# How to Ace the Data Science Case Interview

## 1. Understanding the Situation and Problem Statement

Steps 1-4 are the steps to follow in a case interview. Step 5 gives more details on ML models.

Define the business problem clearly: Take notes, repeat the problem and ask questions.

## 2. Data Collection and Preparation

### 2.1. Data Sources
- Databases, APIs, files
- Social media networks

What data types are those?

### 2.2. Data Cleaning
- Imputation or removal
- Z-score: How many standard deviations a specific value is from the mean $\rightarrow > 2$ ($< -2$)
- IQR: Difference between Q3 and Q1 (50% of the data), upper limit: $Q3 + 1.5 \times IQR$

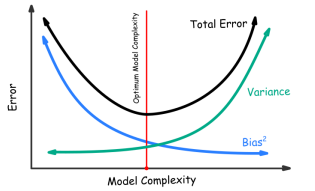
### 2.3. Exploratory Data Analysis (EDA)
- Visualization: Histograms, Scatterplots, correlation matrices
- Statistical analysis: Distribution, trends, relationships between variables

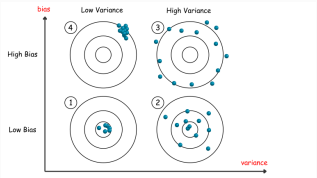
### 2.4. Data Preparation
- Min-Max or Z-standardization
- Relevant features
- Train, val, and test split

### 2.5. What else to consider
- Legal: GDPR, Anonymization
- Scalability: Fast datatypes (dask), PCA

## 3. Model Selection and Development

### 3.1. Model Selection
**Supervised Learning**: Algorithms that learn from labeled data (e.g., Regression, Classification).
**Unsupervised Learning**: Algorithms that learn from unlabeled data (e.g., Clustering, Dimensionality Reduction).
**Reinforcement Learning**: Algorithms that learn through reward and punishment.
**Bias-Variance Tradeoff**: Tradeoff between overfitting (high variance) and underfitting (high bias).
Bias: learn the wrong relationship
Variance: learns random relations that do not correlate with the data




**Cross-Validation:** Technique for avoiding overfitting by splitting the data into multiple folds.

### 3.2. Model Development
**Feature Engineering**: Selection and transf. of features to improve performance.
**Hyperparameter Tuning**: Optimization of parameters that are not learned from the data (e.g., Learning Rate).
**Ensemble Learning**: Combining multiple models to improve accuracy (e.g., Random Forest, Bagging, Boosting).
**Regularization**: Techniques to avoid overfitting (e.g., L1/L2 regularization).
**Loss Functions**: Functions to measure the error of a model (e.g., Mean Squared Error, Cross-Entropy).

### 3.3. Models

| | Linear Regression | Logistic Regression | Decision Tree | Random Forest |
|---|---|---|---|---|
| Type | Supervised | Supervised | Supervised | Supervised |
| Target Variable | Continuous | Binary/Categorical | Both | Both |
| Expl. | Linear relationship between features and target. | Estimates probabilities for categorical target variable. | Splits data based on feature values. | Ensemble of decision trees for robustness. |
| Use Case | Price prediction, Trend analysis | Classification, Spam detection | Classification, Regression | Classification, Regression |
| Prereqs. | Linear relationship | Linear boundaries | None | None |
| Complex. | Low | Low | Medium | Medium |
| Params | Learning rate, Regularization | Learning rate, Regularization | Depth, Split criteria | Number of trees, Depth |
| Metric | RMSE, MAE | Accuracy, F1 | Accuracy, RMSE | Accuracy, RMSE |
| Interpretability | High | High | Medium | Medium |

| | SVM | K-NN | K-Means | Neural Networks |
|---|---|---|---|---|
| Type | Supervised | Supervised | Unsupervised | Supervised |
| Target Variable | Binary/Categorical | Both | None | Both |
| Expl. | Finds the best separating line/plane between classes. | Classifies elements based on their neighbors. | Groups data into k similar clusters. | Simulates a neural network for complex patterns. |
| Use Case | Text classification, Face recognition | Recommendation systems, Classification | Customer segmentation, Anomaly detection | Image recognition, NLP |
| Prereqs. | Scaling, Labeling | Feature scaling | None | Large data set |
| Complex. | High | Low | Medium | High |
| Params | C, Kernel | Number of neighbors | Number of clusters | Learning rate, Neurons, Layers |
| Metric | Accuracy, F1 | Accuracy, RMSE | Silhouette Score, Inertia | Accuracy, F1 |
| Interpretability | Low | High | Medium | Low |

| | Naive Bayes | ARIMA |
|---|---|---|
| Type | Supervised | Supervised |
| Target Variable | Categorical | Time series |
| Expl. | Uses Bayesian theory for classification. | Models time series through trend, seasonality, and noise. |
| Use Case | Text classification, Spam filter | Financial markets, Weather prediction |
| Prereqs. | Independent features | Stationarity |
| Complex. | Low | Medium |
| Params | Laplace smoothing | p, d, q parameters |
| Metric | Accuracy, F1 | RMSE, MAE |
| Interpretability | High | Medium |

### 3.4. Evaluation Metrics

#### 3.4.1. Regression

**Mean Squared Error (MSE)** $\frac{1}{n}\sum(y_i - \hat{y}_i)^2$
**Sum of Squared Errors (SSE)** $\sum(y_i - \hat{y}_i)$
**Total Sum of Squares (SST)** $\sum(y_i - \overline{y_i})$
$R^2$ **Score**: $1 - \frac{SSE}{SST}$ ratio of explained variability in the data, not valid for nonlinear models, ¡0: worse than predicting just the mean

#### 3.4.2. Classification

**Accuracy**: Ratio of correct predictions to the total number of predictions. BUT: problem with imbalanced classes → good for classification problems with balanced classes
**Precision**: $\frac{TP}{TP+FP}$ →ideal if no FP (Spam detection: negative/no-spam mail is detected as positive/spam)
**Recall (TPR)**: $\frac{TP}{TP+FN}$ → ideal if no FN (Disease detection: positive/sick patient is detected as healthy/negative)
**F1-Score**: $2\frac{\text{precision}*\text{recall}}{\text{precision}+\text{recall}}$ → for imbalanced classes
**ROC-AUC**: Plot TPR vs FPR for different classification thresholds, aread under the curve = how likely the model differentiates positives vs negatives (1: perfect classification, 0.5: like random), not dependent on threshold, not always interpretable

| | | Actual | |
|---|---|---|---|
| | | Positive | Negative |
| **Prediction** | Positive | TP<br>Correct as positive | FP<br>Incorrect as positive |
| | Negative | FN<br>Incorrect as negative | TN<br>Correct as negative |

## 4. Implementation and Communication

### 4.1. Implementation

**Version Control (e.g., Git)**: For collaborative work and tracking changes.
**Continuous Integration / Continuous Deployment (CI/CD)**: Automatic testing and deployment of code.
**Model Serving**: Hosting models for real-time access, e.g., via REST APIs.
**Containerization (e.g., Docker)**: Packaging of software and dependencies for consistent execution.
**Scaling**: Adjusting resources to load, e.g., through horizontal scaling.
**Monitoring & Logging**: Monitoring system performance and capturing important information.

### 4.2. Communication

**Data Visualization**: Effective presentation of data using charts (e.g., Matplotlib, Seaborn).
**Storytelling with Data**: Connecting analyses with a narrative structure to convey insights.
**Interactive Dashboards (e.g., Tableau, Power BI, Dash)**: Creating interactive reports for end users.
**Technical Documentation**: Clear and precise documentation of code, architecture, and decisions.
**Non-Technical Communication**: Explaining technical concepts to a broader audience (e.g., stakeholders).

- ROI Analysis (Return on Investment): Evaluating how the results affect the business outcome.
- KPIs (Key Performance Indicators): Understanding and measuring the key metrics relevant to the business goal.
- Statistical Significance & Confidence Intervals: Assessing the reliability of the results.
- Cost-Benefit Analysis: Weighing the implementation costs against expected benefits.
- Risk Assessment: Assessing potential risks or downsides of implementing the results.
- Scenario Analysis: Conducting "what-if" analyses to understand different business scenarios.

**Ethics and Compliance**: Understanding and adhering to legal and ethical guidelines (e.g., GDPR).

### 4.3. Model Management

#### 4.3.1. Model Tracking (e.g., MLflow)

Tracking experiments, models, and metrics.

#### 4.3.2. A/B Testing

Comparing model versions to select the best.

**Procedure**

1. Randomly split the traffic (or data) into two groups: one served by model A and the other by model B.
2. Measure the performance metric of interest (e.g., conversion rate, click-through rate, etc.) for both groups.
3. Use statistical hypothesis testing to determine if the difference in performance is statistically significant.

## 5. Machine Learning Models

### 5.1. Linear Regression

Most basic and widely used techniques in predictive modeling
linear relationship between the dependent variable and one or more independent continuous variables

#### 5.1.1. Assumptions

- Linear Relationship: Assumes that the relationship between the dependent and independent variables is linear.
- Independence: Observations are independent of each other.
- Homoscedasticity: Constant variance of errors.
- Normality: The errors follow a normal distribution.

#### 5.1.2. Equation

$y = \beta_0 + \beta_1 x_1 + \epsilon$, where $y$ is the dependent variable, $x_1$ is the independent variable, $\beta_0$ is the y-intercept, $\beta_1$ is the slope, and $\epsilon$ is the error term.

#### 5.1.3. Training Parameters

Learning Rate: For gradient descent optimization.
Regularization: To prevent overfitting.

- Ridge Regularization (L2): $\|Y - X\beta\|^2 + \lambda\|\beta\|^2$ → coefficients to almost zero (keeps all features), less computationally complex, better if predictors are highly correlated
- Lasso Regularization (L1): $\|Y - X\beta\|^2 + \lambda\|\beta\|_1$ → some coefficients to zero (feature selection)
- Elastic net: combined

#### 5.1.4. Use-cases

- Price prediction
- Trend forecasting
- Risk assessment

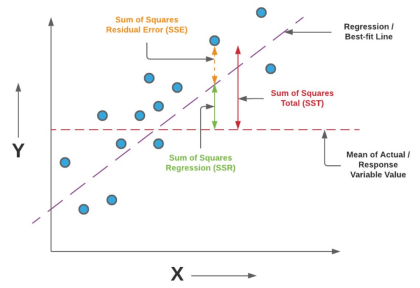
| Strengths | Weaknesses |
|---|---|
| Simple and interpretable | Only captures linear relationships |
| Fast to model and predict | Sensitive to outliers |
| Works well on small datasets | Cannot handle multicollinearity (independent variables are highly correlated) well |

#### 5.1.5. Data Preprocessing

- Feature scaling: Often required.
- Missing values: Need to be handled.
- Categorical variables: One-Hot Encoding.

#### 5.1.6. Interpretability

High; coefficients represent the change in the dependent variable for a one-unit change in an independent variable.



## 5.2. Logistic Regression

for classification problems, especially for binary outcomes

Transforms its output to lie in the range of 0 to 1 using the logistic function.

### 5.2.1. Assumptions

- Linear Relationship: Assumes a linear relationship between the logit of the outcome and the predictor variables.
- Independence: Observations should be independent of each other.
- Large Sample Size: Needs a large sample size for maximum likelihood estimates to be truly asymptotic.

### 5.2.2. Equation

$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \epsilon$, where $p$ is the probability of the dependent event occurring, $x_1$ is the independent variable, $\beta_0$ is the intercept, $\beta_1$ is the slope, and $\epsilon$ is the error term.

### 5.2.3. Training Parameters

- Learning Rate: For optimization using methods like stochastic gradient descent.
- Regularization: To avoid overfitting, commonly used methods include L1 and L2 regularization.

### 5.2.4. Use-cases

- Customer churn prediction
- Image classification
- Medical diagnosis

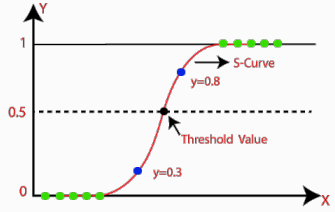
| Strengths | Weaknesses |
|---|---|
| High interpretability | Not suitable for non-linear problems |
| Efficient to train | Sensitive to feature scale |
| Can provide calibrated probabilities | Prone to underfitting |

### 5.2.5. Data Preprocessing

- Feature Scaling: Required for regularization and gradient descent.
- Missing Values: Must be addressed.
- Categorical Variables: Label encoding or one-hot encoding often used.

### 5.2.6. Interpretability

Coefficients signify the odds ratio for the given predictor variable. A coefficient greater than 1 represents increased odds of the outcome, while a coefficient less than 1 represents decreased odds.




## 5.3. Decision Tree

For both classification and regression tasks.

Decision trees work by splitting the dataset into subsets based on the most significant attributes, making decisions at every level.

### 5.3.1. Assumptions

- None: One of the few algorithms that does not assume any underlying data distribution.

### 5.3.2. Equation

No single equation represents a decision tree. They are built using algorithms like ID3, C4.5, or CART, which use measures like Information Gain, Gain Ratio, or Gini Index to decide splits.

### 5.3.3. Training Parameters

- Max Depth: Maximum depth of tree.
- Min Samples Split: Minimum number of samples required to split an internal node.
- Min Samples Leaf: Minimum number of samples required to be at a leaf node.
- Criterion: Metric used for splitting (Gini, Entropy)
  - **Gini Impurity** ($Gini = 1 - \sum_{i=1}^{n} p(i)^2$):
    - Formula: Gini impurity measures the disorder of a set. The lower the Gini impurity, the better.
    - Benefits: Faster to compute and works well in practice.
  - **Cross-Entropy** ($H = -\sum_{i=1}^{n} p(i)\log(p(i))$):
    - Formula: Cross-Entropy measures the dissimilarity between the distribution of observations and the distribution of predicted probabilities.
    - Benefits: Tends to distribute error more evenly compared to Gini.

### 5.3.4. Use-cases

- Customer segmentation
- Fraud detection
- Medical diagnosis

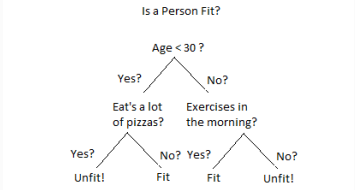
| Strengths | Weaknesses |
|---|---|
| High interpretability | Sensitive to noisy data |
| Can handle both categorical and numerical data | Prone to overfitting |
| No assumptions about data | Not suitable for unstructured data like images, text |

### 5.3.5. Data Preprocessing

- Feature Scaling: Not required.
- Missing Values: Can be handled but better to address.
- Categorical Variables: Label encoding sufficient, though one-hot can be used.

### 5.3.6. Interpretability

High; each node represents a decision based on one attribute, and the path from root to leaf gives the reasoning for the final decision.




## 5.4. Random Forests

An ensemble learning method that combines multiple decision trees to create a more robust and accurate model

For both classification and regression tasks.

### 5.4.1. Assumptions

- None: Like Decision Trees

### 5.4.2. Architecture

collections of decision trees

- **Bagging (Bootstrap Aggregating)**: Each tree is trained on a different subset of data, subsets are created by sampling with replacement → diversity among trees and minimize overfitting
  1. Take a boostrap sample from training data; 2. Build decision tree using this sample
- **Feature Randomness**: standard decision tree: each node, consider all features to find best split ↔ Random Forest: at each node only a random subset of features, best feature from this subset is used to make the split → diversity among trees and more robust

### 5.4.3. Training Parameters

- Number of Trees: The more trees, the less likely the model will overfit.
- Same as decision tree

### 5.4.4. Use-cases

- Predictive Maintenance
- Recommendation Systems
- Financial Modeling

| Strengths | Weaknesses |
|---|---|
| High accuracy | Computationally expensive |
| Low overfitting risk | Less interpretable compared to a single decision tree |
| Handles unbalanced datasets well | Not ideal for real-time predictions |

### 5.4.5. Data Preprocessing

- Feature Scaling: Generally not required.
- Missing Values: Can handle missing values, but better to impute.
- Categorical Variables: Label encoding usually sufficient.

### 5.4.6. Interpretability

Moderate; while individual trees are interpretable, the ensemble model complicates interpretation. However, feature importance can still be extracted.

## 5.5. Support Vector Machine (SVM)

Effective for both linear and non-linear classification, regression, and outlier detection.

### 5.5.1. Assumptions

- No Noise: Assumes that the data is clean and that there are clear margins of separation.
- Two Classes: For basic SVM, the algorithm is inherently binary.

### 5.5.2. Equation

The objective is to maximize the margin, defined by $\frac{2}{\|w\|}$, where $w$ is the weight vector, subject to constraints $y_i(w \cdot x_i + b) \geq 1$.

### 5.5.3. Training Parameters

Kernel: Specifies the type of hyperplane used to separate the data.

- Linear Kernel: $K(x, y) = x \cdot y$
- Polynomial Kernel: $K(x, y) = (1 + x \cdot y)^d$
- RBF Kernel: $K(x, y) = e^{-\gamma||x-y||^2}$

Regularization (C parameter): Balances classification and margin maximization.

### 5.5.4. Use-cases

- Text classification
- Image recognition
- Bioinformatics (e.g., cancer classification)

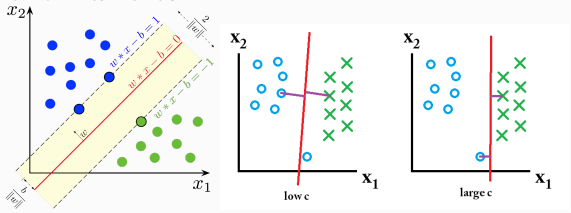
| Strengths | Weaknesses |
|---|---|
| High accuracy | Computationally expensive |
| Good for high-dimensional spaces | Sensitive to choice of kernel and regularization |
| Handles non-linear data well | Difficult to interpret |

### 5.5.5. Data Preprocessing

- Feature scaling: Essential due to distance-based optimization.
- Missing values: Must be handled prior.
- Categorical variables: Ordinal Encoding or One-Hot Encoding.

### 5.5.6. Interpretability

Low; the model parameters are hard to interpret in the context of the data, especially for non-linear kernels.










## 5.6. k-Nearest Neighbors (k-NN)

A non-parametric method used for classification and regression tasks based on similarity measures.

### 5.6.1. Assumptions

- No underlying model: Assumes no prior knowledge about the underlying data distribution.
- Similarity Measure: Assumes similar instances are near each other in the feature space.

### 5.6.2. Architecture

Classification is performed by a majority vote among the k-nearest points. Distance measures can include Euclidean, Manhattan, etc.

### 5.6.3. Training Parameters

- $k$: Number of neighbors to consider.
- Distance Metric: How to measure the distance between points (e.g., Euclidean, Manhattan).
- Weighting: Assign weights to contributions of the neighbors.

### 5.6.4. Use-cases

- Text categorization
- Fraud detection
- Recommender systems

| Strengths | Weaknesses |
|---|---|
| Simple to implement | Computationally expensive |
| Adapts easily to multi-class problems | Sensitive to irrelevant features |
| Effective if the feature dimension is low | Requires meaningful distance function |

### 5.6.5. Data Preprocessing

- Feature scaling: Crucial because k-NN uses distance measures.
- Missing values: Must be handled; otherwise, they'll impact distance calculations.
- Categorical variables: Ordinal Encoding or One-Hot Encoding.

### 5.6.6. Interpretability

Moderate; results can be interpreted by examining the closest neighbors, though this becomes harder as dimensionality increases.

## 5.7. K-Means Clustering

An unsupervised learning technique for partitioning data into clusters based on similarity.

### 5.7.1. Assumptions

- Distance Metric: Assumes Euclidean distance as a similarity measure.
- Spherical Clusters: Assumes clusters to be spherical and equally sized.

### 5.7.2. Architecture

Objective is to minimize the sum of squared distances from each point to its cluster centroid.

$$J = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

where $J$ is the objective function, $C_i$ is the $i^{th}$ cluster, and $\mu_i$ is the centroid of $C_i$.

**K-Means**

1. Initialize $k$ cluster centroids randomly or based on some criterion.
2. Assign each data point to the nearest centroid, and it becomes a member of that cluster.
3. Recalculate the centroid of each cluster as the mean vector of all points in the cluster.
4. Repeat steps 2-3 until convergence or a set number of iterations.

**K-Means++**

1. Choose one centroid uniformly at random from the data points.
2. For each data point, compute its distance to the nearest, already chosen centroid.
3. Select the next centroid from the remaining data points with probability proportional to the square of the distance calculated in the previous step.
4. Repeat steps 2-3 until $k$ centroids are chosen.
5. Proceed with steps 2-4 of the standard K-Means algorithm.

### 5.7.3. K-Means vs K-Means++

- **Initialization**: K-Means++ provides a smarter initialization than the random initialization in K-Means.
- **Convergence**: K-Means++ often requires fewer iterations to converge, resulting in performance improvements.
- **Quality**: K-Means++ generally produces clusters with lower intra-cluster variance compared to K-Means.

### 5.7.4. Training Parameters

- $k$: Number of clusters.
- Initialization: Method for initializing centroids (e.g., random, k-means++).
- Max Iterations: Maximum number of iterations for the algorithm to run.
- Tolerance: Convergence criteria.

### 5.7.5. Use-cases

- Market segmentation
- Document clustering
- Anomaly detection

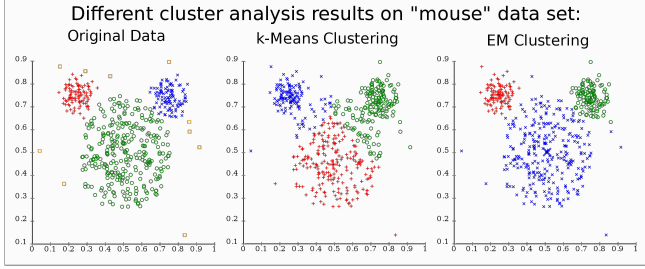
| Strengths | Weaknesses |
|---|---|
| Simple and fast | Must specify $k$ in advance |
| Works well for spherical clusters | Sensitive to initial conditions |
| Easily interpretable | Struggles with different cluster sizes |

### 5.7.6. Data Preprocessing

- Feature scaling: Important due to the use of distance metrics.
- Missing values: Need to be handled.

### 5.7.7. Interpretability

High; centroids and clusters can be easily examined, but interpretation can be subjective.



### 5.8. Hierarchical Clustering

A clustering algorithm that builds a hierarchy of clusters by either successively merging smaller clusters into larger ones (Agglomerative), or recursively dividing larger clusters into smaller ones (Divisive).

#### 5.8.1. Assumptions

- No assumptions about the shape or size of the clusters.
- Proximity Measures: Various distance metrics can be used (e.g., Euclidean, Manhattan).

#### 5.8.2. Linkage Criteria

Defines the rule for calculating distance between clusters.

- Single Linkage: Minimum distance between clusters.
- Complete Linkage: Maximum distance between clusters.
- Average Linkage: Average distance between clusters.
- Ward's Method: Minimizes the total within-cluster variance.

#### 5.8.3. Algorithm Steps

1. Start with each data point as a single cluster.
2. Merge the closest pair of clusters based on linkage criteria.
3. Update the distance matrix.
4. Repeat steps 2-3 until only one cluster remains.

#### 5.8.4. Use-cases

- Phylogenetic analysis
- Market segmentation
- Document clustering

| Strengths | Weaknesses |
|---|---|
| No need to specify number of clusters | Computationally expensive |
| Suitable for non-spherical clusters | Sensitive to choice of linkage criteria |
| Produces a dendrogram, useful for interpretation | No objective way to decide number of clusters |

#### 5.8.5. Data Preprocessing

- Feature scaling: Often necessary due to distance metrics.
- Missing values: Should be treated carefully.

#### 5.8.6. Interpretability

High; the dendrogram provides a visual representation and can help understand the nested cluster structure, but choosing the right number of clusters can be subjective.

### 5.9. Naive Bayes

Probabilistic classifiers based on Bayes' theorem, with strong independence assumptions between features.

#### 5.9.1. Assumptions

- Conditional Independence: Assumes that features are conditionally independent given the class label.
- Prior Probability: Requires an estimate of the prior probability of each class.

#### 5.9.2. Equation

The posterior probability for a class $C$ given features $x_1, x_2, \ldots, x_n$ is:

$$P(C|x_1, x_2, \ldots, x_n) \propto P(C) \prod_{i=1}^{n} P(x_i|C),$$

where $P(C)$ is the prior of class C and $P(x_i|C)$ the likelihood

#### 5.9.3. Training Parameters

- Smoothing: Additive smoothing (Laplace or Lidstone).
- Feature Type: Multinomial, Gaussian, or Bernoulli.

#### 5.9.4. Use-cases

- Spam filtering
- Sentiment analysis
- Document classification

| Strengths | Weaknesses |
|---|---|
| Simple and fast | Assumes feature independence |
| Works well with high dimensions | Sensitive to irrelevant features |
| Effective with small data | Limited to categorical data |

#### 5.9.5. Data Preprocessing

- Feature selection: Useful to remove irrelevant features.
- Data transformation: Often required for Gaussian Naive Bayes.

#### 5.9.6. Interpretability

High; easily understandable probabilities and strong assumptions make the model interpretable.

### 5.10. Neural Networks

A class of models inspired by biological neural networks, mainly used for complex pattern recognition and nonlinear function approximation.

#### 5.10.1. Assumptions

- Universality: Capable of approximating any function given enough neurons.
- Data-Driven: Does not impose explicit assumptions on the underlying data distribution.

#### 5.10.2. Equation

A typical layer in a neural network is defined as:

$$h = \sigma(Wx + b)$$

where $h$ is the output, $\sigma$ is the activation function, $W$ is the weight matrix, $x$ is the input, and $b$ is the bias.

#### 5.10.3. Training Parameters

- Learning Rate: Step size in optimization algorithm.
- Activation Function: ReLU, Sigmoid, Tanh, etc.
- Epochs: Number of times the entire dataset is passed through the network.
- Batch Size: Number of samples used in each update.

#### 5.10.4. Use-cases

- Image classification
- Natural language processing
- Game playing

| Strengths | Weaknesses |
|---|---|
| Highly flexible | Requires large data sets |
| Good for complex tasks | Computationally intensive |
| Self-learning features | Difficult to interpret |

#### 5.10.5. Gradient Descent

An optimization algorithm for minimizing the cost function $J(\theta)$ iteratively.

**Steps**

1. **Initialize Weights**: Randomly initialize model weights $\theta$.
2. **Compute Gradient**: Calculate the gradient $\nabla J(\theta)$ of the cost function with respect to the weights.

$$\nabla J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla J_i(\theta)$$

3. **Update Weights**: Update the weights in the opposite direction of the gradient.

$$\theta = \theta - \alpha \nabla J(\theta)$$

where $\alpha$ is the learning rate.

#### 5.10.6. Stochastic Gradient Descent (SGD)

A variant of gradient descent that updates weights after each training example.

**Steps**

1. **Initialize Weights**: Randomly initialize model weights $\theta$.
2. **Randomize Data**: Shuffle the training dataset.
3. **Compute & Update**: For each training example $x_i$, compute the gradient $\nabla J_i(\theta)$ and immediately update $\theta$.

$$\theta = \theta - \alpha \nabla J_i(\theta)$$

#### 5.10.7. Comparative Analysis

| Strengths | Weaknesses |
|---|---|
| Fast updates | Noisy updates |
| Good for large datasets | May not converge to global minimum |
| Efficient in terms of memory | Sensitive to learning rate |

#### 5.10.8. Data Preprocessing

- Feature scaling: Generally required.
- Missing values: Not well-suited; must be handled externally.

#### 5.10.9. Interpretability

Low; often considered as "black-box" models due to their complex structure.

### 5.11. Time Series Forecasting

Using historical data points to predict future observations.

#### 5.11.1. Characteristics

**Stationarity**
Statistical properties like mean and variance are constant over time, often a key issue in time series forecasting.

**Seasonality**
Patterns that repeat at known intervals (e.g., daily, monthly, annually).

**Trend**
The underlying trend in the data. Can be upward, downward, or stable.

**Cycles**
Long-term oscillations driven by economic conditions. Unlike seasonality, the duration is unpredictable.

**Autocorrelation**
The correlation of a series with its own lags.

#### 5.11.2. Common Methods

**Exponential Smoothing**
A weighted average method that considers the most recent observations to be more relevant. The simplest form is Simple Exponential Smoothing for univariate time series without trend and seasonality.

**ARIMA (AutoRegressive Integrated Moving Average)**
A linear equation that utilizes time-lagged values, lagged forecast errors, and a trend component. Used for non-seasonal time series data.

**Assumptions:**

- Stationarity: Assumes the data is stationary or can be made stationary through differencing.
- Linearity: Assumes a linear relationship between lagged variables.

**Equation** The ARIMA model can be represented as ARIMA$(p, d, q)$, where $p$ is the AR term, $d$ is the number of differencing required to make the series stationary, and $q$ is the MA term.

$$(1 - \Phi_1 B - \ldots - \Phi_p B^p)(1 - B)^d Y_t = (1 + \theta_1 B + \ldots + \theta_q B^q)\epsilon_t$$

| Strengths | Weaknesses |
|---|---|
| Effective for univariate time-series | Sensitive to parameter choices |
| Handles seasonality with SARIMA | Requires stationary data |
| Good interpretability | Complexity increases with parameters |

**SARIMA (Seasonal ARIMA)**
An extension of ARIMA that explicitly models seasonality. Useful for time series with seasonal components.

**Prophet**
An open-source forecasting tool designed for business forecast tasks, providing intuitive parameters to handle trends, seasonality, and holidays.

#### 5.11.3. Use-cases

- Stock price forecasting
- Sales prediction
- Weather forecasting

## 6. Statistics

### 6.1. Descriptive Statistics

Basic metrics to understand data distributions:

- Mean: $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$
- Median: Middle value in a sorted list
- Mode: Most frequently occurring value
- Variance: $\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$
- Standard Deviation: $\sigma = \sqrt{\sigma^2}$

### 6.2. Inferential Statistics

Tools for making predictions and inferences from sampled data.

- Confidence Interval: $\text{CI} = \bar{x} \pm Z\left(\frac{\sigma}{\sqrt{N}}\right)$
- Hypothesis Testing: Using p-values and significance level ($\alpha$) to make decisions.

### 6.3. Probability Distributions

Different types of data distributions:

- Normal Distribution: $f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/(2\sigma^2)}$
- Binomial Distribution: $f(k, n, p) = C(n, k)p^k(1-p)^{(n-k)}$
- Poisson Distribution: $P(x; \lambda) = \frac{\lambda^x e^{-\lambda}}{x!}$

#### 6.3.1. Probability Formulas

- Probability: $P(A) = \frac{\text{Number of favorable outcomes}}{\text{Total number of outcomes}}$
- Conditional Probability: $P(A|B) = \frac{P(A \cap B)}{P(B)}$
- Bayes' Theorem: $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$
- Expected Value: $E(X) = \sum_{i=1}^{n} p(x_i)x_i$
- Variance: $\text{Var}(X) = E(X^2) - [E(X)]^2$
- Standard Deviation: $\sigma = \sqrt{\text{Var}(X)}$
- Covariance: $\text{Cov}(X, Y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})$
- Pearson Correlation Coefficient: $r = \frac{\text{Cov}(X,Y)}{\sigma_x \sigma_y}$

## 7. Other

### 7.1. Multiclass Prediction

Output variable can take on more than two classes, aims to assign an observation to one of several classes.

- **One-vs-All (OvA)**:
  - *Method*: For $N$ classes, $N$ separate binary classifiers are trained. Each one discriminates between one of the classes and the rest.
  - *Advantages*: Simplicity and ease of implementation.
- **One-vs-One (OvO)**:
  - *Method*: For $N$ classes, $\binom{N}{2}$ classifiers are trained, one for each pair of classes.
  - *Advantages*: Each classifier needs only to be trained on the subset of the data containing its two target classes, potentially providing more accurate results.
- **Softmax Regression**:
  - *Method*: Generalizes logistic regression to multiple classes without requiring multiple binary classifiers.
  - *Advantages*: Provides probabilities for each class, useful for interpretability.
- **Tree-based Methods**:
  - *Method*: Decision trees or ensemble methods like Random Forest can naturally handle multiclass classification.
  - *Advantages*: Interpretability and handling of missing values.

### 7.2. Dimensionality Reduction

Techniques to reduce the number of features while preserving the underlying data structure. Useful for visualization, reducing noise, and improving model performance.

#### 7.2.1. Principal Component Analysis (PCA)

- Objective: To find orthogonal axes (principal components) that maximize the variance.
- Algorithm: Eigen-decomposition of the covariance matrix.
- Use-cases: Image compression, visualization, feature extraction.

**Steps**

1. **Standardize Data**: Scale the data to have zero mean and unit variance.
$$X_{\text{std}} = \frac{X - \mu}{\sigma}$$
2. **Compute Covariance Matrix**: Calculate the covariance matrix of the standardized data.
$$\Sigma = \frac{1}{n} X_{\text{std}}^T X_{\text{std}}$$
3. **Eigenvalue Decomposition**: Calculate the eigenvalues and eigenvectors of the covariance matrix.
$$\Sigma v = \lambda v$$
4. **Sort Eigenvalues**: Sort the eigenvalues in descending order and select the top $k$ corresponding eigenvectors.
5. **Projection**: Project the original data onto the lower-dimensional subspace.
$$X_{\text{pca}} = X_{\text{std}} \times W$$
where $W$ is the matrix formed by the top $k$ eigenvectors.

#### 7.2.2. Linear Discriminant Analysis (LDA)

- Objective: To find a linear combination of features that maximizes the separation between different classes.
- Algorithm: Maximizes the ratio of between-class variance to within-class variance.
- Use-cases: Classification, feature extraction.

**Steps**

1. **Compute Class Means**: Calculate the mean vectors for each class.
$$\mu_i = \frac{1}{n_i} \sum_{x \in C_i} x$$
2. **Within-class Scatter Matrix** $S_W$:
$$S_W = \sum_{i=1}^{c} \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T$$
3. **Between-class Scatter Matrix** $S_B$:
$$S_B = \sum_{i=1}^{c} n_i (\mu_i - \mu)(\mu_i - \mu)^T$$
where $\mu$ is the overall mean.
4. **Eigenvalue Decomposition**: Solve the generalized eigenvalue problem for $S_W^{-1} S_B$.
$$S_W^{-1} S_B v = \lambda v$$
5. **Sort Eigenvalues**: Sort the eigenvalues in descending order and select the top $k$ corresponding eigenvectors.
6. **Projection**: Project the original data onto the lower-dimensional subspace.
$$X_{\text{lda}} = X \times W$$
where $W$ is the matrix formed by the top $k$ eigenvectors.

#### 7.2.3. Factor Analysis

- Objective: To model observed variables as linear combinations of latent variables and error terms.

$$\underset{\text{p}\times\text{n}}{\begin{bmatrix}\text{math scores}\\ \text{reading scores}\\ \text{science scores}\end{bmatrix}} = \underset{\text{p}\times\text{k}}{\begin{bmatrix}.13 & .95\\ .78 & -.28\\ -.87 & .05\end{bmatrix}} \underset{\text{k}\times\text{n}}{\begin{bmatrix}-1.25 & 1.88 & \dots & -0.55\\ 0.71 & -0.17 & \dots & -1.20\end{bmatrix}}$$

(data = factor loadings × common factors)

- Algorithm: Factor loadings are estimated through methods like maximum likelihood.
- Use-cases: Psychology tests, market research, gene expression data.

| Strengths | Weaknesses |
|---|---|
| Reduces overfitting | Loss of interpretability |
| Speeds up training | Assumes linear relationships |
| Improves model generalization | May lose important features |

#### 7.2.4. Data Preprocessing

- Feature scaling: Mandatory for most algorithms due to distance computation.
- Handling missing values: Impute or remove rows/columns.

#### 7.2.5. Interpretability

Medium; interpreting reduced dimensions may be non-intuitive but techniques like loading plots can help.

### 7.3. Hypothesis Testing in A/B Testing

#### 7.3.1. Steps

1. **Null Hypothesis ($H_0$)**: Assume that there is no difference between the performance of model A and model B.
2. **Alternative Hypothesis ($H_a$)**: Assume that there is a significant difference between the performance of model A and model B.
3. **Choose Significance Level ($\alpha$)**: Commonly set at 0.05, implying a 95% confidence level.
4. **Collect Data**: Record the performance metric for both groups over a set period.
5. **Calculate Test Statistic**: This could be a t-statistic, z-score, etc., depending on the data distribution.
6. **Compute P-value**: The probability of observing the data if the null hypothesis were true.
7. **Draw Conclusions**:
   - If $p \leq \alpha$, reject the null hypothesis. This suggests that the difference in performance is statistically significant.
   - If $p > \alpha$, fail to reject the null hypothesis. This suggests that the difference in performance is not statistically significant.

#### 7.3.2. Important Notes

- A smaller p-value suggests stronger evidence against the null hypothesis.
- Rejecting the null hypothesis does not prove the alternative hypothesis; it only suggests that the alternative hypothesis is more likely.
- Hypothesis testing only indicates if a difference exists, not the magnitude of the difference.