🧠 Core Topics to Prepare

1. Machine Learning Fundamentals

- Supervised vs. unsupervised learning
- Bias-variance tradeoff, overfitting/underfitting
- Evaluation metrics: accuracy, precision, recall, F1-score, ROC-AUC
- Feature engineering and selection techniques

2. Modeling & Algorithms

- Regression (linear, logistic), decision trees, random forests
- SVM, k-NN, Naive Bayes
- Ensemble methods: Bagging, Boosting (e.g., XGBoost, LightGBM)
- Deep learning basics (CNNs, RNNs, transformers if relevant)

3. Data Processing & Pipelines

- Data cleaning, handling missing values, outliers
- Normalization vs. standardization
- Pandas, NumPy workflows
- Real-time vs. batch processing (especially relevant to your video/image experience)

4. SQL & Databases

- Joins, aggregations, subqueries
- Window functions, CTEs
- Query optimization and indexing

5. Python & Libraries

- Scikit-learn, TensorFlow, PyTorch
- Writing modular, reusable code
- Exception handling and logging (especially for production-grade ML)

6. Cloud Platforms

- AWS: S3, EC2, SageMaker, Lambda
- Azure ML or GCP equivalents
- Deployment strategies: containerization (Docker), CI/CD pipelines

7. Problem-Solving & Case Studies

- Expect scenario-based questions like:
  - “How would you improve a model with low recall?”
  - “Design a pipeline for detecting anomalies in streaming data.”
  - “How would you deploy a model that updates weekly?”

🔍 Bonus Prep Tips

- **Brush up on your recent RAG pipeline notes**—they may ask about semantic search or embeddings if the role leans toward NLP.
- **Review your OCR and computer vision work**—even if not directly asked, it's a great differentiator.
- **Prepare a few STAR-format stories** (Situation, Task, Action, Result) to showcase your impact and problem-solving.

🧩 What Is SQL?

SQL (Structured Query Language) is the standard language for querying and manipulating relational databases. It's essential for data scientists to extract, transform, and analyze data efficiently.

🔍 Core SQL Concepts for Interviews

1. Data Retrieval

- `SELECT`: Pulls data from tables.
- `WHERE`: Filters rows based on conditions.
- `ORDER BY`: Sorts results.
- `LIMIT`: Restricts number of rows returned.

2. Joins

Used to combine data from multiple tables:

Join Type	Description
`INNER JOIN`	Returns matching rows from both tables
`LEFT JOIN`	All rows from left + matched from right
`RIGHT JOIN`	All rows from right + matched from left
`FULL JOIN`	All rows from both, matched where possible

3. Aggregation

- `GROUP BY`: Groups rows for aggregation.
- Functions: `COUNT()`, `SUM()`, `AVG()`, `MAX()`, `MIN()`
- `HAVING`: Filters groups (like `WHERE`, but for aggregates)

4. Subqueries

- Nested queries inside `SELECT`, `FROM`, or `WHERE`
- Useful for filtering based on computed values

5. Window Functions

- Operate over a set of rows related to the current row
- Examples: `ROW_NUMBER()`, `RANK()`, `LEAD()`, `LAG()`
- Syntax: `OVER(PARTITION BY ... ORDER BY ...)`

6. CTEs (Common Table Expressions)

- Temporary named result sets using `WITH`
- Improves readability and modularity of complex queries

7. Data Manipulation

- `INSERT`, `UPDATE`, `DELETE`
- `MERGE` (for upserts, depending on DBMS)

8. Optimization Tips

- Use indexes wisely
- Avoid `SELECT *` in production
- Watch out for Cartesian joins (missing `ON` clause)

🧠 Sample Interview Challenge

> "Find the top 3 customers by total purchase amount in the last 6 months."

This tests:

- Date filtering (`WHERE`)
- Aggregation (`SUM`)
- Ranking (`ROW_NUMBER()` or `RANK()`)

Technoques related to Data processing

🧹 1. **Data Cleaning**

Ensures data quality and consistency.

- **Handling missing values**: Imputation (mean, median, mode), deletion, or model-based filling
- **Outlier detection**: Z-score, IQR method, DBSCAN
- **Noise reduction**: Smoothing, binning, or filtering (e.g., Gaussian filters for CV)
- **Data type correction**: Ensuring correct formats (e.g., datetime parsing)

🔄 2. **Data Transformation**

Prepares data for modeling.

- **Normalization**: Scales data to [0,1] (e.g., MinMaxScaler)
- **Standardization**: Centers data around mean with unit variance (e.g., StandardScaler)
- **Encoding categorical variables**:

    - One-hot encoding
    - Label encoding
    - Target encoding (for tree-based models)
- **Log transformation**: For skewed distributions
- **Binning**: Converts continuous data into discrete intervals

---

🧠 3. **Feature Engineering**

Creates new features to improve model performance.

- **Polynomial features**: Captures non-linear relationships
- **Interaction terms**: Combines features to reveal dependencies
- **Date/time features**: Extracts day, month, weekday, etc.
- **Text features**: TF-IDF, word embeddings, sentiment scores
- **Image features**: Edge detection, histogram of gradients (HOG), CNN embeddings

---

🧮 4. **Dimensionality Reduction**

Reduces feature space while retaining information.

- **PCA (Principal Component Analysis)**: Projects data into lower dimensions
- **t-SNE / UMAP**: For visualization of high-dimensional data
- **Feature selection**:
    - Filter methods (e.g., correlation threshold)
    - Wrapper methods (e.g., RFE)
    - Embedded methods (e.g., Lasso, tree importance)

⚙️ 5. **Data Integration & Aggregation**

Combines multiple sources or levels of data.

- **Joining datasets**: SQL joins, pandas `merge()`
- **Group-wise aggregation**: `groupby()` in pandas, window functions in SQL
- **Pivoting/unpivoting**: Reshaping data for analysis

📈 6. **Pipeline Automation**

Ensures reproducibility and scalability.

- **Scikit-learn Pipelines**: Chain preprocessing and modeling steps
- **Custom transformers**: For reusable logic
- **Airflow / Prefect / Dagster**: For orchestrating ETL workflows
- **Docker + CI/CD**: For deploying data pipelines in production

---

⚖️ What Is Unbalanced Data?

It occurs when one class significantly outnumbers the other(s). For example, in a binary classification:

- Class 0: 95%
- Class 1: 5%

A naive model might predict only Class 0 and still achieve 95% accuracy—while completely missing the minority class.

🛠 Techniques to Handle Unbalanced Data

1. Resampling Methods

- **Oversampling**: Duplicate or synthetically generate minority class samples (e.g., SMOTE, ADASYN)
- **Undersampling**: Remove samples from the majority class
- **Hybrid**: Combine both for balance

2. Algorithmic Adjustments

- **Class weighting**: Assign higher penalty to misclassifying minority class (e.g., `class_weight='balanced'` in scikit-learn)
- **Cost-sensitive learning**: Customize loss functions to penalize errors on minority class more

3. Evaluation Metrics

Accuracy is misleading—use:

- **Precision, Recall, F1-score**
- **ROC-AUC** and **PR-AUC**
- **Confusion matrix**: Visualize true/false positives/negatives

4. Ensemble Methods

- **Boosting algorithms** (e.g., XGBoost, LightGBM) often handle imbalance better
- Use `scale_pos_weight` or similar parameters to adjust for imbalance

5. Data Augmentation (for CV/NLP)

- For images: rotation, flipping, cropping
- For text: synonym replacement, back translation

6. Anomaly Detection Framing

- Treat minority class as anomaly and use unsupervised methods like Isolation Forest or One-Class SVM

🧠 Interview Tip

If asked how you'd handle imbalance in a real-world project, walk through:

1. **Exploratory analysis** to detect imbalance

2. **Metric selection** (e.g., F1-score over accuracy)
3. **Resampling or weighting strategy**
4. **Model tuning and validation**

🧮 What Is a Confusion Matrix?

It's a **2x2 table** (for binary classification) that compares the **actual labels** with the **predicted labels**. It helps you see not just how often your model is correct, but *how* it makes mistakes.

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

🔍 Definitions

- **True Positive (TP)**: Model correctly predicts positive class.
- **True Negative (TN)**: Model correctly predicts negative class.
- **False Positive (FP)**: Model incorrectly predicts positive (Type I error).
- **False Negative (FN)**: Model incorrectly predicts negative (Type II error).

📊 Why It Matters

From the confusion matrix, you can derive key metrics:

Metric	**Formula**	**Use Case**
Accuracy	((TP + TN) / (TP + TN + FP + FN))	Overall correctness
Precision	(TP / (TP + FP))	When false positives are costly (e.g., spam detection)
Recall	(TP / (TP + FN))	When false negatives are costly (e.g., disease detection)
F1-score	(2 \cdot (Precision \cdot Recall) / (Precision + Recall))	Balance between precision and recall

---

🧠 Interview Tip

If asked to interpret a confusion matrix:

- Start by explaining the layout and what each cell means.
- Then derive precision, recall, and F1-score.
- Finally, relate it to the business context (e.g., "In fraud detection, recall is more critical than precision").

--

Evaluation Metrices:

📊 1. **Accuracy**

Definition: Proportion of correct predictions over total predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Use Case:

- Works well when classes are balanced.
- Misleading in imbalanced datasets (e.g., 95% accuracy when minority class is ignored).

🎯 2. **Precision**

Definition: Of all predicted positives, how many are truly positive?

$$\text{Precision} = \frac{TP}{TP + FP}$$

Use Case:

- When **false positives** are costly (e.g., spam detection, cancer diagnosis).
- High precision = fewer false alarms.

🧠 3. **Recall (Sensitivity)**

Definition: Of all actual positives, how many did we correctly identify?

$$\text{Recall} = \frac{TP}{TP + FN}$$

Use Case:

- When **false negatives** are costly (e.g., fraud detection, disease screening).
- High recall = fewer missed cases.

⚖️ 4. **F1-Score**

Definition: Harmonic mean of precision and recall.

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Use Case:

- When you need a balance between precision and recall.
- Especially useful in imbalanced datasets.

📈 5. **ROC-AUC (Receiver Operating Characteristic – Area Under Curve)**

- **ROC Curve**: Plots **True Positive Rate (Recall)** vs. **False Positive Rate** across thresholds.
- **AUC**: Measures the area under the ROC curve.

$$\text{AUC} = \text{Probability that a randomly chosen positive is ranked higher than a negative}$$

Use Case:

- Great for comparing models.
- Threshold-independent.
- Works well with imbalanced data.

🧠 Quick Decision Guide

Metric	Best For	Avoid When...
Accuracy	Balanced datasets	Classes are highly imbalanced
Precision	False positives are costly	You care more about recall
Recall	False negatives are costly	You care more about precision
F1-Score	Balanced precision & recall	You need threshold-independent metric
ROC-AUC	Model comparison, threshold tuning	You need interpretability at fixed threshold

---