Machine Learning Task

Smart Targeting for Bank Term Deposits

Instructions for the candidate: Choose **either 1 or 2** and complete it in **30 minutes**. You may use Python/R. Keep it simple and readable.

Dataset: Kaggle — Bank Marketing (code tab):
https://www.kaggle.com/datasets/henriqueyamahata/bank-marketing/data

---

Option 1 — Single-Campaign Ranking (hands-on, coding)

Goal. Train a quick binary model that ranks customers for a term-deposit campaign.
Timebox. 30 minutes. **Lang.** Python or R. **Keep it simple.**

Data + Split

- Load the Kaggle Bank Marketing dataset (link above).
- **Exclude leakage:** drop `duration` from features.
- Train/test split: stratified by target `y`, with a fixed random seed (e.g., 42).

Minimal Preprocessing

- One-hot encode categoricals; keep numerics as is (standardize only if your model needs it).
- Optional: handle rare categories by “other” or let one-hot handle naturally.

Model

- Train one fast classifier (e.g., Logistic Regression with class_weight="balanced", or a small tree/GBM).
- Use the train set only (no peeking).

Scoring + Metrics

- Score the **test** set → predicted probability `p = P(y=1)`.
- Compute:
    - **ROC-AUC** (and/or PR-AUC; ROC-AUC required).
    - **Top-k precision** where `k = round(0.10 * test_size)`.
- Build a **Top-15 table** sorted by `p` desc with columns: `row_id (or customer_id), predicted_prob`.

What to Print (exact)

Print these lines (replace ALL CAPS with your values):

- `AUC_ROC: XX.XXX`
- `TopK_Precision@10%: XX.XXX`

Then display:

- **Top 15** table: `row_id, predicted_prob` (rounded to 3–4 decimals).

Short Notes (final cell, 3–5 lines total)

- **Which features mattered:** list 2–3 (e.g., top coefficients or importances).
- **How we'd deploy daily:** batch scoring on latest data, rank by `p`, contact top N given capacity; monitor AUC, conversion rate, and drift.

Option 2 — Multi-Campaign & Channel Policy (hands-on, coding)

Goal. Learn a policy to pick `(campaign_intensity, channel)` per customer to maximize conversions.
Timebox. 30 minutes. **Lang.** Python or R. **Keep it fast.**
Tip. Don't use `duration` (leakage).

Setup

- Load data; stratified train/test split on `y`, fixed seed.
- Features = pre-contact info only (exclude `duration`). One-hot categoricals.
- Define **actions** from data:
    - `campaign_intensity ∈ {1, 2+}` derived from `campaign`.
    - `channel ∈ {cellular, telephone}` from `contact`.
    - Actions: **A1** (1, cellular), **A2** (1, telephone), **A3** (2+, cellular), **A4** (2+, telephone).
- Reward: `y == "yes"` → 1 else 0.

Modeling (choose one)

- **Per-action models:** train 4 binary models, each on rows where that action occurred; at inference, score all 4 and pick argmax `P(y=1 | action)`.
  OR
- **Shared policy:** train a multiclass policy to choose the action; also train a single binary outcome model for `P(y=1)` to report probabilities/ties.

Offline Eval (implement end-to-end)

- **Diagnostics:** ROC-AUC per binary model (or for your outcome model). Print class balance and note if you used class weights.

- **Policy value (action-match):** apply policy to test, keep rows where chosen action equals logged action, compute `conversions / matched_rows`. Also compute two baselines: random action (4-way) and a fixed policy (e.g., always (1, cellular)).

- **Top-k lift:** define a **policy score** per user (e.g., max predicted `P(y=1|a)`). With `k = round(0.10 * test_size)`, compute Top-k precision for policy vs random; report **lift = policy / random**.

- **Recommendation table (10 rows):** columns = `row_id, chosen_campaign_intensity, chosen_channel, policy_score, reason`. Show top 10 by `policy_score`.

- **Feature importance:** quick print (coefficients or importances).

What to Print (exact)

- `AUC_per_action: {A1: XX.XXX, A2: XX.XXX, A3: XX.XXX, A4: XX.XXX}`
- `PolicyValue_Match: XX.XXX`
- `Baseline_Random_Match: XX.XXX`
- `Baseline_Fixed_Match: XX.XXX`
- `TopK_Precision (policy): XX.XXX`
- `TopK_Precision (random): XX.XXX`
- `Lift: XX.XXX`

Then display:

- The **10-row** recommendation table.
- A short top-features printout (per model or overall).