ChatGPT

Predicting Early-Stage Startup Success with an RNN

Project Goal: Build a prototype ML pipeline to predict whether a U.S. tech startup (Seed–Series A stage) will **"succeed"** (achieve a Series B round or be acquired) using an RNN or similar time-series model. The plan covers data sources, engineering, modeling, evaluation, and a Colab/Kaggle demo.

1. Data Sources

We will rely on public datasets about startups and funding. Key candidates include Crunchbase and Kaggle compilations:

- **Crunchbase (Daily CSV Export):** Crunchbase provides comprehensive data on companies, funding rounds, acquisitions, etc. It offers daily CSV dumps (organizations.csv, funding_rounds.csv, acquisitions.csv, etc.) covering millions of startups [1]. We can use the *daily CSV export* (or Kaggle mirror) and filter for U.S.-based, tech-sector startups founded ≥ 2000 [2].
- **Kaggle – Startup Investments (Crunchbase snapshot):** An existing Kaggle dataset ("Startup Investments") contains Crunchbase tables (organizations, funding_rounds, acquisitions, IPOs) [3]. This offers convenient access to Crunchbase-like data (organization profiles, rounds, exits).
- **Kaggle – Startup Success Datasets:** Several Kaggle datasets directly target startup success (e.g. "Startup Success Prediction" by Manishkc06, "Global Startup Success" by HamnakaleemDS, "Startup Success/Fail from Crunchbase" by Yanmaksi). These typically include features like funding amounts, industry, founding date, etc., labeled by an outcome (often Series B or exit). (We will cross-check these sources for usable data tables, or at least for validation and feature ideas.)
- **Kaggle – US Startup Data:** Datasets focusing on U.S. startups (e.g. timeseries of US startup counts, industry breakdowns) may provide additional features (such as ecosystem growth trends or city-level info). Even AngelList or Crunchbase profiles for individual founders could be used if easily scraped.
- **Other Open Data:** Dealroom and PitchBook are popular but generally not openly available without subscription. We prioritize Crunchbase-based and Kaggle datasets for feasibility (per constraints). We will filter *any* global dataset to only U.S. companies, early-stage, tech-related (e.g. Software, Internet, AI, FinTech, etc.) [2].

The table below summarizes candidate datasets:

Dataset / Source	Contents (Approx.)	Key Fields / Notes	Link / Access
Crunchbase Daily CSV Export	Organization profiles, funding rounds, acquisitions, IPOs, etc. (millions of records) [1] [2]	*organizations*: `company_name`, `founded_date`, `country`, `city`, `industry`; *funding_rounds*: `company_id`, `round_type`, `raised_amount`, `announced_on`; *acquisitions*: `company_id`, `acquisition_date`, *successor_company*; etc. Filter by country_code=US, tech categories [2].	Crunchbase Data Export (requires signup)
Kaggle: Startup Investments (Justinas)	Crunchbase-derived snapshot tables (~5 CSVs) including organizations, funding_rounds, acquisitions [3].	Similar schema to Crunchbase above. Includes news and investor data.	Kaggle Dataset
Kaggle: Crunchbase Startups (Chhinna)	Crunchbase companies and related data snapshot.	Organization profiles (with founding date, category, location), plus funding rounds tables.	Kaggle Dataset
Kaggle: Startup Success/Fail (Yanmaksi)	Aggregated startup stats for success analysis.	Features like funding totals, R&D, founding team size; outcome label (success/fail).	Kaggle Dataset
Kaggle: Global Startup Success (Kaleem)	~5,000 startups (10 countries) with 15 features.	Fields include country, industry, founding year, funding rounds, Series A/B flag, acquisition flag. Can filter to US.	Kaggle Dataset
Kaggle: USA Startup Timeseries (thedevastator)	US startup counts by year, state (2008–?), sector.	Useful for contextual features (e.g. regional trends).	Kaggle Dataset
AngelList / Other	(Optional) AngelList data or curated smaller sets.	e.g. AngelList AI startups (~500 rows). Quick win features: YC involvement flag, etc.	Kaggle AngelList AI startups

In summary, Crunchbase (via its CSV export or Kaggle mirrors) is our primary data source, as it comprehensively lists funding rounds and exits [1] [2]. Kaggle provides convenient packaged snapshots of this data and some labeled startup success datasets. All sources will be filtered to U.S. early-stage tech companies.

2. Data Engineering & Feature Pipeline

Data Ingestion: Download/obtain the Crunchbase tables (organizations, funding_rounds, acquisitions, etc.). In Kaggle/Colab, this may involve using the Kaggle API to fetch datasets or uploading CSVs to Google Drive. Alternatively, a publicly-shared Crunchbase export (sample or partial) can be loaded if available. Ensure data is limited to U.S.-based companies (country_code = "USA") and industries like Software, Internet, AI, FinTech, etc., as in [52] ("specific categories" of tech) [2]. Filter founding dates (e.g. ≥2000) to focus on modern startups.

Labeling (Dependent Variable): Define the binary target as *success = 1 if the company eventually raised a Series B (or higher) or was acquired, else 0.* This matches the literature: e.g., Kim & Shin (2024) label any company "currently operating and attracting Series B or higher" as successful [4]. (We consider an acquisition as analogous to a positive outcome.) We will need to derive this label by checking each startup's funding rounds: if any `round_type` = "series_b" (or "series_c+"), or if it appears in the acquisitions table (as a target), label = 1. All others (no B, no M&A) are label = 0.

Feature Construction: We will engineer both *static* and *time-series* features:
- *Static features (company-level):* e.g. `founding_year`, `city/state`, `industry_category`, `number_of_founders`, `lead_investor_count`, etc. We may encode location or industry as categorical embeddings. Draw on founder data if available: e.g. founder education level, major, work experience. Prior work highlights founder attributes as important predictors [5]. For example, include `founders' education level`, `founders' major field` (if in data), and `work_experience_years` as numerical or categorical features [5]. If Crunchbase has social media or link fields (Twitter/LinkedIn flags), we can use those as binary indicators. Company sector (e.g. "AI", "Mobile", "FinTech") can be one-hot or embedding. - *Time-series features (sequential events):* Represent each startup as an ordered sequence of its funding-related events. For instance, at each time step *t* we could include: `(round_type_t, amount_raised_t, time_since_prev_round, num_investors_t)` etc. Round types are ordinal (seed=1, A=2, B=3, …) which we can embed or one-hot. Log-transform funding amounts. Compute **cumulative** features by round, e.g. total funding to date, cumulative number of rounds, current valuation if available. (These aggregated features can be fed as additional inputs or as features at each time step.)

Sequence Formatting: We'll sort each startup's events chronologically (founding date as t0, then seed, Series A, etc.). For RNN input, we'll pad or truncate sequences to a fixed length (e.g. max 5–10 events) with masking. Static features (industry, location, founder count, etc.) can be concatenated to each time step's input vector or injected via an initial hidden state.

Data Cleaning: Handle missing values (e.g. unknown funding amounts) by imputation (e.g. zero or median). Drop companies with no funding data beyond seed if insufficient info. Remove duplicates or subsidiaries (as done in prior work) [6]. Scale numeric features (e.g. StandardScaler on log-amounts, ages). Split data by company (not by event) to avoid leakage. Use a cutoff date (e.g. include only companies founded before 2019) so we have enough follow-up to observe eventual Series B/acquisition.

Class Imbalance: Expect <15% successes (e.g. Kim & Shin found ~13.7% labeled "success" [7]). We will address imbalance by weighting the loss or oversampling. Possibilities: SMOTE or GAN-based oversampling of the minority class [8]. (Kim & Shin successfully used a GAN to augment rare successes [9].) Alternatively, use class weighting in the loss (e.g. PyTorch's `pos_weight`).

Feature Pipeline Summary (example steps):
1. **Load data:** Read organizations, funding_rounds, acquisitions CSVs into DataFrames.
2. **Filter & join:** Select US tech startups; join funding_rounds by company; label "success" if Series B/A etc.
3. **Engineer static features:** e.g. `age = current_year - founding_year`, `sector = industry category`, founder count, etc. Encode categoricals.
4. **Build time sequences:** For each company, sort rounds by date; extract per-round features (type, amount, investors). Compute deltas (time gaps) and cumulative sums. Pad sequences to fixed length.
5. **Train/Test split:** Use time-based split (e.g. companies founded $\leq$2018 for train, >2018 for test) or stratified K-fold. Ensure no leakage of future info.

3. Modeling Strategy (RNN and Alternatives)

Primary Model – RNN (LSTM/GRU): We will use a recurrent neural network (e.g. LSTM or GRU) to process the sequence of funding events for each startup and output a binary prediction. The architecture might be: embedding/linear layers → LSTM layer(s) → dropout → dense layer → sigmoid output. Static features (industry, location, founder info) can be concatenated to the RNN output or fed into an auxiliary dense network whose outputs are merged. The RNN will learn temporal patterns (e.g. accelerating funding, large early rounds, frequent investments) that signal future success. Training will use binary cross-entropy loss. A representative design: a 1–2 layer BiLSTM (hidden size ~64–128) with dropout (0.2–0.5) and L2 weight decay, trained on GPU.

Alternatives and Model Comparison: We will compare the RNN approach with other strategies:

Model / Approach	Type	Strengths	Challenges
RNN (LSTM/GRU)	Sequence model (DL)	Captures temporal dependencies in funding events. Handles variable-length sequences. Known success in sequential financial data. [10]	Requires careful sequence design. May overfit on small data.
Transformer / Attention	Sequence model (DL)	Learns long-range dependencies without recurrence; can use attention over funding events. Handles multi-step prediction.	Heavier compute, overkill for short sequences, needs more data.
Temporal CNN (TCN)	Sequence model (DL)	Captures sequence patterns with convolutions; often faster to train.	Less common for non-regular time steps; may need manual feature engineering of intervals.
Survival Analysis (Cox PH, DeepSurv)	Time-to-event model	Models time until "failure" (here, Series B/exit). Naturally handles censored data. Gives hazard ratios (interpretability).	Assumes proportional hazards (Cox PH). May need custom deep-learning survival models for non-linear effects.

Model / Approach	Type	Strengths	Challenges
Baseline ML (XGBoost/ LogReg)	Static classifier (ML)	Simple to implement; interpretable. Good baseline using aggregate features (total funding, #rounds, static features).	Ignores event order (unless features are pre-aggregated); often outperformed by DL on sequences.

We will prototype the RNN first, then evaluate whether a transformer or a time-interval model adds value. Traditional models (e.g. Random Forest or logistic regression on engineered features) will serve as baselines. In related work, Potanin et al. (2023) and others have achieved ~0.80–0.86 ROC-AUC on similar startup success tasks using deep models [10], suggesting a well-tuned RNN could be competitive.

4. Evaluation Plan

We treat this as a binary classification. Key metrics include:

- **ROC-AUC:** Area under the ROC curve. (Prior work reports ~0.86 AUC [10] as achievable.)
- **Precision / Recall / F1:** Especially important if we prioritize finding *any* likely successes (recall) vs. minimizing false alarms (precision). If class imbalance is strong, report precision-recall AUC as well.
- **Confusion Matrix & Accuracy:** For completeness, though accuracy is less informative in imbalanced data.
- **Calibration:** Check predicted probabilities for well-calibration, since VC decisions may use probability thresholds.
- **Cross-validation:** Use time-split CV or rolling-origin validation to simulate forecasting. For example, train on companies founded $\leq$2017, test on 2018–2020 startups. This simulates "predicting new startups" and avoids peeking.

We will also analyze feature importance or attention weights (for LSTM with attention or transformer) to see which factors drive predictions. Given the imbalance (~10–15% positives [7]), we will tune class weights or oversampling to balance recall/precision. If an investor cares more about not missing a potential unicorn, we might emphasize recall (e.g. tune for high recall at a certain precision).

5. Deployment / Demo Plan

To showcase the MVP, we will implement the pipeline on Kaggle or Colab:

- **Notebook Setup:** Create a self-contained Jupyter notebook (or Kaggle Kernel) in Python, using common libraries (Pandas, NumPy, PyTorch or TensorFlow/Keras). We can store data in the notebook (if small) or fetch via Kaggle API/Direct URL. Kaggle provides free GPUs (Tesla K80/P100) which should suffice for an LSTM on a moderate dataset【42†】. Colab offers similar (with T4) and easily installs Kaggle API for data access.

- **Prototype Demo:** The notebook will demonstrate: data loading/preprocessing, model training (with progress outputs), and evaluation (plots of ROC curve, confusion matrix). For interactivity, we could include a widget or function where the user enters a hypothetical startup's key features (or toggles a

time-sequence slider) and the model outputs a success probability. (If time permits, a simple Gradio/Streamlit demo could be deployed.)

- **Presentation of Results:** Embed figures (training curves, ROC) and tables of metrics in the notebook. We will document each step so that a VC data science audience can follow the workflow.

- **Sharing:** The final deliverable can be a Kaggle Notebook (publicly shared) or a Colab link, along with this write-up. We will ensure that all data sources used have accessible links.

In summary, the project will ingest publicly available funding data (primarily Crunchbase), engineer startup-level sequences and features, train an LSTM-based classifier (with alternatives tested), and evaluate using standard metrics. A finished Kaggle/Colab notebook with sample results will demonstrate feasibility to a VC team. Throughout, we draw on best practices from recent studies [2] [4] [5] and aim for a reproducible, end-to-end MVP.

References: Public data and studies cited above (e.g. Crunchbase CSV schema [1]; industry analyses and ML papers [11] [12] [4] [5] [10]) inform our approach and validate that the chosen methods are grounded in current research.

---

[1] Daily CSV Export
https://data.crunchbase.com/v3.1/docs/daily-csv-export

[2] [6] [10] [11] [12] arxiv.org
https://arxiv.org/pdf/2309.15552

[3] GitHub - Gracekadiri/EDA-of-Startup-Ecosystem
https://github.com/Gracekadiri/EDA-of-Startup-Ecosystem

[4] [5] [7] [8] [9] Predicting startup success using two bias-free machine learning: resolving data imbalance using generative adversarial networks | Journal of Big Data | Full Text
https://journalofbigdata.springeropen.com/articles/10.1186/s40537-024-00993-8