ML Coding: Supervised Cheatsheet

1. Intro to ML Engineering

Definition: The discipline of taking ML models from research/experimentation to production.

- **The ML Lifecycle:**
  1. **Problem Definition:** Business KPI → ML Metric.
  2. **Data Engineering:** Collection, Cleaning, Labeling.
  3. **Modeling:** Feature Engineering, Training, Tuning.
  4. **Deployment (MLOps):** Serving (API/Batch), Monitoring (Drift).

---

2. Linear Regression (I, II, III)

Part I: The Basics

Used for predicting a continuous value (Regression).

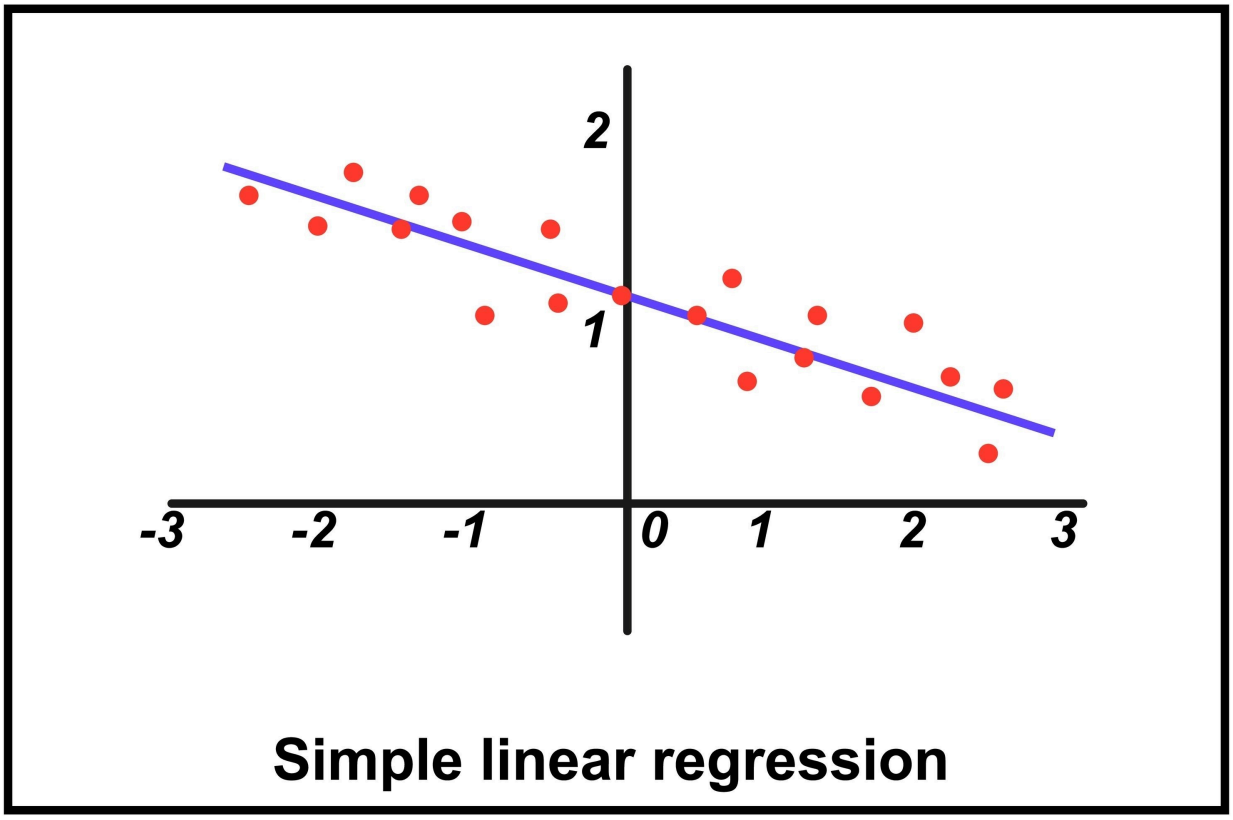
- **Equation:**

  $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon$

- **Objective:** Find the "Line of Best Fit" that minimizes error.
- **Cost Function (MSE):** Mean Squared Error. The algorithm tries to minimize this.

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- **Optimization: Gradient Descent** is used to update weights (β) iteratively to reach the global minimum.



Part II: The Assumptions (Crucial for Interviews)

If these are violated, your model is unreliable.

1. **Linearity:** The relationship between X and y is linear.
2. **Independence:** Observations are independent of each other (no autocorrelation).
3. **Homoscedasticity:** The variance of residuals is constant across all levels of X.
4. **Normality of Residuals:** The error terms are normally distributed.
5. **No Multicollinearity:** Independent variables should not be highly correlated with each other (Check using VIF - Variance Inflation Factor).

Part III: Regularization (Handling Overfitting)

When the model learns the "noise" in the training data (High Variance).

- **Ridge Regression (L2 Penalty):** Adds *squared* magnitude of coefficients to the loss function.
    - *Effect:* Shrinks coefficients towards zero but not *to* zero. Good for multicollinearity.
- **Lasso Regression (L1 Penalty):** Adds *absolute* value of coefficients to loss.
    - *Effect:* Can shrink coefficients to exactly zero. **Performs feature selection.**
- **Elastic Net:** A combination of L1 and L2.

---

3. Sklearn Pipelines

Purpose: Chains preprocessing and modeling steps into a single object. Prevents **Data Leakage** (statistics from test set leaking into training).

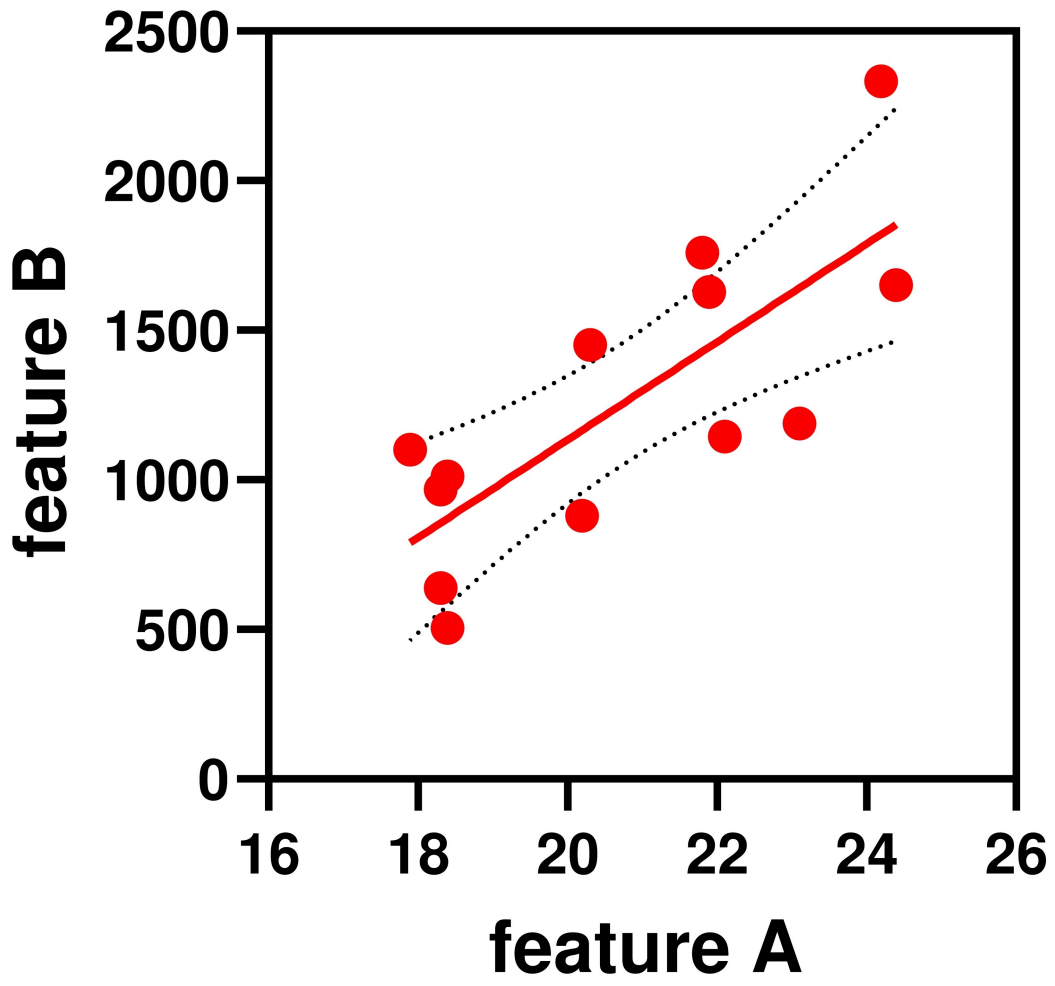
```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

pipe = Pipeline([
    ('scaler', StandardScaler()),      # Step 1: Scale data
    ('model', LogisticRegression())    # Step 2: Classifier
])

# pipe.fit(X_train, y_train)
# pipe.predict(X_test) → Automatically scales X_test using X_train stats
```

---

4. Logistic Regression

Despite the name, this is a **Classification** algorithm (Binary).

- **Core Idea:** Instead of fitting a straight line, it fits an "S" shaped curve (Sigmoid).
- **Sigmoid Function:** Maps any real value into a probability between 0 and 1.
  $\sigma(z) = \frac{1}{1+e^{-z}}$
- **Decision Boundary:** Usually 0.5. If $P(y=1) > 0.5$, predict Class 1.
- **Loss Function:** Log Loss (Binary Cross Entropy).



5. Classification Metrics

How to evaluate if your classifier is good.

Metric	Formula	When to use?
Accuracy	$\frac{TP+TN}{\text{Total}}$	Only when classes are balanced.
Precision	$\frac{TP}{TP+FP}$	When **False Positives** are costly (e.g., Spam detection).
Recall	$\frac{TP}{TP+FN}$	When **False Negatives** are dangerous (e.g., Cancer detection).

F1-Score	$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$	Harmonic mean. Best balance for uneven classes.

Export to Sheets

- **Confusion Matrix:** A table showing TP (True Positive), TN, FP, FN.
- **ROC-AUC Curve:** Plots TPR (Recall) vs. FPR. AUC = 1 is perfect, 0.5 is random guessing.

---

6. K-Nearest Neighbors (KNN)

- **Type:** Supervised, Lazy Learner (doesn't "train," just memorizes data).
- **Logic:** "Tell me who your neighbors are, and I'll tell you who you are."
- **Distance Metrics:**
  - *Euclidean:* Straight line distance.
  - *Manhattan:* Grid-like (city block) distance.
- **Hyperparameter K:**
  - *Low K (e.g., K=1):* High Variance (Overfitting), sensitive to noise.
  - *High K:* High Bias (Underfitting), smooths out boundaries too much.
- **Note:** Requires **Feature Scaling** because it relies on distance calculation.

---

7. Naive Bayes

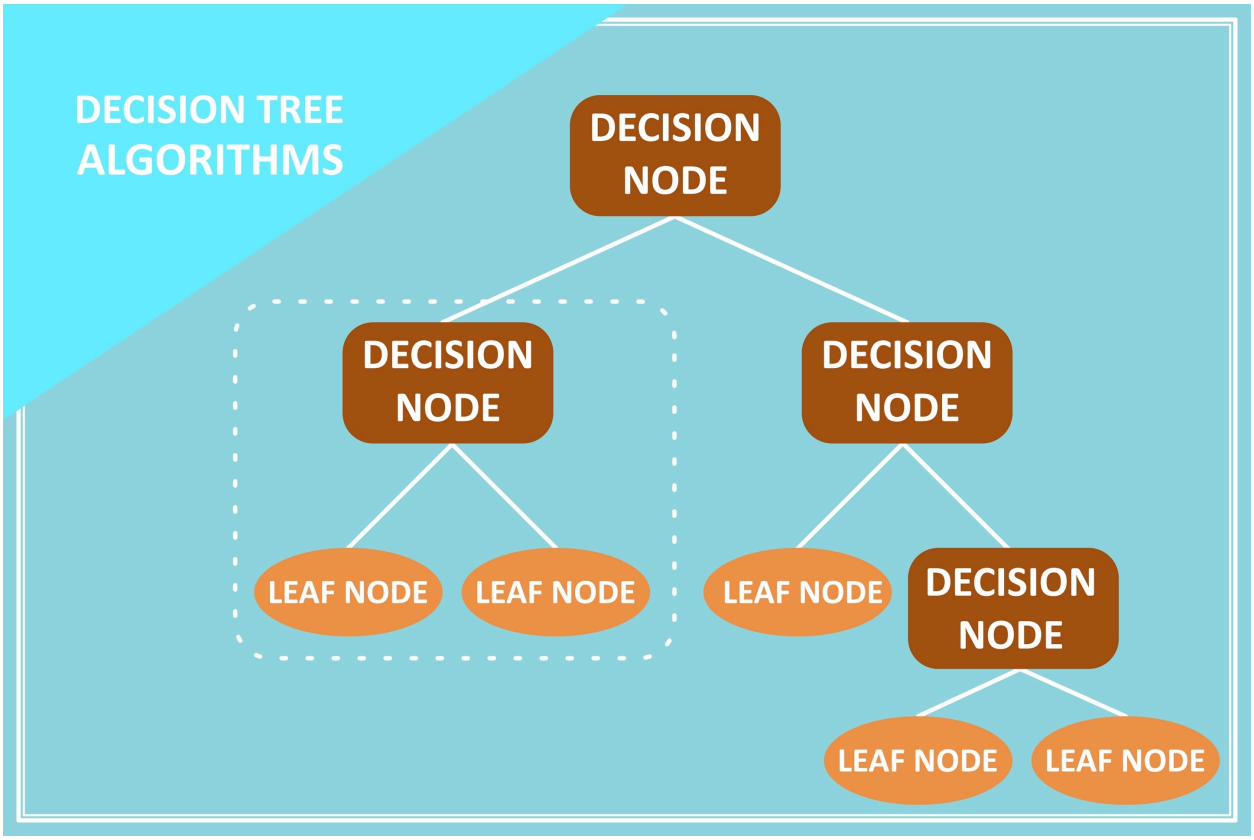
- **Type:** Probabilistic Classifier based on **Bayes' Theorem**.

  $$P(A \mid B) = P(B)P(B \mid A) \cdot P(A)$$

- **The "Naive" Assumption:** Assumes all features are **independent** of each other (rarely true in real life, but the model still performs well).
- **Variants:**
  - *Gaussian NB:* For continuous data (assumes normal distribution).
  - *Multinomial NB:* For discrete counts (e.g., Text classification/Word counts).
  - *Bernoulli NB:* For binary/boolean features.

8. Decision Trees

- **Logic:** Splits data into subsets based on asking a series of questions (If X>5, go left; else right).
- **Splitting Criteria (How it chooses the question):**
  - **Gini Impurity:** Measures "impurity" (0 = all elements belong to same class). Preferred for speed.
  - **Entropy (Information Gain):** Measures disorder. High entropy = high disorder.
- **Pros:** Easy to interpret/visualize, no scaling required.
- **Cons:** Extremely prone to **Overfitting** (memorizing the tree).
- **Solution:** Pruning (cutting branches) or using Ensembles.



9. Ensemble Models

Intro to Ensembling

Combining multiple "weak learners" (usually simple decision trees) to create a "strong learner."

A. Bagging (Bootstrap Aggregating)

- **Mechanism:** Train multiple models in **parallel** on different random subsets of data (with replacement).
- **Goal:** Reduce **Variance** (Overfitting).
- **Star Player: Random Forest.**
  - Constructs many decision trees.
  - Output is the majority vote (classification) or average (regression) of all trees.

B. Boosting

- **Mechanism:** Train models **sequentially**. Each new model tries to correct the errors made by the previous model.
- **Goal:** Reduce **Bias** (Underfitting).
- **Star Players:**
  - **AdaBoost:** Assigns higher weights to misclassified points.
  - **Gradient Boosting (GBM):** Optimizes the loss function using gradient descent.
  - **XGBoost / LightGBM:** Optimized, high-performance versions of GBM (Standard for competitive ML).