**Memo**



Essential ML Terms

## **1️⃣ Overfitting & Underfitting**

* **Overfitting:** Model learns the training data too well, including noise, leading to poor generalization.
* **Underfitting:** Model is too simple to capture patterns, leading to poor performance on both training and test data.
* **In simple:** Model performs well on training data, and poor on unseen(test) data

## **2️⃣ Generalization**

* The ability of a model to perform well on unseen data.
* Good generalization means the model captures the underlying pattern rather than memorizing the data.

## **3️⃣ Data Leakage**

* When information from outside the training dataset influences the model, leading to unrealistic performance.
* Example: Using target variable-related features while training

## **4️⃣ Bias-Variance Tradeoff**

* **High Bias (Underfitting):** Model is too simple and fails to capture patterns.
* **High Variance (Overfitting):** Model is too complex and captures noise.
* **Solution:** Choose the right complexity, use cross-validation, and collect more data.

## **5️⃣ Curse of Dimensionality**

* When the number of features is too high, it makes models inefficient and prone to overfitting.
* **Solution:** Use dimensionality reduction (PCA, LDA) or feature selection methods.

## **6️⃣ Class Imbalance**

* When one class is significantly underrepresented in classification problems (e.g., fraud detection, medical diagnosis).
* **Solution:** Use oversampling, undersampling, or class-weighted models.

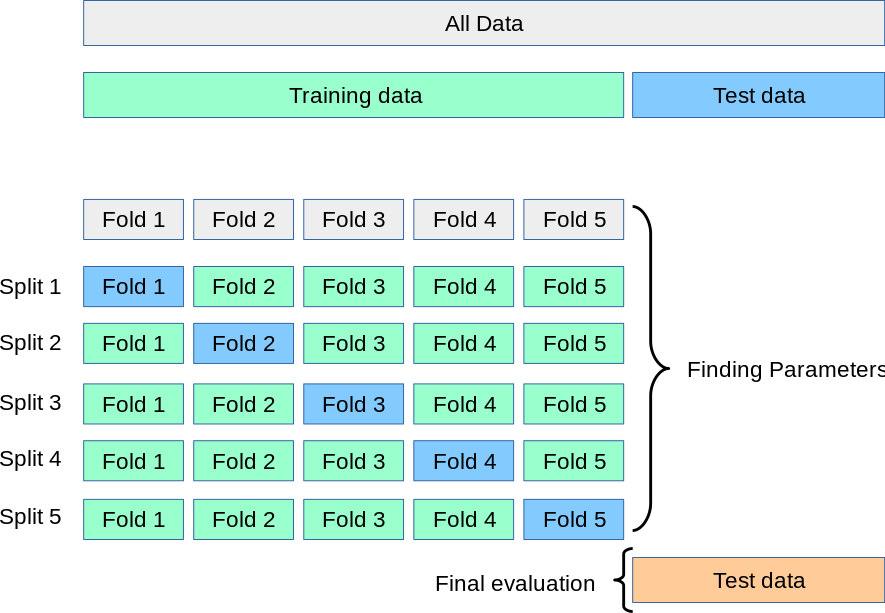
### **7️⃣ Oversampling:**

* **Increases the minority class samples** by duplicating or generating synthetic data to balance class distribution.
* Helps **prevent bias** toward the majority class but may lead to overfitting if not done carefully.

### **8️⃣ Undersampling:**

* **Reduces the majority class samples** by randomly removing instances to balance the dataset.
* Helps **speed up training** and reduce storage needs but may lead to **loss of important information.**

## **9️⃣ Cross-Validation**



## 

* Splitting the dataset into multiple parts to test the model’s performance on unseen data.
* **Types:**
  + **K-Fold Cross-Validation:** Splits data into K subsets and rotates training/testing.
  + **Stratified K-Fold:** Ensures class distribution remains the same across folds.

## **🔟 Feature Engineering**

* The process of creating new features from raw data to improve model performance.
* **Examples:**
  + Converting dates into day/month/year features.
  + Extracting text embeddings from documents.

## **1️⃣1️⃣ Feature Selection vs. Feature Importance**

* **Feature Selection:** Choosing the most relevant features (RFE, SelectKBest).
* **Feature Importance:** Understanding which features impact predictions (SHAP, Tree-based models).

## **1️⃣2️⃣ Dimensionality Reduction**

* Reducing the number of features while preserving information.
* **Methods:** PCA (Principal Component Analysis), LDA (Linear Discriminant Analysis).

## **1️⃣3️⃣ Hyperparameter Tuning**

* Optimizing model parameters like learning rate, tree depth, etc.
* **Techniques:** GridSearchCV, RandomizedSearchCV

## **1️⃣4️⃣ Regularization (L1 & L2)**

* Prevents overfitting by adding a penalty to large coefficients in models.
* **L1 Regularization (Lasso):** Shrinks some coefficients to zero (feature selection).
* **L2 Regularization (Ridge):** Reduces all coefficients but doesn’t eliminate them.

## **1️⃣5️⃣ Bootstrapping & Bagging**

* **Bootstrapping:** Sampling with replacement to create multiple datasets.
* **Bagging (Bootstrap Aggregating):** Training multiple models on different bootstrap samples and averaging predictions (e.g., Random Forest).

## **1️⃣6️⃣ Boosting**

* A sequential learning method where weak models are trained iteratively, correcting previous errors.
* **Examples:** AdaBoost, Gradient Boosting, XGBoost, LightGBM.

## **1️⃣8️⃣ One-Hot Encoding vs. Label Encoding**

* **One-Hot Encoding:** Converts categorical variables into binary vectors.
* **Label Encoding:** Assigns a numerical value to each category.

## **1️⃣9️⃣ F1-Score, Precision, Recall, ROC-AUC**

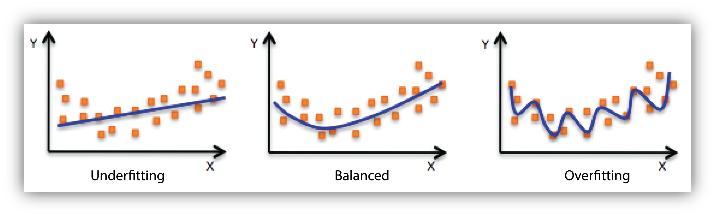
* **Precision:** How many predicted positives are actually positive?
* **Recall:** How many actual positives were correctly identified?
* **F1-Score:** Harmonic mean of precision and recall.
* **ROC-AUC:** Measures the trade-off between true positive rate and false positive rate.

### **2️⃣**0️⃣**Regression Evaluation Metrics**

* **Mean Absolute Error (MAE)** – Measures the average absolute difference between predicted and actual values. Lower is better.
* **Mean Squared Error (MSE)** – Penalizes larger errors more than MAE by squaring the differences. Lower is better.
* **Root Mean Squared Error (RMSE)** – Square root of MSE, making it interpretable in the same unit as the target variable.
* **R-squared (R2)** – Indicates how well the model explains the variance in the target variable. Closer to 1 is better.

## **2️⃣**1️⃣ **Anomaly Detection**

* Identifying rare or abnormal patterns in data (e.g., fraud detection, network intrusion).
* **Techniques:** Isolation Forest, One-Class SVM, Autoencoders.



## 📖 Real-World Analogy: Overfitting, Underfitting & Generalization

Imagine a **student preparing for a math exam.** 📚

### **🔴 Overfitting: The Memorization Trap**

* The student **memorizes** every question from the model paper instead of understanding concepts.
* On the practice test (training data), he scores **100%** since the questions are the same. ✅
* However, in the final exam (test data), when faced with unseen problems, he **fails** because he cannot apply concepts. ❌

📌 **In ML:** The model learns noise and patterns that are specific to training data but don’t generalize to new data.

### **🟡 Underfitting: The Clueless Student**

###### The student **barely studies** and just skims through basic formulas without solving problems.

###### In the practice test (training data), he performs **poorly** because he hasn’t learned enough. ❌

###### In the final exam (test data), he also performs **poorly** because he doesn’t understand the subject. ❌

###### 📌 **In ML:** The model is too simple and doesn’t capture patterns, leading to bad predictions on both training and test data.

### **🟢 Generalization: The Balanced Learner**

* The student **understands concepts** and practices solving different types of problems instead of just memorizing.
* On the practice test (training data), he performs **well but not perfectly** because he’s solving questions logically. ✅
* In the final exam (test data), he performs **well** because he can apply his knowledge to unseen problems. ✅

📌 **In ML:** The model learns the general pattern in data and performs well on unseen data.

### **Conclusion🎯**

A good ML model should **learn concepts, not just memorize data.** Balancing **bias and variance** helps a model generalize well, just like a student who understands math instead of just memorizing.

### **Some Interview Questions**

#### **1️⃣ Difference Between Linear and Logistic Regression**

| **Feature** | **Linear Regression** | **Logistic Regression** |
| --- | --- | --- |
| **Type** | Regression | Classification |
| **Output** | Continuous values | Probabilities (0-1) |
| **Equation** | **y=mx+c** | Z = mx+c |
| **Used for** | Predicting numerical values | Predicting class labels |

#### 

#### **2️⃣ What is Train-Test Splitting and Why?**

* **Train-Test Split**: Divides data into **training** (to learn patterns) and **testing** (to evaluate performance).
* **Why?** To check model generalization and prevent overfitting.

#### **3️⃣ Why Scaling?**

* Ensures all features contribute equally, preventing dominance of large-scale features.
* Helps models converge faster.

#### **4️⃣ Which Models Require Scaling?**

| **Model Type** | **Scaling Required?** |
| --- | --- |
| Linear Regression | ✅ Yes |
| SVM | ✅ Yes |
| Logistic Regression | ✅ Yes |
| KNN | ✅ Yes |
| Decision Trees | ❌ No |
| Random Forest | ❌ No |
| XGBoost | ❌ No |

#### 

#### 

#### **5️⃣ Why Tree-based models don’t need scaling?**

#### Tree models use Gini impurity, entropy, or variance reduction to determine splits.

#### These are independent of feature scales, so standardization or normalization does not change the decision boundaries

* Decision trees split data **based on feature thresholds** (e.g., **"Is Age > 30?"**). They **do not** rely on Euclidean distances,so the magnitude of feature values does not affect the splits.
* Random Forest and Gradient Boosting **combine multiple trees**, each making independent decisions based on feature splits

#### 

#### **6️⃣ Why Encoding?**

* Machine learning models work with numbers, not text.
* Categorical variables need encoding to be used in models.

**Encoding Methods**

### **🔢 1. Label Encoding**

#### Converts categories into integers (e.g., red, blue, green → 0, 1, 2)

#### ✅ Use when:

#### Categories are ordinal (have a meaningful order)

#### ❌ Not recommended for **linear models** or **distance-based models** (e.g., KNN, SVM) when the data is **nominal** — it can introduce fake ordinal relationships.

#### 

### **🧮 2. Ordinal Encoding**

#### Categories are ordinal or can define the order explicitly (e.g., low < medium < high)

#### ✅ Use when:

#### The categories have a natural rank/priority

#### 

### **🧊 3. One-Hot Encoding (OHE)**

#### Creates a new binary column for each category

#### red, blue, green → is\_red, is\_blue, is\_green

#### ✅ Use when:

#### Categories are nominal (no order)

#### The number of unique values is low to moderate

#### ⚠️ Avoid with high cardinality (creates too many columns)

#### 

### **🔁 4. Frequency Encoding**

#### Replace each category with its frequency count or proportion

#### ✅ Use when:

#### High-cardinality features

#### Models like Trees (Random Forest, XGBoost)

#### ⚠️ Can leak target info if not used carefully

#### 

### **🔡 5. Binary Encoding**

#### Converts categories into binary format and splits digits into separate columns

#### Category → Integer → Binary → Split bits

**Example:**

city = [‘Paris’,’London’]

* + - * ‘Paris’ → 1 -→ 01 → column\_1 (0) → column\_2(1)
      * ’London’ →2 -→ 10 → column\_1 (1) → column\_2(0)

#### ✅ Use when:

#### You have high-cardinality categorical features

#### Less sparse than OHE

#### 

### **🎯 6. Target Encoding (Mean Encoding)**

#### Replaces a category with the mean of the target variable

#### ✅ Use when:

#### You’re working with high-cardinality features

#### You’re using models like Logistic/Linear Regression

#### ⚠️ Must use cross-validation to prevent target leakage

#### 

### **📊 Summary Table**

| **Encoder** | **Ordered Data** | **High Cardinality** | **Risk of Leakage** | **Model Compatibility** |
| --- | --- | --- | --- | --- |
| **Label Encoding** | ✅Sometimes | ❌ | ❌ | Trees (OK), Linear (⚠️ risky) |
| **Ordinal Encoding** | ✅ Yes | ❌ | ❌ | Trees, Linear |
| **One-Hot Encoding** | ❌ No | ❌ Avoid | ❌ | Trees, Linear |
| **Frequency Encoding** | ❌ No | ✅ Yes | ⚠️ Possible | Trees |
| **Binary Encoding** | ❌ No | ✅ Yes | ❌ | Trees, Linear |
| **Target Encoding** | ❌ No | ✅ Yes | ✅ High | Linear (Good), Trees (CV needed) |

#### 

#### **7️⃣ When to Use Transformations?**

| **Scenario** | **Transformation** |
| --- | --- |
| **Right-skewed data** (long right tail) | Log, Square Root, Box-Cox |
| **Left-skewed data** (long left tail) | Power Transformation, Exponential |
| **Data with varying magnitudes** | MinMaxScaler |
| **Normally distributed data** | StandardScaler |

#### 

#### 

#### 

#### **8️⃣ Handling Outliers**

* **IQR Method**: Cap values within [Q1−1.5×IQR,Q3+1.5×IQR][Q1 - 1.5 \times IQR, Q3 + 1.5 \times IQR].
* **Winsorization**: Replace extreme values with nearest percentile values.
* **Transformations**: Apply log or power transformation to reduce extreme values.

#### 

#### **9️⃣ What is Skewness and How to Handle It?**

* **Skewness**: Measure of asymmetry in data distribution.
* **Handling Methods**: Using different transformations

**🔟 Advantages of Tree-Based Models Over Linear Models**

| **Feature** | **Tree-Based Models** | **Linear Models** |
| --- | --- | --- |
| **Handles non-linearity** | ✅ Yes | ❌ No |
| **Feature interactions** | ✅ Yes | ❌ No |
| **Works with missing data** | ✅ Yes | ❌ No |
| **No need for scaling** | ✅ Yes | ❌ No |
| **Feature selection** | ✅ Yes | ❌ No |

#### 

#### 

#### 

#### **1️⃣1️⃣ What is Entropy?**

* A measure of disorder or randomness in data.
* In decision trees, lower entropy means purer nodes.

#### **1️⃣2️⃣ What is Information Gain?**

* Measures how much a feature reduces uncertainty in the target variable.

**1️⃣3️⃣ How Do Tree-Based Models Calculate Feature Importances?**

* **Gini Importance**: Measures how much a feature reduces impurity.
* **Permutation Importance**: Shuffles a feature and observes performance drop.

**1️⃣4️⃣When should we prioritize recall over precision, and vice versa?**

* **Recall is important** when **missing a positive case** has severe consequences.  
   📌 *Example: In a* ***medical diagnosis for cancer****, missing a positive case (False Negative) could be life-threatening. So, we prioritize recall to* ***catch all possible positive cases****, even if some are false positives.*
* **Precision is important** when **false positives have serious consequences.** 📌 *Example: In a* ***spam email filter****, marking an important work email as spam (False Positive) is bad. So, we prioritize precision to* ***reduce false alarms****, even if some spam emails slip through.*

**1️⃣5️⃣ When should we use the F1-score instead of precision or recall?**

📌 **Use F1-score when you need a balance between precision and recall**, especially when:

* The dataset is **imbalanced** (one class has much fewer samples).
* Both **false positives and false negatives are equally important**.

🔹 **Real-Life Examples:**

👉 **Hiring Process (Resume Screening)**

* If a model screens resumes for job applications:  
  + **Precision-focused**: Selects **only** candidates who match **all** required skills, but might miss some who are a good fit.
  + **Recall-focused**: Selects **many** candidates, even if they have **only one** matching skill
  + **F1-score**: Balances both, ensuring selected candidates **mostly** match the job requirements without excluding too many.

👉 Imagine searching for **"best Python courses"**:

* **High precision**: Shows only top-rated courses but misses some useful ones.
* **High recall**: Shows every Python-related course, including low-quality ones.
* **F1-score**: Strikes a balance, showing a mix of **highly relevant and diverse** results.

**1️⃣6️⃣What is Confusion Matrix?**

A **confusion matrix** is a table used to evaluate the performance of a classification model. It shows the number of:

* **True Positives (TP)**: Correctly predicted positive cases
* **True Negatives (TN)**: Correctly predicted negative cases
* **False Positives (FP)**: Incorrectly predicted as positive (Type I error)
* **False Negatives (FN)**: Incorrectly predicted as negative (Type II error)

### **🔹 Example Dataset & Confusion Matrix**

| ID | Actual  (True Label) | Predicted |
| --- | --- | --- |
| 1 | 1 | 1 |
| 2 | 0 | 0 |
| 3 | 1 | 0 |
| 4 | 0 | 1 |
| 5 | 1 | 1 |
| 6 | 0 | 0 |
| 7 | 1 | 1 |
| 8 | 0 | 0 |
| 9 | 1 | 1 |
| 10 | 0 | 1 |

### **Actual Positives (1):** IDs 1, 3, 5, 7, 9

### **Actual Negatives (0):** IDs 2, 4, 6, 8, 10

|  | **Predicted: Positive (1)** | **Predicted: Negative (0)** |
| --- | --- | --- |
| **Actual: Positive (1)** | **TP**: 4 | **FN**: 1 |
| **Actual: Negative (0)** | **FP**: 2 | **TN**: 3 |

**Interpretation:**

* **True Positives (TP)**  → Correctly predicted positives
* **True Negatives (TN)** → Correctly predicted negatives
* **False Positives (FP)**→ Incorrectly classified as positive
* **False Negatives (FN)** → Incorrectly classified as negative

### **🧮 From this we can calculate:**

* **Accuracy** = (TP + TN) / Total = (4 + 3)/10 = 0.70
* **Precision** = TP / (TP + FP) = 4 / (4 + 2) = 0.67
* **Recall** = TP / (TP + FN) = 4 / (4 + 1) = 0.80
* **F1 Score** = 2 × (Precision × Recall) / (Precision + Recall) ≈ 0.73

### **1️⃣7️⃣ Why is K-Nearest Neighbors considered a lazy or slow learner?**

* KNN is called a **lazy learner** because it **does not learn a model during training**. Instead, it simply stores the training data and waits until a new data point is given.
* At prediction time, it computes the distance between the new point and **all training points**, which can be **computationally expensive and slow** for large datasets.

### **1️⃣8️⃣ What are weak and strong learners in ensemble methods?**

* **Weak Learner**:  
   A model that performs **just slightly better than random guessing**.  
   For example, a decision stump (1-level decision tree) is a weak learner.
* **Strong Learner**:  
   A model that achieves **high accuracy** and generalizes well to unseen data.
* In ensemble methods like **Boosting**, multiple weak learners are combined **sequentially** to create a **strong learner**.
* In **Bagging (like Random Forest)**, several weak or strong models are trained in parallel and their outputs are aggregated.

### **🧠 Analogy:**

* A **weak learner** is like a student who gets just above passing marks.
* A **strong learner** is like a student who consistently scores high.
* Ensemble methods **combine the efforts** of many weak students to outperform the top student.

### **1️⃣9️⃣What happens if the value of k in KNN is too small or too large?**

### **✅ If k is too small (e.g., k=1):**

* The model becomes **very sensitive to noise** in the training data.
* Even an outlier can drastically affect the prediction.
* This leads to **high variance** and **overfitting**.

**Example**: A mislabeled point might make the model predict wrong for nearby test points.

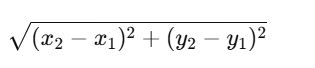
### **✅ If k is too large (e.g., k=100):**

* The model includes **too many neighbors**, even from other classes.
* It becomes **too generalized**, possibly missing important patterns.
* This leads to **high bias** and **underfitting**.

**🎯 Ideal Approach:**

* **Cross-validation** : Cross-validation helps test multiple values of k on different subsets of data, ensuring the selected k works well overall—not just on a single train-test split. This avoids **overfitting or underfitting**.
* A common starting point is to try odd numbers between 3 and 15.

### **2️⃣**0️⃣ **What is Euclidean Distance in KNN?**

* It's the **straight-line distance** between two points.
* Formula (2D):  
   
* Works well when the features are **continuous** and the **geometry of the space matters**.

### **2️⃣**1️⃣**What is Manhattan Distance in KNN?**

* It’s the **sum of absolute differences** between points (like moving in a grid or city blocks).
* Formula:  
   
* Useful when:  
  + You have **high-dimensional** or **sparse data** (like text features).
  + The movement is restricted to **right angles** (like chessboard or roads).

### **2️⃣2️⃣ When to Use Which?**

| **Scenario** | **Use Euclidean 📏** | **Use Manhattan 🛣️** |
| --- | --- | --- |
| Continuous, normalized data | ✅ Yes | ❌ Optional |
| High-dimensional data (e.g., NLP) | ❌ Risky | ✅ Better |
| Sparse features (mostly zeros) | ❌ Not ideal | ✅ Recommended |
| Grid-like movement or features | ❌ Less relevant | ✅ Best fit |

### 

### **⚠️ Important Tip:**

Always **scale your features** before using Euclidean or Manhattan distances in KNN. Otherwise, one feature can dominate the distance metric.

###### 