Real-Estate Professionals Scraper + ML + Deep Learning

1. Overview

This project collects **LinkedIn profiles of real estate professionals in India** using Selenium-based scraping, then applies **two machine-learning pipelines** to automatically detect whether a profile is **relevant to the real estate domain**:

1. **Classical ML Baseline** – TF-IDF vectoriser + Logistic Regression.
2. **Deep-Learning Model** – Keras **Embedding + Bi-LSTM** network.

The pipeline produces:

- **Raw scraped CSV** of all profiles.
- **Scored CSV** with ML & DL probabilities and predicted relevance labels.
- **Filtered CSV** with only high-confidence “relevant” profiles.
- **Uncertain CSV** with mid-confidence profiles for manual review.
- A **ZIP archive** for easy sharing.
- Saved **vectoriser / ML model / DL model/tokenizer** for later reuse.

The code is designed to **recover gracefully from LinkedIn session drops** and to **handle very small datasets**.

---

2. Usage

1. **Configure credentials**
   Edit these two variables near the top of the script:
2. LINKEDIN_EMAIL = "your_email@example.com"
3. LINKEDIN_PASSWORD = "your_password"
4. **Run the script**
5. python real_estate_scraper_ml_dl.py
6. **Outputs** (all saved in the working directory):
   - real_estate_professionals_india_scored.csv – all profiles with ML + DL scores
   - real_estate_professionals_india_filtered.csv – only high-probability relevant profiles
   - real_estate_professionals_india_uncertain.csv – medium-confidence profiles
   - real_estate_professionals_india.csv – identical to *filtered.csv* (legacy name)
   - real_estate_professionals_india.zip – zipped filtered CSV
   - models/ folder with:
     - relevance_vectorizer.joblib
     - relevance_model.joblib
     - dl_tokenizer.joblib
     - dl_model.h5
7. **Typical workflow**

- The script first generates random (company, city, job title) queries.
- Launches Chrome, logs into LinkedIn.
- Scrapes profiles per query, deduplicates them.
- Builds weak labels using keyword-based rules.
- **Trains or loads** both the baseline and DL models.
- Scores every profile with **probabilities** and **predicted labels**.
- Saves final CSVs and ZIP for distribution.

---

3. Rationale for Model Choices

3.1 Weak-Label + ML Approach

- Scraped data is **unlabelled**.
- A **rule-based weak-labelling** step (keywords such as “broker”, “property manager”) provides **noisy binary labels** to bootstrap training.

3.2 TF-IDF + Logistic Regression (Baseline)

- **Interpretable & light-weight**: good for small datasets and explainable terms.
- **TF-IDF** captures important words & bigrams in titles/positions.
- **Logistic Regression** is **fast to train**, robust with sparse data, and supports **class-imbalance weighting**.

3.3 Deep-Learning Bi-LSTM

- For larger datasets, **word order and context** can be informative.
- The **Embedding + Bidirectional LSTM** network captures **sequential patterns** that TF-IDF ignores.
- Uses **Early-Stopping** and neutral fall-backs to avoid over-fitting or failures on small data.

3.4 Hybrid Output

- Both models’ **probabilities** are retained:

  prob_relevant – baseline TF-IDF + LogReg probability.

  dl_prob_relevant – Deep-Learning probability.

- Users can compare or ensemble them later.

---

4. Robustness & Safety Features

- **Validation-split auto-adjustment:** skips validation if <10 samples.
- **Dummy-model fall-back:** if dataset <2 samples, DL model is skipped and outputs a neutral probability (0.5) for each profile.
- **Shape-mismatch guard:** fixes the Keras output-length bug encountered with tiny datasets.
- **Auto-restart of the Selenium driver** if the LinkedIn session expires.
- **Deduplication** to avoid repeated profiles.