Performance Measures and Cross-validation

Cheng Peng

West Chester University

Contents

1 Introduction **1**

2 Regression Modeling Basics **2**

2.1 Modeling Process . . . 2

2.2 Linear Regression Models . . . 3

2.2.1 **Objectives** . . . 3

2.2.2 **Structure** . . . 3

2.2.3 **Assumptions** . . . 3

2.3 Logistic Regression Model . . . 4

2.3.1 **Objectives** . . . 4

2.3.2 **Structure** . . . 4

2.3.3 **Assumptions** . . . 4

2.4 Parameter Interpretation . . . 5

2.4.1 Linear Regression . . . 5

2.4.2 Logistic Regression . . . 9

2.5 Paremeter Estimation . . . 12

2.5.1 Concepts of Likeliood *(Optional)* . . . 12

2.5.2 Normal Linear Regression Likelihood *(Optional)* . . . 12

2.5.3 Logistic Regression Likelihood *(Optional)* . . . 13

2.5.4 Finding MLE Using Software Programs . . . 14

3 Performance Measures **17**

4 Cross-Validation **18**

4.1 Common Types of CV . . . 18

4.2 Essence ang Logic of CV . . . 18

4.3 Purpose of CV . . . 19

1 Introduction

Machine learning (ML) has revolutionized various industries by enabling the development of systems capable of learning from data to make accurate predictions or decisions. However, the success of any ML model hinges on its ability to perform well not only on the data it was trained on but also on unseen data. This is where performance measures and cross-validation play crucial roles. Together, they provide a robust framework for assessing and validating machine learning models, ensuring their reliability and generalizability.

2 Regression Modeling Basics

Regression modeling and machine learning algorithms are practically important because they enable data-driven decision-making, predictions, and insights across diverse industries. Their importance lies in their ability to process and analyze large volumes of data, uncover patterns, and optimize processes, leading to better outcomes in business, healthcare, technology, and more. Among many regression models, linear regression and logistic regression models are the two most important family models in both classical statistics for association analysis under relatively strong distributional assumptions and modern machine learning fields for prediction with less assumptions.

The essence of regression modeling lies in understanding the relationship between a dependent variable and one or more independent variables, or in making predictions based on this relationship.

Assessing the relationship: In this case, regression helps identify how changes in the independent variables (predictors) influence the dependent variable. It is often used to quantify the strength, direction, and nature of the relationship.

Prediction: Regression models can also be used for predicting the value of the dependent variable given new or unseen data for the independent variables. Once the model is trained on historical data, it can be applied to make predictions for future data.

2.1 Modeling Process

The process of regression modeling is an iterative process that includes several key stages, Each step involves feedback loops, enabling the modeler to refine assumptions, data, and model specifications for optimal performance and meaningful insights.

The iterative nature of regression modeling ensures that the model evolves to better fit the data and the problem to improve its performance and alignment with the problem at hand. Here's a breakdown of the steps

- **Problem Definition**: Clearly define the objective of the regression model, including the target variable and predictors and **revisit** problem assumptions based on data understanding or business feedback.
- **Data Collection and Preparation** Collect,clean and preprocess data by handling missing values, outliers, and inconsistent entries. Transform features if necessary (e.g., logarithmic transformations for skewed data). Split data into training, validation, and testing sets. **Refine** data preprocessing steps if initial results indicate poor data quality.
- **Exploratory Data Analysis (EDA)**: Visualize data to understand relationships and distributions. Identify potential correlations between predictors and the target variable. Check for multicollinearity and other issues. **Adjust** features or identify new ones based on insights.
- **Model Specification**: Choose the type of regression model (e.g., linear regression, polynomial regression) and specify the form of the model and include initial predictors. **Test different forms** of the model (e.g., adding interaction terms, transforming variables) based on feedback from residual analysis.
- **Model Fitting**: Fit the model to the training data using appropriate estimation methods (e.g., Ordinary Least Squares, Maximum Likelihood Estimation). **Tune model parameters or adjust features** if the initial model does not perform well.
- **Model Evaluation**: Evaluate model performance on the training and validation sets using metrics like R-squared, Mean Squared Error (MSE), or Mean Absolute Error (MAE) and check diagnostic plots for residuals to ensure assumptions (linearity, independence, homoscedasticity, and normality) are met. **Modify** the model if diagnostic checks reveal violations of assumptions.
- **Feature Selection**: Use techniques like backward elimination, forward selection, or regularization (Lasso, Ridge) to identify the most important predictors. **Re-run** the model with selected features and reassess its performance.

- **Model Validation and Testing**: Validate the model on a separate validation set to test its ability to generalize to unseen data, perform cross-validation to assess model stability and robustness, and evaluate the model on the test set for final performance metrics. **Adjust** the model based on validation results to improve generalization.
- **Interpretation and Insight Generation**: Interpret the coefficients and their significance to draw actionable insights and ensure the model aligns with domain knowledge and business expectations. **Refine** the model to improve interpretability or accommodate new insights.
- **Deployment and Monitoring**: Deploy the model in production for predictions or decision-making and continuously monitor model performance over time to ensure it remains accurate and relevant. **Update or retrain** the model as new data becomes available or when performance degrades.

2.2 Linear Regression Models

Linear regression modeling is a foundational technique in statistics and machine learning used to describe the relationship between one or more independent variables (predictors) and a dependent variable (response). Below are the fundamentals of linear regression modeling.

2.2.1 Objectives

Linear regression aims to model the relationship between variables by fitting a linear equation to observed data. The goal is to either predict the dependent variable y based on the values of independent variables $(x_1, x_2, \cdots, x_k)$ or assess the relationship between y and predictor variables in the linear and nonlinear forms.

2.2.2 Structure

The general form of the multiple linear regression model is given by

$$y = f(x_1, x_2, \cdots, x_k) + \epsilon$$

where $f(x_1, x_2, \cdots, x_k)$ is a deterministic expression of predictor variables $x_1, x_2, \cdots, x_k$ and some unknown parameters (coefficients). ϵ is a random variable that follows a certain probability distribution. If the $f(x)$ is a **linear function of unknown parameters** and $\epsilon \sim N(0, \sigma)$, the corresponding model is called **linear models**. Here are some examples

- $f(\cdot)$ is a linear combination of predictor feature variables in the following form

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \epsilon, \text{ where } \epsilon \sim N(0, \sigma).$$

- $f(\cdot)$ is a polynomial function of predictor variables. For example,

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{11} x_1^2 + \beta_{22} x_2^2 + \beta_{12} x_1 x_2 + \epsilon, \text{ where } \epsilon \sim N(0, \sigma).$$

is a polynomial regression model. **Caution**: *The above polynomial (i.e., quadratic) regression is a linear regression model.*

2.2.3 Assumptions

Every models and algorithms have some explicit and implicit assumptions to be checked and validated before reporting the model.

- **Data Set**: the data must be IID.
- **Structure**: the function relationship between the response and the predictor variables must be correctly specified.

- **Distribution of Response Variable**: $Y \sim N[f(x_1, x_2, \cdots, x_k), \sigma]$ which is equivalent to $\epsilon \sim N(0, \sigma)$. The notation $N(\mu, \sigma)$ stands for normal distribution with mean μ and standard deviation σ. **These assumptions are crucial to estimate the regression coefficients, define valid p-values for significance tests of regression coefficients, and construct prediction intervals using t-distributions.**

2.3 Logistic Regression Model

Logistic regression is a statistical method used to model the relationship between a dependent variable (binary or categorical) and one or more independent variables. It predicts the probability of the dependent variable belonging to a particular category (e.g., success/failure, 1/0) by using a logistic function (sigmoid curve), which maps predicted values to a range between 0 and 1.

2.3.1 Objectives

Logistic regression aims to model the relationship between the **binary** response variable and other feature variables by fitting a nonlinear equation (of predictor features) to observed data. The goal is to either predict the dependent variable y based on the values of independent variables $(x_1, x_2, \cdots, x_k)$ or assess the relationship between y and predictor variables in the linear and nonlinear forms.

2.3.2 Structure

Assume that the response variable takes character values `0` and `1` and $P(Y = "1")$ is the probability of observing the **bigger value of the response** (in alphabetical order). The general form of the logistic regression is given by

$$P(Y = 1) = \frac{e^{f(x_1, x_2, \cdots, x_k)}}{1 + e^{f(x_1, x_2, \cdots, x_k)}}.$$

The deterministic function $f(\cdot)$ can take either linear or polynomial expression of predictor feature variables. For example, if $f(x_1, x_2, \cdots, x_k) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k$, the explicit logistic model is given by

$$P(Y = 1) = \frac{e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k}}{1 + e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k}}.$$

or equivalently

$$\log \frac{P(Y = 1)}{1 - P(Y = 1)} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k.$$

The above formulation of logistic regression model has important terms: **The odds of observing** $Y = 1$ is $P(Y = 1)/[1 - P(Y = 1)]$. The left hand of the above formulation is called logarithmic odds of observing $Y = 1$. That is,

$$\text{log odds of (Y = 1)} = \log \frac{P(Y = 1)}{1 - P(Y = 1)}.$$

2.3.3 Assumptions

Different models have different assumptions. Logistic regression models have the following assumptions.

- **Data Set**: the data must be IID and the sample size must be large.
- **Structure**: the function relationship between $P(Y = 1)$ and the predictor variables must be correctly specified. That is, $f(\cdot)$ **as a function of the predictor variables** must be correctly specified.

- **error Distribution**: $Y \sim Binom(p)$ where $p = P(Y = 1)$ is a binomial distribution. **These assumptions are crucial to estimate the regression coefficients, define valid p-values for significance tests of regression coefficients, and construct prediction intervals using t-distributions.**

2.4 Parameter Interpretation

In predictive analysis and classification applications, interpretation of the regression coefficient is not the the focus but the accuracy of the underlying model and algorithms. However, in association analysis, the interpretation of regression coefficients in the final model is critical for ensuring that models are not only correct but also reliable and responsible in their applications.

The interpretability of a regression coefficients is dependent on the structure of the underlying regression model which is also one of the criteria in assessing goodness of a model. This subsection discusses the the interpretation of regression coefficients of linear and logistic regression models.

2.4.1 Linear Regression

We provide interpretation of regression coefficient in three different scenarios.

- **First Order Linear Regression**

Recall that the first order linear regression model with k predictor feature variable has the following form

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_i x_i + \cdots + \beta_k x_k + \epsilon, \quad \text{where} \quad \epsilon \sim N(0, \sigma).$$

where $1 \le i \le k$ and $\epsilon \sim N(0, \sigma)$. Let

$$y^{(i,0)} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_i (x_i + 0) + \cdots + \beta_k x_k,$$

and

$$y^{(i,1)} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_i (x_i + 1) + \cdots + \beta_k x_k,$$

then

$$\beta_i = y^{(i,1)} - y^{(i,0)}.$$

The coefficient of x_i, representing the change in y for a one-unit increase in x_i, holding all other variables constant. **That is, under the same condition, if x_i increases one-unit, the corresponding change in y is the coefficient β_i.**

- **Polynomial Regressions with no Interaction**

Without loss of generality, we consider the following polynomial regression with one predictor variable x.

$$y = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3 + \cdots + \alpha_k x^k + \epsilon, \quad \text{where} \quad \epsilon \sim N(0, \sigma).$$

Interpreting polynomial regression coefficients in a practical sense can be challenging. The coefficients in a polynomial regression model correspond to the terms of the polynomial equation reflect the behavior of the change of the curvature of the (curve of the) polynomial function.

From practical perspective, we suggest,rather than interpreting coefficients directly, using plots to visualize the relationship between x and y.

- **Regression with 2nd Order Interaction Only**

We will not discuss high-order interaction terms in any polynomial regression model. The following is a end order polynomial regression model with an interaction term only

$$y = \gamma_0 + \gamma_1 x_1 + \gamma_2 x_2 + \gamma_{12} x_1 x_2 + \epsilon, \quad \text{where} \quad \epsilon \sim N(0, \sigma)$$

We fix x_2 and increase x_1 by one unit, we evaluate the change of the response y in the following.

$$y^{(x_1+1,x_2)} - y^{(x_1,x_2)} = [\gamma_0 + \gamma_1(x_1+1) + \gamma_2 x_2 + \gamma_{12}(x_1+1)x_2] - [\gamma_0 + \gamma_1 x_1 + \gamma_2 x_2 + \gamma_{12} x_1 x_2] = \gamma_1 + \gamma_{12} x_2$$

The coefficient γ_{12} indicates the extent to which the effect of x_1 on y depends on the value of x_2, and vice versa. A context-specific explanation of the interaction is given in the following agricultural example.

Consider a study examining the effects of fertilizer and water levels on crop yield.

- y: Crop yield (in tons per hectare).
- x_1: Amount of fertilizer applied (in kg per hectare).
- x_2: Amount of water supplied (in liters per hectare).

Assume the regression model is

$$\text{crop} = \gamma_0 + \gamma_1 \text{water amount} + \gamma_2 \times \text{fertilizer} + \gamma_{12} \text{fertilizer} \times \text{water amount}.$$

The interpretation: when we increase fertilizer by one kg and keep the same amount of water supply, the increment of the crop yield is given by

$$\text{The increment of crop yield} = \gamma_1 + \gamma_{12} \times \text{water amount (kg)}$$

A positive γ_{12} suggests that the benefits of adding fertilizer (x_1) on crop yield are amplified when water supply (x_2) is high. Similarly, increased water supply is more beneficial when sufficient fertilizer is applied. That is, doubling water supply while fertilizer levels are already high may lead to more than a proportional increase in yield.

Here's an example in R to view the interaction effect using a polynomial regression with one two-way interaction term.

```r
# Load necessary library
#library(ggplot2)
# Simulate data
set.seed(123)  # For reproducibility
n = 100
x1 = runif(n, 0, 10)
x2 = runif(n, 0, 10)
z = 3 + 2*x1 - 1.5*x2 + 0.5*x1^2 + 0.3*x1*x2 + rnorm(n, 0, 3)

# Combine into a data frame
data = data.frame(x1 = x1, x2 = x2, y = z)
data$x1x2 = x1*x2

# Fit a quadratic regression model with interaction term
model = lm(z ~ x1 + x2 + I(x1*x2), data = data)
# Summary of the model
#summary(model)
coef1 = coef(model)
data$z1 = coef1[1] + coef1[2]*x1 + coef1[3]*x2 + coef1[4]*data$x1x2
```

```r
##
model0 = lm(z~x1+x2, data = data)
coef0 = coef(model0)
data$z0 = coef0[1] + coef0[2]*x1 + coef0[3]*x2
```

```r
# Visualizing the quadratic regression surface
quad = plot_ly(data, x= ~x1, y= ~x2, z= ~z1,
               type='mesh3d',
               intensity = ~z1,
               text = "Interaction",
               colors= colorRamp(rainbow(5))
           )
lin.quad = add_trace(p = quad,
           z = data$z0,
           x = x1,
           y = x2,
           text = "No Interaction",
           type = "mesh3d") %>%
           hide_colorbar() %>%
           layout(title = 'Polynomial Regression Surface: No Interaction vs Interaction',
              margin = list(l = 5, r = 5, b = 50, t = 50, pad = 4))
lin.quad
```

Polynomial Regression Surface: No Interaction vs Interaction

2.4.2 Logistic Regression

Unlike linear regression model, the regression coefficients are directly related to the change of the response variable. In the logistic regression, the change of the values of predictor feature variables influences the probability of observing $Y = 1$. If express logistic regression as logarithm of odds of observing $(Y = 1)$, that is,

$$\log \mathrm{O} = \log \frac{P(Y=1)}{1-P(Y=1)} = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_k x_k.$$

$O =$ Odds of oberserving Y=1. Then we can mimic the way of interpreting the coefficients in linear regression, the i-th regression coefficient in the above multiple logistic regression model is expressed as

$$\alpha_i = \log \mathrm{O}^{(x_i+1)} - \log \mathrm{O}^{(x_i)} = \log \frac{\mathrm{O}^{(x_i+1)}}{\mathrm{O}^{(x_i)}}$$

We exponentiate both sides of the above equation, we have

$$\frac{\mathrm{O}^{(x_i+1)}}{\mathrm{O}^{(x_i)}} = e^{\alpha_i}, \quad \text{which is equivalent to} \quad \frac{\mathrm{O}^{(x_i+1)} - \mathrm{O}^{(x_i)}}{\mathrm{O}^{(x_i)}} = e^{\alpha_i} - 1.$$

We re-write the last equation as the form of the percentage change in the following

$$100 \times \frac{\mathrm{O}^{(x_i+1)} - \mathrm{O}^{(x_i)}}{\mathrm{O}^{(x_i)}} = 100(e^{\alpha_i} - 1).$$

Therefore, we have following interpretation of the α_i: under the same condition, increasing x_i by one unit leads to the change of odds of observing (Y=1) by $100(e^{\alpha_i} - 1)\%$.

To illustrate the relationship between a predictor variable and the response variable with a 2-way interaction, we can use a graphical approach such as interaction plots or 3D surface plots. The following example uses 3D surface plot to view the relationship in the logistic regression model with a 2nd order interaction in the following example based on simulated data.

```
# Load necessary library
library(ggplot2)
# Simulate a working data set
set.seed(123)  # For reproducibility
n = 100
x1 = rnorm(n, 0.4,2)
x2 = rexp(n, 1.2)
l0 =.1 + .2*x1 - 1.5*x2  + 0.5*x1*x2
p = exp(l0)/(1+exp(l0))
z = ifelse(p>0.48, 1, 0)
# Combine into a data frame
data = data.frame(x1 = x1, x2 = x2, y = z)
# Fit a quadratic regression model with interaction term
logit0 = glm(y ~ x1 + x2, family = binomial(link=logit), data = data)
# Summary of the model
#summary(logit0)
coef0 = coef(logit0)
ln0 = coef0[1] + coef0[2]*x1 + coef0[3]*x2
data$z0 = exp(ln0)/(1+exp(ln0))
###
logit1 = lm(z~x1+x2+x1*x2,family = binomial(link=logit), data = data)
```

```r
#summary(logit1)
coef1 = coef(logit1)
ln1 = coef1[1] + coef1[2]*x1 + coef1[3]*x2 + coef1[4]*data$x1*x2
data$z1 = exp(ln1)/(1+exp(ln1))
```

```r
# Visualizing the quadratic regression surface
#library(ggplot2)
#library(plot_ly)
quad = plot_ly(data,
               x= ~x1,
               y= ~x2,
               z= ~z1,
               type='mesh3d',
               intensity = ~z1,
               text = "Interaction",
               colors= colorRamp(topo.colors(5, alpha=0.7, rev = TRUE))
           )
quad.ln = add_trace(
           p = quad,
           z = ~z0,
           x = ~x1,
           y = ~x2,
           text = "No Interaction",
           type = "mesh3d",
           intensity = ~z0,
           colors= colorRamp(rainbow(5))) %>%
           hide_colorbar() %>%
       layout(title = 'Logistic Surface: No Interaction vs Interaction',
              margin = list(l = 5, r = 5, b = 50, t = 50, pad = 4))
quad.ln
```

Logistic Surface: No Interaction vs Interaction

2.5 Paremeter Estimation

The regression coefficients in both (normal) linear and logistic regression models are estimated through maximizing the likelihood function of the parameters - Maximum Likelihood Estimation (MLE).

2.5.1 Concepts of Likeliood *(Optional)*

Fisher's Maximum Likelihood Estimation (MLE) is a method for estimating the parameters of a statistical model that maximizes the likelihood function. The likelihood function represents the probability of the observed data given the parameters of the model. Fisher's MLE is widely used because it has desirable properties under certain conditions, such as consistency, efficiency, and asymptotic normality.

In general, let $\{x_1, x_2, \cdots, x_n\} \overset{\text{i.i.d}}{\sim} f(x; \alpha, \beta)$, then **the likelihood of observing the random sample** is defined to be

$$\mathbb{L}(\alpha, \beta) = \prod_{i=1}^{n} f(x_i; \alpha, \beta).$$

Fisher's proposal of estimating parameters (α, β) is to find the values for α and β, denoted by $(\hat{\alpha}, \hat{\beta})$, that maximize the likelihood $\mathbb{L}(\alpha, \beta)$. We can use the following notation to denote the above optimization

$$(\hat{\alpha}, \hat{\beta}) = \arg\max_{(\alpha,\beta)} \mathbb{L}(\alpha, \beta)$$

For computational convenience, Instead of maximizing the likelihood function $\mathbb{L}(\alpha, \beta)$, we maximize the log-likelihood $\updownarrow(\alpha, \beta) = \log \mathbb{L}(\alpha, \beta)$. This converts the product of probabilities into a sum, simplifying calculations and reducing numerical instability. That is,

$$(\hat{\alpha}, \hat{\beta}) = \arg\max_{(\alpha,\beta)} \lessdot(\alpha, \beta)$$

where

$$\lessdot(\alpha, \beta) = \sum_{i=1}^{n} \log f(x_i; \alpha, \beta).$$

2.5.2 Normal Linear Regression Likelihood *(Optional)*

Without loss of generality, we consider the following normal linear regression model

$$Y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_k x_k + \epsilon, \quad \text{where} \quad \epsilon \sim N(0, \sigma)$$

which implies that

$$\mu = E[Y] = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_k x_k \quad \text{and} \quad \text{var}(Y) = \sigma^2.$$

Or equivalent to

$$Y \sim N(\mu, \sigma).$$

The predictor feature variables $x_1, x_2, \cdots, x_k$ are assumed to be non-random and the response variable y is a random variable with the above distribution. The explicit density of Y is given by

$$f(y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{[y-(\alpha_0+\alpha_1 x_1+\alpha_2 x_2+\cdots+\alpha_k x_k)]^2}{2\sigma^2}}.$$

Let $\{(y_1, x_{1i}, x_{2i}, \cdots, x_{ki})\}_{i=1}^n \overset{\text{i.i.d}}{\sim} f(y)$, the likelihood of observing the `i.i.d` sample based on the above distributional assumption is given by

$$\mathbb{L}(\alpha_0, \alpha_1, \cdots, \alpha_k) = (\sqrt{2\pi}\sigma)^{-n} e^{-\sum_{i=1}^n \frac{[y_i-(\alpha_0+\alpha_1 x_{1i}+\alpha_2 x_{2i}+\cdots+\alpha_k x_{ki})]^2}{2\sigma^2}}$$

The log-likelihood function is

$$\lessdot(\alpha_0, \alpha_1, \cdots, \alpha_k) = -n\log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2}\sum_{i=1}^n [y_i - (\alpha_0 + \alpha_1 x_{1i} + \alpha_2 x_{2i} + \cdots + \alpha_k x_{ki})]^2$$

The MLE is the solution to the above optimization problem.

2.5.3 Logistic Regression Likelihood *(Optional)*

Note that the response variable in the binary logistic regression model is assumed to a Bernoulli random variable with `success` probability p that is defined as

$$p = \frac{\exp(\alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_k x_k)}{1 + \exp(\alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_k x_k)}$$

with probability mass function

$$P(Y = y) = p^y(1-p)^{1-y}, \quad \text{where} \quad y = 0 \ \text{ or } \ 1.$$

Let $\{(y_1, x_{1i}, x_{2i}, \cdots, x_{ki})\}_{i=1}^n \overset{\text{i.i.d}}{\sim} f(y)$, the likelihood of observing the `i.i.d` sample based on the above distributional assumption is given by

$$\mathbb{L}(\alpha_0, \alpha_1, \cdots, \alpha_k) = \prod_{i=1}^n p^{y_i}(1-p)^{1-y_i}.$$

The corresponding log-likelihood function is given by

$$\lessdot(\alpha_0, \alpha_1, \cdots, \alpha_k) = \sum_{i=1}^n [y_i \log p + (1-y_i)\log(1-p)]$$

$$= \sum_{i=1}^n y_i \left[\frac{\exp(\alpha_0 + \alpha_1 x_{1i} + \alpha_2 x_{2i} + \cdots + \alpha_k x_{ki})}{1 + \exp(\alpha_0 + \alpha_1 x_{1i} + \alpha_2 x_{2i} + \cdots + \alpha_k x_{ki})}\right] \left[1 - \frac{\exp(\alpha_0 + \alpha_1 x_{1i} + \alpha_2 x_{2i} + \cdots + \alpha_k x_{ki})}{1 + \exp(\alpha_0 + \alpha_1 x_{1i} + \alpha_2 x_{2i} + \cdots + \alpha_k x_{ki})}\right].$$

The MLE of the regression coefficients maximizes the above log-likelihood. To be more specific,

$$(\hat{\alpha}_0, \hat{\alpha}_1, \cdots, \hat{\alpha}_k) = \arg \max_{(\alpha_0, \alpha_1, \cdots, \alpha_k)} \mathbb{L}(\alpha_0, \alpha_1, \cdots, \alpha_k).$$

2.5.4 Finding MLE Using Software Programs

In R, there are several libraries for performing linear and logistic regression, each with unique strengths for different use cases.

1. Base R (stats package)

- Linear Regression

```r
# Load the data
data(mtcars)

# Fit a linear regression model
model_lm <- lm(mpg ~ wt + hp, data = mtcars)

# Summary of the model
summary(model_lm)
```

```

Call:
lm(formula = mpg ~ wt + hp, data = mtcars)

Residuals:
   Min     1Q Median     3Q    Max 
-3.941 -1.600 -0.182  1.050  5.854 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 37.22727    1.59879  23.285  < 2e-16 ***
wt          -3.87783    0.63273  -6.129 1.12e-06 ***
hp          -0.03177    0.00903  -3.519  0.00145 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.593 on 29 degrees of freedom
Multiple R-squared:  0.8268,	Adjusted R-squared:  0.8148 
F-statistic: 69.21 on 2 and 29 DF,  p-value: 9.109e-12
```

```r
# Predict new values
predict(model_lm, newdata = data.frame(wt = 3, hp = 100))
```

```
       1 
22.41648 
```

- Logistic Regression

```r
# Load the data
data(mtcars)
mtcars$am <- as.factor(mtcars$am) # Convert to factor for logistic regression

# Fit a logistic regression model
model_glm <- glm(am ~ wt + hp, family = binomial, data = mtcars)

# Summary of the model
summary(model_glm)
```

```

Call:
```

```
glm(formula = am ~ wt + hp, family = binomial, data = mtcars)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 18.86630    7.44356   2.535  0.01126 *
wt          -8.08348    3.06868  -2.634  0.00843 **
hp           0.03626    0.01773   2.044  0.04091 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 43.230  on 31  degrees of freedom
Residual deviance: 10.059  on 29  degrees of freedom
AIC: 16.059

Number of Fisher Scoring iterations: 8
```

```
# Predict probabilities
predict(model_glm, newdata = data.frame(wt = 3, hp = 100), type = "response")
```

```
        1
0.1469699
```

2. MASS Package: Robust Linear Regression

```
#library(MASS)

# Robust linear regression
model_rlm <- rlm(mpg ~ wt + hp, data = mtcars)

# Summary
summary(model_rlm)
```

```

Call: rlm(formula = mpg ~ wt + hp, data = mtcars)
Residuals:
    Min      1Q  Median      3Q     Max
-3.6639 -1.3057  0.1727  1.3162  6.3392

Coefficients:
            Value   Std. Error t value
(Intercept) 36.5840  1.4380    25.4407
wt          -3.8801  0.5691    -6.8180
hp          -0.0293  0.0081    -3.6050

Residual standard error: 2.006 on 29 degrees of freedom
```

3. glmnet Package: Penalized Regression (Lasso and Ridge)

```
#library(glmnet)

# Prepare data
X <- as.matrix(mtcars[, c("wt", "hp")])
y <- mtcars$mpg

# Fit a ridge regression model (alpha = 0)
```

```r
model_ridge <- glmnet(X, y, alpha = 0)

# Fit a lasso regression model (alpha = 1)
model_lasso <- glmnet(X, y, alpha = 1)

# Cross-validation for lambda
cv_model <- cv.glmnet(X, y, alpha = 1)

# Best lambda
cv_model$lambda.min
```

```
[1] 0.2176783
```

```r
# Predict using the lasso model
predict(model_lasso, newx = as.matrix(data.frame(wt = 3, hp = 100)), s = cv_model$lambda.min)
```

```
           s1
[1,] 22.29602
```

4. caret Package: Unified Interface for Training Models

```r
#library(caret)

# Linear regression
model_caret_lm <- train(mpg ~ wt + hp, data = mtcars, method = "lm")
summary(model_caret_lm)
```

```

Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
   Min     1Q Median     3Q    Max 
-3.941 -1.600 -0.182  1.050  5.854 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 37.22727    1.59879  23.285  < 2e-16 ***
wt          -3.87783    0.63273  -6.129 1.12e-06 ***
hp          -0.03177    0.00903  -3.519  0.00145 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.593 on 29 degrees of freedom
Multiple R-squared:  0.8268,	Adjusted R-squared:  0.8148 
F-statistic: 69.21 on 2 and 29 DF,  p-value: 9.109e-12
```

```r
# Logistic regression
model_caret_glm <- train(am ~ wt + hp, data = mtcars, method = "glm", family = binomial)
summary(model_caret_glm)
```

```

Call:
NULL

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
```

```
(Intercept) 18.86630    7.44356   2.535  0.01126 *
wt          -8.08348    3.06868  -2.634  0.00843 **
hp           0.03626    0.01773   2.044  0.04091 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 43.230  on 31  degrees of freedom
Residual deviance: 10.059  on 29  degrees of freedom
AIC: 16.059

Number of Fisher Scoring iterations: 8
```

3 Performance Measures

Performance measures are metrics used to assess the quality of statistics machine learning model. They quantify how well a model makes predictions, capturing its strengths and limitations. The choice of performance measures depends on the problem being addressed—classification, regression, clustering, etc.

The quality of a statistical or machine learning model/algorithm can be assessed based on the following criteria

Regression Metrics

- **Mean Absolute Error (MAE)**: Represents the average absolute difference between predicted and actual values.
- **Mean Squared Error (MSE) and Root Mean Squared Error (RMSE)**: MSE penalizes larger errors more heavily, while RMSE offers interpretability in the same units as the target variable.
- **R-squared**: Indicates the proportion of variance in the target variable explained by the model.

Classification Metrics

- **Accuracy**: The ratio of correctly predicted instances to the total number of instances. While widely used, accuracy can be misleading in imbalanced datasets.
- **Precision, Recall, and F1-Score**: Precision measures the proportion of correctly predicted positive instances, while recall captures the proportion of actual positives identified by the model. The F1-score provides a harmonic mean of precision and recall, balancing their trade-off.
- **ROC-AUC**: The Area Under the Receiver Operating Characteristic Curve evaluates a model's ability to distinguish between classes, providing a threshold-independent measure.

Clustering Metrics

- **Silhouette Score**: Measures how similar an instance is to its own cluster compared to other clusters.
- **Adjusted Rand Index (ARI)**: Evaluates clustering results against a ground truth.

Performance measures not only allow comparison between different models but also highlight areas for improvement. For instance, a high recall but low precision might indicate excessive false positives, suggesting the need to refine the model's decision boundaries.

4 Cross-Validation

Cross-validation is a **computational statistical technique** used to estimate the performance of a machine learning model on unseen data. Its essence lies in assessing how well a model generalizes to an independent data set, ensuring that the model isn't over-fitted to the training data.

By applying cross-validation, practitioners ensure that models are rigorously tested and refined, leading to better real-world performance and reliable decision-making. As data sets grow and models become more complex, cross-validation will continue to be a critical tool in the pursuit of predictive accuracy and reliability.

4.1 Common Types of CV

K-Fold Cross-Validation: In k-fold cross-validation, the data set is divided into k equally sized subsets or folds. The model is trained on k-1 folds and tested on the remaining fold. This process is repeated k times, with each fold serving as the test set once. The final performance metric is averaged over all k iterations, reducing bias and variance in the evaluation.

Stratified K-Fold Cross-Validation: For imbalanced data sets, stratified k-fold ensures that each fold maintains the same class distribution as the original data set, providing a more representative evaluation.

Leave-One-Out Cross-Validation (LOOCV): LOOCV is a special case of k-fold where k equals the number of instances in the dataset. While thorough, it is computationally expensive and sensitive to outliers.

4.2 Essence ang Logic of CV

Cross-validation is a cornerstone technique in machine learning and statistics, serving as a robust method for assessing the performance and generalizability of predictive models. Its essence lies in dividing a dataset into separate subsets to train and test a model in a systematic and reliable manner. The logic behind cross-validation is grounded in the fundamental principles of empirical validation and the need to mitigate overfitting while maximizing predictive accuracy. This subsection explores the essence and logic of cross-validation, emphasizing its theoretical underpinnings, practical applications, and variations.

The Essence of Cross-Validation

At its core, cross-validation seeks to estimate how well a predictive model will perform on unseen data. This is critical because the ultimate goal of most machine learning models is not merely to perform well on training data but to generalize to new, unseen datasets. Without an unbiased estimate of generalization performance, a model risks overfitting—capturing noise or specific patterns in the training data that do not generalize beyond it.

Cross-validation addresses this issue by partitioning the available data into complementary subsets: one used for training the model and the other for testing its performance. By systematically rotating through different partitions of the data, cross-validation ensures that every data point is used for both training and validation, reducing the likelihood of biased evaluations.

The Logic of Cross-Validation

The logic of cross-validation is rooted in sound statistical principles. It leverages the following key ideas:

Data Partitioning for Validation: By dividing the dataset, cross-validation creates a scenario where the model's predictions are tested on data not seen during training. This mimics real-world conditions, where models encounter entirely new data.

Variance Reduction: Single train-test splits can lead to variance in performance metrics due to the arbitrary nature of the split. Cross-validation, by aggregating results across multiple splits, provides a more stable and reliable estimate of model performance.

Minimizing Overfitting: By exposing the model to multiple train-test configurations, cross-validation reduces the risk of overfitting to a specific subset of data.

Efficiency in Resource-Constrained Environments: Especially with limited datasets, cross-validation makes optimal use of all available data for both training and validation, maximizing informational efficiency.

Practical Applications

Cross-validation is ubiquitous in machine learning and statistics. Its applications include:

Model Selection: Comparing the performance of multiple models or hyperparameter settings.

Bias-Variance Tradeoff Analysis: Understanding how a model's complexity affects its generalization performance.

Algorithm Validation: Providing robust performance metrics for research papers and industrial applications.

Feature Selection: Evaluating the importance of different features in predictive modeling.

4.3 Purpose of CV

It is a method of assessing model performance by dividing data into subsets, training the model on some of these subsets, and validating it on others. In an era where data-driven decisions are increasingly critical, cross-validation remains a cornerstone of sound analytical practice.

Ensuring Generalization

A primary purpose of cross-validation is to evaluate how well a model generalizes to new, unseen data. When a model is trained on a given dataset, there is a risk that it performs well only on that data due to overfitting, where the model learns noise and specific patterns in the training data rather than underlying trends. Cross-validation mitigates this issue by repeatedly training and testing the model on different subsets of the data. This process provides a more reliable estimate of the model's ability to make accurate predictions on independent datasets, ensuring it captures the true structure of the data.

Model Selection

In machine learning, there are often multiple candidate models or algorithms available for a particular task. Choosing the best model involves comparing their performance on a given dataset. Cross-validation provides a systematic way to conduct this comparison by generating performance metrics for each model across multiple training and validation splits. By averaging these metrics, practitioners can objectively identify the model that performs best on average, reducing the likelihood of selecting a model based on an overly optimistic or pessimistic single evaluation.

Hyperparameter Tuning

Many machine learning models have hyperparameters that control their behavior, such as the regularization strength in linear regression or the number of neighbors in a k-nearest neighbors classifier. These hyperparameters significantly impact model performance, and selecting their optimal values is a critical step in the modeling process. Cross-validation allows practitioners to test different hyperparameter settings and evaluate their effects on the model's performance. This iterative process, often paired with techniques like grid search or random search, helps ensure that the chosen hyperparameters lead to a model that performs well on unseen data.

Performance Estimation

Another key purpose of cross-validation is to provide a realistic estimate of a model's predictive performance. Unlike evaluating a model on a single holdout validation set, cross-validation uses multiple training-validation splits to account for variability in the data. The performance metrics obtained—such as accuracy, precision, recall, or mean squared error—are averaged across splits to produce a more robust estimate. This reduces the likelihood of overestimating or underestimating the model's effectiveness, particularly when working with limited data.

Avoiding Data Overlap Issues

In certain scenarios, such as time-series forecasting or grouped data, data points in the training and validation sets may overlap or be correlated. Cross-validation frameworks, such as time-series cross-validation or grouped k-fold cross-validation, are designed to address these situations. By structuring the splits appropriately, these techniques ensure that validation sets are independent of the training data, preventing data leakage and producing more trustworthy performance evaluations.

Ensuring Reproducibility

Cross-validation also plays a critical role in ensuring reproducibility and rigor in applied machine learning. By adopting standardized cross-validation protocols, practitioners can compare results more fairly across studies. Additionally, the use of cross-validation reduces the risk of drawing misleading conclusions based on anomalous splits or small sample sizes.

Cross-validation is an indispensable tool in machine learning and statistical modeling, serving multiple purposes that enhance model reliability and performance. From ensuring generalization and enabling model selection to facilitating hyperparameter tuning and providing realistic performance estimates, cross-validation addresses many challenges inherent in predictive modeling. By incorporating this technique into the modeling process, practitioners can build models that are not only accurate but also robust and trustworthy in real-world applications. In an era where data-driven decisions are increasingly critical, cross-validation remains a cornerstone of sound analytical practice.

Among various cross-validation methods, the CV is the most frequently used in practice for various purposes. We will discuss k-fold CV and LOOCV in the subsequent sections.

Performance measures and cross-validation work hand in hand to provide a comprehensive evaluation framework. While performance measures quantify model effectiveness, cross-validation ensures that these metrics are reliable and not the result of overfitting. For example, a model with high accuracy on the training data but low cross-validated accuracy indicates overfitting. Similarly, cross-validation results allow practitioners to choose the best model or hyperparameter configuration based on robust estimates of real-world performance.