DSA5205 Data Science for Quantitative Finance

AY2024/25 Sem1 By Zhao Peiduo

Introduction and Background

Introduction to Data Science and Quantitative Finance

What is Data Science?
Data science is an interdisciplinary field that uses scientific methods, algorithms, and systems to extract insights from structured and unstructured data. It combines knowledge from:

- Mathematics and Statistics
- Computer Science (especially algorithms and systems)
- Domain expertise

Big Data and "V"s
Big Data is characterized by several "V"s:
- **Volume**: Massive amounts of data
- **Velocity**: The speed of data processing
- **Variety**: Different types of data (structured/unstructured)
- **Veracity**: The quality or trustworthiness of the data
- **Value**: The potential benefit derived from analyzing the data

Machine Learning Overview
Machine learning involves algorithms that allow computers to learn from data. Important types of machine learning include:
- **Supervised Learning**: Learn a function from labeled data
- **Unsupervised Learning**: Find patterns without labeled data
- **Reinforcement Learning**: Learn from rewards and punishments

R Code for Loading Data:
```
data <- read.csv("data.csv")
head(data)
```
Quantitative Finance
Quantitative finance uses statistical and mathematical models to analyze financial markets and manage risks. Common models include:
- **CAPM**: Capital Asset Pricing Model
- **Black-Scholes**: Option pricing model

The Data Science Process
Data science typically follows these steps:
1. Problem definition
2. Data collection
3. Data cleaning and preprocessing
4. Model building (Machine learning, statistics)
5. Evaluation and interpretation
6. Reporting and visualization

Challenges in Data Science
Some challenges include:
- **Data Cleaning**: Handling missing values, outliers
- **Model Selection**: Choosing the right model
- **Ambiguity**: Dealing with uncertainty in data

R Code for Model Training:
```
model <- lm(Sepal.Length ~Sepal.Width + Petal.Length, data = iris)
summary(model)
```
Quantitative Finance Example

- **Example:** Predict whether a stock will drop by 10% in the next Y days.
- Requires: Feature engineering, domain knowledge, and model validation.

Distribution and Risks

Moments
Let X be a random variable. The k^{th} moment of X is defined as: $E(X^k)$
The first moment is the expectation. The k^{th} central moment is: $\mu_k = E[(X - E(X))^k]$
Variance: $\mu_2 = \mathrm{Var}(X)$
Skewness: $S_k(X) = \frac{\mu_3}{(\mu_2)^{3/2}}$
Kurtosis: $Kur(X) = \frac{\mu_4}{(\mu_2)^2}$

Skewness

Skewness is a measure of symmetry. For a continuous random variable Y: $Sk(Y) = E\left[\frac{(Y-E(Y))^3}{\sigma^3}\right]$

For a discrete random variable: $Sk(Y) = \sum \frac{(y-E(Y))^3}{\sigma^3} f(y)$

Kurtosis

Kurtosis measures tail thickness. The kurtosis of a random variable Y is: $Kur(Y) = E\left[\frac{(Y-\mu_Y)^4}{\sigma_Y^4}\right]$

For a normal distribution $Y \sim N(\mu, \sigma^2)$, the kurtosis is 3. The excess kurtosis is: $Kur(Y) - 3$

Heavy-Tailed Distributions
A distribution is heavy-tailed if its tails are thicker than the normal distribution. A right Pareto tail has the form: $f(y) \sim Ay^{-(a+1)} \quad \text{as} \quad y \to \infty$
The parameter a is called the tail index.

Student's t-Distributions
The probability density function (pdf) of the t-distribution with ν degrees of freedom is: $f(x) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\,\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}}$
Mean: $E[X] = 0$ for $\nu > 1$
Variance: $Var(X) = \frac{\nu}{\nu-2}$ for $\nu > 2$

Quantile-Based Measures
Quantiles of a distribution: Let $X \sim F(x)$. The p^{th} quantile of $F(x)$ is defined as: $q_p = F^{-1}(p)$
Interquartile range (IQR): $IQR = q_{0.75} - q_{0.25}$

Risk Measures
Value at Risk (VaR) is defined as: $VaR_\alpha = q_\alpha(F_L) \quad \text{such that} \quad P(L \geq VaR_\alpha) = \alpha$
Expected Shortfall (ES) is: $ES_\alpha = E(L|L \geq VaR_\alpha)$

Jarque-Bera Test
The Jarque-Bera test is a test of normality based on skewness and kurtosis. For a sample $Y_1, \ldots, Y_n$:

$$Sk = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \bar{Y}}{s}\right)^3$$

$$Kur = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \bar{Y}}{s}\right)^4$$

$$JB = n\left(\frac{Sk^2}{6} + \frac{(Kur-3)^2}{24}\right)$$

Under the null hypothesis of normality, the test statistic JB follows a chi-squared distribution with 2 degrees of freedom.
In R, the function `jarque.bera.test()` returns the test statistic and its p-value.

Heavy-Tailed and Skewed Distributions
A generalized error distribution (GED) with shape parameter ν is used for modeling heavy tails and skewness:

$$f_{\text{GED}}(y|\nu) = k(\nu)\exp\left(-\frac{1}{2}\left|\frac{y}{\lambda_\nu}\right|^\nu\right), \quad -\infty < y < \infty$$

The tail behavior is controlled by ν, with smaller values indicating heavier tails.

Profile Likelihood
Profile likelihood is used for constructing confidence intervals by fixing one parameter at a time and maximizing over the others:

$$L_{\max}(\theta_1) = \max_{\theta_2} L(\theta_1, \theta_2)$$

Fitting Skewed-t Distribution in R
```
fit = sstdFit(male.earnings, hessian = TRUE)
para = fit$estimate
xgrid = seq(0, max(male.earnings) + 5, length.out = 100)
plot(density(male.earnings), main = "Male Earnings", ylim = c(0, 0.1), col = 4, lwd = 2)
lines(xgrid, dsstd(xgrid, mean = para[1], sd = para[2], nu = para[3], xi = para[4]), col = 4, lty = 5, lwd = 2)
```

Value at Risk and Expected Shortfall (VaR and ES)
Using t-distribution for estimating VaR and ES: `S = 5000`
```
alpha = 0.05
mu = mean(returnsCo)
sd = sd(returnsCo)
Finv = qnorm(alpha, mean = mu, sd = sd)
VaR = -S * Finv
ES = S * (-mu + sd * dnorm(qnorm(alpha)) / alpha)
```

Generalized Error Distributions (GED)
The Generalized Error Distribution (GED) has a flexible tail weight controlled by the shape parameter ν.
If $Y \sim GED(\mu, \sigma, \nu)$, its pdf is:

$$f_{\text{GED}}(y|\nu) = k(\nu)\exp\left(-\frac{1}{2}\left|\frac{y}{\lambda_\nu}\right|^\nu\right)$$

The distribution is symmetric about μ, with $\nu = 2$ corresponding to the normal distribution, and $\nu = 1$ being the double-exponential. The tail weight increases as ν decreases.

Quantile-Quantile (QQ) Plot
QQ plots are used to compare the quantiles of a dataset to those of a theoretical distribution, typically the normal distribution.
If the data follows the theoretical distribution, the points should lie approximately on the 45-degree line. Deviations indicate skewness, heavy tails, or other non-normal behavior.

Fisher Information
Fisher information is defined as the expected value of the negative second derivative of the log-likelihood function. For a parameter θ:

$$I(\theta) = -E\left[\frac{\partial^2}{\partial\theta^2}\log L(\theta)\right]$$

Fisher information quantifies the amount of information that an observable random variable Y carries about an unknown parameter θ. The standard error of an estimator is inversely proportional to the square root of the Fisher information.

Likelihood Ratio Tests (LRT)
The likelihood ratio test (LRT) compares the likelihood of the data under two models: a full model and a reduced model (where some parameters are constrained).
The test statistic is:

$$\text{LRT} = 2\left(\log L(\hat{\theta}_{\text{full}}) - \log L(\hat{\theta}_{\text{reduced}})\right)$$

The statistic follows a chi-squared distribution with degrees of freedom equal to the number of constraints.

Robust Estimation
Robust estimators are resistant to outliers and deviations from model assumptions.
An example is the trimmed mean, where a proportion of the largest and smallest values are excluded. Another is the Median Absolute Deviation (MAD), which is a robust measure of scale:

$$\hat{\sigma}_{MAD} = 1.4826 \times \text{median}\left(|Y_i - \text{median}(Y)|\right)$$

MAD is less sensitive to extreme values than standard deviation.

AIC and BIC
The Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) are used to compare models. Both criteria balance goodness of fit with model complexity:

$$AIC = -2\log L(\hat{\theta}_{\text{MLE}}) + 2p$$

$$BIC = -2\log L(\hat{\theta}_{\text{MLE}}) + p\log n$$

Where p is the number of parameters and n is the sample size. Lower values of AIC or BIC indicate better models, but BIC penalizes complexity more than AIC.

Variance-Covariance Method
In the variance-covariance method, the loss distribution is assumed to be multivariate normal. The linearized loss is approximated by:

$$L \sim N(\mu, \sigma^2)$$

Value at Risk (VaR) and Expected Shortfall (ES) are estimated from the distribution. For normal distribution:

$$V\hat{a}R_\alpha = \mu + \sigma\Phi^{-1}(1-\alpha)$$

$$\hat{ES}_\alpha = \mu + \sigma\frac{\phi(\Phi^{-1}(1-\alpha))}{\alpha}$$

where ϕ is the standard normal density and Φ^{-1} is the inverse cumulative distribution function (quantile) of the normal distribution.

Hill Estimator
The Hill estimator is used to estimate the tail index of heavy-tailed distributions. For order statistics $X_{(1)}, X_{(2)}, \ldots, X_{(n)}$, the Hill estimator is defined as:

$$\hat{\gamma}_H = \frac{1}{k}\sum_{i=1}^{k}\log\left(\frac{X_{(i)}}{X_{(k+1)}}\right)$$

The parameter k determines how many of the extreme values (largest observations) are used. A Hill plot helps to determine the appropriate value of k. The Hill estimator is particularly useful for Pareto-like heavy-tailed distributions.

Nonparametric Methods and Resampling

Histogram
A histogram divides the sample space into bins and approximates the density at each bin's center:

$$p_H(X) = \frac{\text{Number of } x(k) \text{ in the same bin as } x}{\text{Width of bin}}$$

Hyperparameters: bin width and starting position of the first bin.
Kernel Density Estimation (KDE)
KDE estimates the probability density of a random variable:

$$\hat{p}_{KDE}(x) = \frac{1}{nh^D}\sum_{i=1}^{n}K\left(\frac{x-x_i}{h}\right)$$

Here, $K(u)$ is the kernel function, and h is the bandwidth (smoothing parameter).

Parzen Windows
For Parzen window estimation, the kernel function is:

$$K(u) = \begin{cases}1 & \text{if } |u| \le \frac{1}{2},\\ 0 & \text{otherwise.}\end{cases}$$

Smooth Kernels
The Parzen window has several drawbacks:

- It yields density estimates that have discontinuities
- It weights equally all points x_i, regardless of their distance to the estimation point x

For these reasons, the Parzen window is commonly replaced with a smooth kernel function $K(u)$
Unimodal pdf, such as the Gaussian

$$K(x) = (2\pi)^{-D/2}e^{-1/2x'x}$$

Which leads to the density estimate:

$$p_{KDE}(x) = \frac{1}{Nh^D}\sum_{n=1}^{N}K\left(\frac{x-x^{(k)}}{h}\right)$$

Usually, but not always, $K(u)$ will be radially symmetric and

$$\int_{\mathbb{R}^D} K(x)dx = 1$$

Bandwidth Selection
The optimal bandwidth h minimizes the mean squared error (MSE) between the KDE and the true density:

$$MSE = \mathbb{E}[(\hat{p}_{KDE}(x) - p(x))^2] = \text{Bias}^2 + \text{Variance}.$$

The optimal bandwidth for a Gaussian kernel is:

$$h^* = 1.06 \cdot \sigma \cdot n^{-1/5}.$$

Maximum Likelihood Cross-validation

- The ML estimate of h is degenerate since it yields $h_{ML} = 0$, a density estimate with Dirac delta functions at each training data point
- A practical alternative is to maximize the "pseudo-likelihood"computed using leave-one-out cross-validation

$$h^* = \arg\max\left\{\frac{1}{N}\sum_{n=1}^{N}\log p_{-n}\left(x^{(n)}\right)\right\}$$

Where

$$p_{-n}\left(x^{(n)}\right) = \frac{1}{(N-1)h}\sum_{m=1,m\neq n}^{N}K\left(\frac{x^{(n)}-x^{(m)}}{h}\right)$$

Multivariate Density Estimation

$$p_{KDE}(x) = \frac{1}{Nh^D}\sum_{n=1}^{N}K\left(\frac{x-x^{(n)}}{h}\right)$$

Product Kernels

$$p_{PKDE}(x) = \frac{1}{N}\sum_{i=1}^{N}K(x, x^{(n)}, h_1, \ldots, h_D)$$

where

$$K(x, x^{(n)}, h_1, \ldots, h_D) = \frac{1}{h_1 \ldots h_D}\prod_{d=1}^{D}K_d\left(\frac{x_d - x_d^{(n)}}{h_d}\right)$$

Transformation kernel density estimator (TKDE):

- transform the data so that the density of the transformed data is easier to estimate by the KDE
- change-of-variables formula

Transformation KDE

Algorithm

1. Start with data $Y_1, \ldots, Y_n$.
2. Transform the data to $X_1 = g(Y_1), \ldots, X_n = g(Y_n)$; g is monotonic.
3. Let f_X be the usual KDE using $X_1, \ldots, X_n$.
4. $f_Y(y) = f_X\{g(y)\}\left|g'(y)\right|$. Plugging in $y = g^{-1}(x)$, then

$$f_Y(g^{-1}(x)) = f_X\{x\}\left|g'(g^{-1}(x))\right|$$

K-fold Cross-validation in Detail

Let the K parts be $C_1, C_2, \ldots, C_K$, where C_k denotes the indices of the observations in part k. There are n_k observations in part k: if n is a multiple of K, then $n_k = n/K$.

Compute $CV_{(K)} = \sum_{k=1}^{K} \frac{n_k}{n} MSE_k$

where $MSE_k = \sum_{i \in C_k} (y_i - \hat{y}_i)^2 / n_k$ and $\hat{y}_i$ is the fit for observation i, obtained from the data with part k removed.

Setting $K = n$ yields n-fold or *leave-one-out cross-validation* (LOOCV).

With least-squares linear or polynomial regression, an amazing shortcut makes the cost of LOOCV the same as that of a single model fit! The following formula holds:

$$CV_{(K)} = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i - \hat{y}_i}{1 - h_i} \right)^2$$

where $\hat{y}_i$ is the ith fitted value from the original least squares fit, and h_i is the leverage (diagonal of the "hat"matrix; see ESL book for details). This is like the ordinary MSE, except the ith residual is divided by $1 - h_i$.

Cross-Validation
K-fold cross-validation divides the data into K parts. The model is trained on $K - 1$ parts and tested on the k-th part:

$$CV(K) = \frac{1}{n} \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i \in C_k} (\hat{y}_i - y_i)^2.$$

Bootstrap
Bootstrap estimates the uncertainty of an estimator by resampling:

$$SE_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1} \sum_{r=1}^{B} (\hat{\alpha}_r^* - \bar{\alpha}^*)^2}.$$

R Code for Transformation Kernel Density Estimation (TKDE):

```
X = qnorm(pstd(Y, mean=fit_std$par[1], sd=fit_std$par[2], nu=fit_std$par[3])) # transformed data
d2 = density(X) # KDE of transformed data
ginvx = qstd(pnorm(d2$x), mean = fit_std$par[1], sd = fit_std$par[2], nu = fit_std$par[3]) # inverse transformation
gprime_num = dstd(ginvx, mean = fit_std$par[1], sd = fit_std$par[2], nu = fit_std$par[3]) # numerator of derivative
gprime_den = dnorm(qnorm(pstd(ginvx, mean = fit_std$par[1], sd = fit_std$par[2], nu = fit_std$par[3]))) # denominator of derivative
gprime = gprime_num / gprime_den # derivative of transformation
plot(ginvx, d2$y * gprime, type="l", lwd=2, col=4) # plot KDE of untransformed data
```

R Code for KDE, TKDE, and Parametric Estimation Comparison:

```
d1 = density(Y) # KDE of untransformed data
plot(d1$x, d1$y, type="l", xlab="y", ylab="Density(y)", xlim=c(.035,.06), ylim=c(0,.8), lwd=2, col=2) # plot KDE of untransfor-
med data
x = seq(-.09, .09, by=0.001)
lines(x, dstd(x, fit_std$par[1], fit_std$par[2], fit_std$par[3]), lty=3, lwd=2, col=3) # parametric fit
lines(ginvx, d2$y * gprime, type="l", lty=2, lwd=2, col=4) # TKDE
legend("topright", c("KDE","TKDE","PARAM"), lty=c(1,2,3), col=c(2,3,4), lwd=2)
```

R Code for Cross-Validation (Auto dataset):

```
library(ISLR)
data(Auto)
attach(Auto)
set.seed(1)
train = sample(392, 196)
lm.fit = lm(mpg ~horsepower, data=Auto, subset=train)
mean((mpg[-train] - predict(lm.fit, Auto[-train,]))^2) # MSE of test set
lm.fit2 = lm(mpg ~poly(horsepower, 2), data=Auto, subset=train)
mean((mpg[-train] - predict(lm.fit2, Auto[-train,]))^2) # Quadratic model
lm.fit3 = lm(mpg ~poly(horsepower, 3), data=Auto, subset=train)
mean((mpg[-train] - predict(lm.fit3, Auto[-train,]))^2) # Cubic model
```

Multivariate Methods and Factor Models

Multivariate Data and Factor Models

Covariance Matrices
Covariance measures the direction but not the strength of a linear relationship between two random variables X and Y. The covariance matrix of a random vector $\mathbf{Y} = (Y_1, \ldots, Y_d)$ is defined as:

$$\text{COV}(\mathbf{Y}) = \mathbb{E}\left[(\mathbf{Y} - \mathbb{E}[\mathbf{Y}])(\mathbf{Y} - \mathbb{E}[\mathbf{Y}])^\top\right].$$

The diagonal elements are the variances of individual components.

Multivariate Normal Distribution
A random vector $\mathbf{Y} = (Y_1, \ldots, Y_d)$ follows a multivariate normal distribution if its probability density function (PDF) is:

$$\phi_d(\mathbf{y}; \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{y} - \boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{y} - \boldsymbol{\mu})\right).$$

Here, $\boldsymbol{\mu}$ is the mean vector and Σ is the covariance matrix.

Principal Component Analysis (PCA)
PCA reduces the dimensionality of multivariate data by finding new orthogonal axes (principal components) that capture the maximum variance. The principal components are eigenvectors of the covariance matrix, and the corresponding eigenvalues represent the variance explained by each component.

R Code for PCA:

```
library(stats)
data(iris)
pca_res <- prcomp(iris[, 1:4], scale. = TRUE)
summary(pca_res)
plot(pca_res)
```

Multivariate t-Distribution
The random vector $\mathbf{Y}$ follows a multivariate t-distribution if:

$$\mathbf{Y} = \boldsymbol{\mu} + \sqrt{\frac{\nu}{W}}\mathbf{Z},$$

where W follows a chi-squared distribution with ν degrees of freedom, and $\mathbf{Z}$ follows a multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix Σ.

R Code for t-Distribution:

```
library(MASS)
set.seed(123)
mu <- c(0, 0)
Sigma <- matrix(c(1, 0.5, 0.5, 1), 2)
t_data <- mvrnorm(n = 500, mu = mu, Sigma = Sigma)
```

Copulas
A copula is a multivariate distribution function where the marginal distributions are uniform. The copula function links the marginal distributions to form a joint distribution. For a random vector $\mathbf{Y} = (Y_1, \ldots, Y_d)$, the copula is defined as:

$$C_{\mathbf{Y}}(u_1, \ldots, u_d) = \mathbb{P}(F_{Y_1}(Y_1) \leq u_1, \ldots, F_{Y_d}(Y_d) \leq u_d),$$

where $F_{Y_i}(Y_i)$ are the marginal cumulative distribution functions (CDFs) of Y_i.

R Code for Copulas:

```
library(copula)
norm_cop <- normalCopula(param = 0.5, dim = 2)
sample_cop <- rCopula(500, norm_cop)
plot(sample_cop)
```

Regression & Prediction

Linear Regression
Linear regression models the relationship between the outcome Y and predictors $X_1, X_2, \ldots, X_d$:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_d X_d + \epsilon$$

The coefficients β are estimated by minimizing the sum of squared residuals (errors):

$$J(\beta) = \frac{1}{2n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

Gradient Descent
To optimize the cost function $J(\beta)$, gradient descent iteratively updates the parameters β as follows:

$$\beta_j \leftarrow \beta_j - \alpha \frac{\partial J}{\partial \beta_j}$$

Here, α is the learning rate. Gradient updates:

$$\frac{\partial J}{\partial \beta_j} = \frac{1}{n} \sum_{i=1}^{n} (h_\beta(X^{(i)}) - Y^{(i)}) X_j^{(i)}$$

R Code for Linear Regression
```
model <- lm(Y ~X1 + X2 + ..., data = mydata)
summary(model)
```
Polynomial Regression
Extends linear regression by allowing higher-order terms:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \ldots + \beta_p X^p + \epsilon$$

This increases flexibility but can lead to overfitting.
R Code for Polynomial Regression
```
poly_model <- lm(Y ~poly(X, degree), data = mydata)
summary(poly_model)
```
Generalized Additive Models (GAM)
GAMs are an extension of linear models where each predictor X_j is modeled with a smooth function:

$$Y = \beta_0 + f_1(X_1) + f_2(X_2) + \ldots + f_p(X_p) + \epsilon$$

GAMs can capture nonlinear relationships without specifying the exact form.
R Code for GAM
```
library(gam)
gam_model <- gam(Y ~s(X1) + s(X2), data = mydata)
summary(gam_model)
```

Model Selection Techniques

Model Selection Overview

Subset Selection
Subset selection identifies a subset of predictors that best relate to the response. The best subset is chosen based on criteria like:

- Residual Sum of Squares (RSS)
- R^2
- AIC, BIC, Cp, or cross-validation error

Best Subset Selection:
1. Start with the null model M_0 which has no predictors.
2. For each k, fit models with exactly k predictors and select the one with the smallest RSS or highest R^2.
3. Use cross-validation or another criterion to choose the best model among all.

Stepwise Selection
Stepwise selection simplifies the search process:

- **Forward Stepwise:** Start with no predictors and iteratively add the one that improves the model the most.
- **Backward Stepwise:** Start with all predictors and iteratively remove the least useful.

Regularization and Shrinkage Methods
Shrinkage methods penalize the model's complexity, reducing variance:

- **Ridge Regression:** Shrinks coefficients by adding a penalty proportional to their squared values:

$$\hat{\beta} = \arg\min \left[RSS + \lambda \sum_{j=1}^{p} \beta_j^2 \right]$$

- **Lasso:** Uses an L_1 penalty to encourage sparsity, setting some coefficients to zero:

$$\hat{\beta} = \arg\min \left[RSS + \lambda \sum_{j=1}^{p} |\beta_j| \right]$$

R Code for Ridge and Lasso Regression
```
library(glmnet)
x <- model.matrix(Salary ~., Hitters)[,-1]
y <- Hitters$Salary

# Ridge regression
ridge.mod <- glmnet(x, y, alpha=0)
# Lasso regression
lasso.mod <- glmnet(x, y, alpha=1)
```
Choosing the Optimal Model
Model selection is based on minimizing test error, often estimated via cross-validation or information criteria like AIC and BIC:

$$AIC = -2\log(L) + 2 \cdot d \quad \text{and} \quad BIC = -2\log(L) + \log(n) \cdot d$$

Here, L is the likelihood of the model, and d is the number of parameters.
Cross-Validation

- Split the data into K folds.
- Train the model on $K-1$ folds and validate on the remaining fold.
- Repeat for each fold and average the validation errors.

This procedure provides a direct estimate of test error.