regression extends linear regression by including polynomial terms of the independent variables.

Equation:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d + \epsilon$$

Example: Modeling nonlinear relationships, such as the growth rate of a population over time.

4. Ridge and Lasso Regression (Regularized Regression)

Regularized regression methods introduce penalties to prevent overfitting.

1. **Ridge Regression:** Adds an $L2$-norm penalty:

$$RSS + \lambda \sum_{j=1}^{p} \beta_j^2$$

2. **Lasso Regression:** Adds an $L1$-norm penalty:

$$RSS + \lambda \sum_{j=1}^{p} |\beta_j|$$

5. Multiple Regression

Multiple regression involves predicting a dependent variable using multiple independent variables.

Equation:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon$$

6. Nonlinear Regression

Nonlinear regression models relationships where the dependent variable is a nonlinear function of the independent variables.

Example: Modeling the growth of bacteria in a lab using an exponential function.

Key Metrics to Evaluate Regression Models

1. **Mean Absolute Error (MAE):**

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

2. **Mean Squared Error (MSE):**

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

3. **R-Squared (R^2):**

$$R^2 = 1 - \frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}}$$

Where:

- SS_{res}: Residual sum of squares,
- SS_{tot}: Total sum of squares.

Python Implementations

1. Linear Regression

Listing 1: Python Implementation: Linear Regression

```python
from sklearn.linear_model import LinearRegression
import numpy as np

# Example data
X = np.array([[1], [2], [3], [4], [5]])  # Independent variable
y = np.array([3, 4, 2, 5, 6])            # Dependent variable

# Fit linear regression model
model = LinearRegression().fit(X, y)

# Coefficients and predictions
print(f"Intercept: {model.intercept_}")
print(f"Slope: {model.coef_[0]}")
y_pred = model.predict(X)
print(f"Predicted values: {y_pred}")
```

2. Logistic Regression

Listing 2: Python Implementation: Logistic Regression

```python
from sklearn.linear_model import LogisticRegression

# Example data
X = np.array([[1], [2], [3], [4], [5]])  # Independent variable
y = np.array([0, 0, 0, 1, 1])            # Binary target variable

# Fit logistic regression model
logreg = LogisticRegression()
logreg.fit(X, y)

# Coefficients and predictions
print(f"Coefficients: {logreg.coef_}")
y_pred = logreg.predict(X)
print(f"Predicted values: {y_pred}")
```

3. Polynomial Regression

Listing 3: Python Implementation: Polynomial Regression

```python
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
import numpy as np

# Example data
X = np.array([[1], [2], [3], [4], [5]])  # Independent variable
y = np.array([1, 4, 9, 16, 25])          # Dependent variable (nonlinear
    relationship)

# Transform to polynomial features
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)

# Fit polynomial regression model
model = LinearRegression().fit(X_poly, y)
print(f"Coefficients: {model.coef_}")
y_pred = model.predict(X_poly)
print(f"Predicted values: {y_pred}")
```

4. Ridge Regression

Listing 4: Python Implementation: Ridge Regression

```python
from sklearn.linear_model import Ridge

# Example data
X = np.array([[1], [2], [3], [4], [5]])  # Independent variable
y = np.array([3, 4, 2, 5, 6])            # Dependent variable

# Fit ridge regression model
ridge = Ridge(alpha=1.0)
ridge.fit(X, y)

print(f"Coefficients: {ridge.coef_}")
print(f"Intercept: {ridge.intercept_}")
```

5. Evaluating a Regression Model

Listing 5: Python Implementation: Evaluating a Regression Model

```python
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# True values and predictions
y_true = np.array([3, 4, 2, 5, 6])
y_pred = np.array([2.8, 3.9, 2.1, 5.2, 6.1])

# Metrics
mae = mean_absolute_error(y_true, y_pred)
mse = mean_squared_error(y_true, y_pred)
r2 = r2_score(y_true, y_pred)

print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-Squared (R^2): {r2}")
```

Applications of Regression

1. **Finance:** Predicting stock prices or market trends.
2. **Healthcare:** Estimating patient outcomes or disease progression.
3. **Marketing:** Predicting sales based on ad spend and other factors.
4. **Real Estate:** Predicting property prices based on location, size, and amenities.
5. **Natural Sciences:** Modeling the relationship between variables in experiments.