**Advanced AI Forecasting with TensorFlow and Natural Language Processing**

**Week 5 - Instructor Notes — Nvidia Next‑Day Closing Meta‑Model**

These notes are designed for an instructor to speak confidently about cell in Nvidia\_Next\_Day\_Closing\_Meta\_Model\_Train\_Week5.ipynb. The notebook builds a stacked (meta) model that learns to blend next-day close price predictions from multiple base models trained on different lookback windows (365D, 270D, …, 1D). It then evaluates the meta‑model and saves both the trained estimator and the feature column order needed for inference.

**Quick Summary (what this notebook does)**

* Sets up the Colab environment and mounts Google Drive to read/write project files.
* Locates the latest prediction CSV from each lookback’s most recent training run.
* Loads those CSVs, renames each base model’s Predicted\_Close to Pred\_<LOOKBACK> (e.g., Pred\_365), and inner‑joins them by Date.
* Uses the corresponding Actual\_Close as the training target.
* Trains a Ridge regression meta‑model (L2‑regularized linear model) on the stacked predictions.
* Reports training metrics (R², MAE, MSE, RMSE) and saves the trained meta‑model and the feature column list with joblib.

**Teaching tip**: Emphasize that a meta‑model here is a simple, transparent way to *learn weights* for combining correlated predictors. Ridge helps when features (e.g., the base model predictions) are collinear.

**Files & Artifacts**

* **Inputs (read)**: Latest \*\_predictions.csv under each lookback folder inside /content/drive/My Drive/Nvidia\_Stock\_Market\_History/Training/ensemble\_inputs/<LOOKBACK>/<TIMESTAMP>/.
* **Outputs (write)** (to Google Drive):
  + Trained meta‑model: Meta\_Model\_Trained/meta\_model\_ridge.joblib
  + Feature order: Meta\_Model\_Trained/feature\_cols.joblib

Assumptions baked into the code: Folders are named by lookback (365D, 270D, …, 1D) and contain timestamped subfolders whose names end with YYYY-MM-DD\_HH-MM-SS. Each \*\_predictions.csv must include Date, Predicted\_Close, and Actual\_Close.

**Setup & package installs (Colab)**

**What it does:**

* Upgrades pip and installs/locks package versions: ipywidgets, numpy==2.0.2, tensorflow==2.18.0, pandas==2.2.2, matplotlib, seaborn, scikit-learn==1.6.1, tqdm, transformers==4.53.1, tokenizers, newsapi-python==0.2.7, requests, beautifulsoup4.

**Why it matters:**

* Colab VMs are ephemeral; pinning versions ensures reproducibility. Even though not every library is used in this notebook, they’re part of the broader project environment.

**Talking points:**

* Version pinning avoids “it worked yesterday” bugs.
* TensorFlow / Transformers aren’t used here directly, but may be needed in upstream notebooks that produced the CSVs.
* If you see import errors later, they are not a concern, as they are not dependencies used in this notebook.

**Pitfalls:**

* Internet / PyPI outages will break installs. Rerun the cell or switch the runtime.
* GPU/CPU wheels: package versions must match the runtime’s CUDA/CuDNN (not critical here, since we don’t train deep nets in this notebook).

**Imports**

**What it does:**

* Imports standard libs (os, glob), data tools (pandas, numpy), Ridge and metrics from sklearn, joblib for model persistence, and time for parsing folder timestamps.

**Why Ridge?**

* Base model predictions are highly collinear (all are price predictions). L2 regularization stabilizes the linear fit and prevents over-reliance on any single lookback.

**Key talking points:**

* **Ridge vs OLS**: Ridge shrinks coefficients and handles multicollinearity better.
  + **OLS (Ordinary Least Squares)** struggles with multicollinearity because highly correlated features make XTX nearly singular, causing unstable and excessively large coefficients that change drastically with small data variations.
  + **Ridge Regression** adds an L2​ penalty (λ∑β2j) to shrink coefficients and modifies XTX to XTX + λI, making it invertible, stabilizing solutions, and distributing weights more evenly across correlated features.
* Feature scaling is not accomplished, because all features are in the same unit ($USD)
  + Scaling wouldn’t change the optimal linear combination materially here.

**Mount Google Drive**

**What it does:**

* Uses google.colab.drive.mount('/content/drive', force\_remount=True) to attach Drive.

**Talking points:**

* force\_remount=True ensures a clean mount, avoiding stale paths.
* Requires user auth; the path /content/drive/My Drive/... becomes accessible.

**Pitfalls:**

* In non‑Colab environments, this cell will fail; skip it and adjust paths accordingly.

**Define meta‑model save folder**

**What it does:**

* Sets META\_MODEL\_SAVE\_FOLDER = "/content/drive/My Drive/Nvidia\_Stock\_Market\_History/Training/Meta\_Model\_Trained".

**Talking points:**

* Keeping trained artifacts in Drive makes them persist beyond the Colab session.
* Use a stable, predictable path since other notebooks/scripts may load these files later.

**Discover the *latest* prediction CSVs**

**What it does:**

* Sets base inputs folder: /Training/ensemble\_inputs.
* Defines lookbacks: LOOKBACK\_FOLDERS = ["365D","270D","180D","90D","60D","30D","14D","1D"] .
* For each lookback:
  1. Verifies the lookback folder exists.
  2. Lists timestamped subfolders and sorts descending by timestamp using time.strptime .
  3. Picks the most recent subfolder.
  4. Finds \*\_predictions.csv inside and appends the first match to csv\_files.

**Why this design:**

* The pipeline may generate multiple runs; always ingest the *latest* to keep the meta‑model current.

**Pitfalls & checks:**

* If naming conventions drift (e.g., timestamp format changes), sorting will break. Check the time.strptime format: "%Y-%m-%d\_%H-%M-%S".
* If a lookback is missing a CSV, a FileNotFoundError is raised by design (fail fast).

**Instructor prompts:**

* “Why inner‑join later?” → to use only dates that *all* models predicted on, ensuring feature alignment.

**Sanity print of discovered paths**

**What it does:**

* Prints every path in csv\_files so you can visually verify one per lookback and that the paths are recent/correct.

**Teaching tip:**

* If anything looks off (e.g., wrong timestamp or duplicate files), fix folder structure before proceeding.

**Load & merge predictions**

**What it does:**

* Iterates files in csv\_files:
  + Extracts the lookback from the filename token containing "Lookback".
  + Reads CSV, keeps Date and Predicted\_Close only.
  + Renames Predicted\_Close → Pred\_<LOOKBACK> (e.g., Pred\_365).
  + Appends to a list.
* Merges all DataFrames on Date with *inner joins*.
* Loads Actual\_Close from the first CSV and merges it on Date.
* Casts Date to datetime.
* Prints shape and head() for inspection.

**Why inner join?**

* Ensures each training row includes *all* base model predictions, so the feature vector is complete.

**Pitfalls:**

* If any lookback is missing dates, the inner‑join shrinks the dataset. Consider outer‑join + imputation if losing too many rows.
* Assumes all CSVs share an identical Actual\_Close for a given Date. This is true if they were generated from the same ground truth.

**Instructor prompts:**

* What’s the consequence of using outer instead? → more rows but requires handling missing predictions.

**Prepare training matrices**

**What it does:**

* Selects feature columns: any column starting with "Pred\_".
* Creates X = merged\_df[feature\_cols].values and y = merged\_df["Actual\_Close"].values.

**Talking points:**

* All features are in dollars; uniform scale reduces the need for standardization.
* These features are the base models’ opinions about price for the same date.

**Sanity checks:**

* Print feature\_cols to confirm expected order.

**Persist feature order**

**What it does:**

* Saves feature\_cols to feature\_cols.joblib under META\_MODEL\_SAVE\_FOLDER.

**Why it matters:**

* At inference time, you must pass columns to the meta‑model in the *same order* used during training. Persisting the list prevents subtle bugs.

**Pitfalls:**

* If you later change lookback set or rename features, regenerate this file alongside retraining.

**Train the Ridge meta‑model**

**What it does:**

* Initializes Ridge(alpha=1.0) and fits meta\_model.fit(X, y).

**Talking points:**

* alpha controls L2 strength
  + higher = stronger shrinkage.
  + Start with 1.0 and tune via Cross Validation — a model evaluation technique where the dataset is split into multiple train/test folds to measure performance and select the best hyperparameters (in this case, the best alpha value).
* With collinear features, Ridge generally outperforms plain linear regression.

**Possible extensions:**

* Add a time‑series aware validation scheme (rolling/expanding windows) to tune alpha.

**Evaluate on the training data**

**What it does:**

* Computes predictions and reports R², MAE, MSE, RMSE.
* R² – Coefficient of Determination: Proportion of variance in the target explained by the model’s predictions.
* MAE – Mean Absolute Error: Average of the absolute differences between predicted and actual values.
* MSE – Mean Squared Error: Average of the squared differences between predicted and actual values.
* RMSE – Root Mean Squared Error: Square root of MSE, giving the average prediction error in the original units of the target.

**Interpretation guide:**

* **R²** (coefficient of determination)
  + near 1.0 → strong fit (on training data).
  + Near 0 → the model explains little variance.
  + Can even be negative if the model performs worse than predicting the mean.
* **MAE** is average absolute dollar error; **RMSE** penalizes larger misses more than MAE.

**Important caveat:**

* Metrics are computed **on the training set**; they will look optimistic. Use a hold‑out or backtest to assess generalization.

**Instructor prompts:**

* Why might RMSE > MAE? → indicates some larger outliers exist.

**Persist the trained meta‑model**

**What it does:**

* Ensures the save folder exists, then writes meta\_model\_ridge.joblib via joblib.dump.

**Talking points:**

* joblib is efficient for scikit‑learn objects.
* Keep model and feature\_cols.joblib together; inference scripts must load both.

**Common failure modes:**

* Save path not found (Drive not mounted); rerun the mount cell or fix META\_MODEL\_SAVE\_FOLDER.

**Conceptual Background (for discussion)**

* **Stacking/meta‑learning:** Learn a weighted combination of base learners instead of hand-picked weights.
* **Multicollinearity:** Base predictions are similar; Ridge stabilizes coefficients and spreads weight across signals.
* **Why inner join on Date:** Ensures a fair comparison (each row has a full feature vector from all lookbacks).
* **Data leakage concerns:** Never use future information in features. Here, we use each lookback’s *predicted* next-day close generated from historical windows. The meta‑model sees only those predictions and the same day’s actual close.

**Troubleshooting & Edge Cases**

* **Missing files:** The discovery cell will raise errors if folders or CSVs are missing. This is intentional so you can fix inputs early.
* **Wrong timestamp format:** Update the time.strptime pattern to match the folder names.
* **Mismatched columns:** If CSVs lack Predicted\_Close or Actual\_Close, standardize the upstream exporters.
* **Too few rows after merge:** Consider switching to outer join and imputing missing predictions; document the trade-offs.
* **Overfitting:** Add a backtesting section with rolling splits; log metrics on unseen dates.

**Live Demo Checklist (optional)**

1. Show the mounted Drive and the lookback folders.
2. Run the discovery cell and read back the printed paths to confirm recency.
3. After merging, inspect merged\_df.shape and merged\_df.head(); verify columns Pred\_365, Pred\_270, …, Actual\_Close.
4. Print feature\_cols before training; open the saved feature\_cols.joblib to show it matches.
5. Train Ridge, display metrics, then confirm both feature\_cols.joblib and meta\_model\_ridge.joblib exist in Drive.

**What this notebook does not do (and where to extend)**

* Doesn’t perform cross-validation or a rolling backtest → add to avoid overfitting.
* Doesn’t log run artifacts/metrics to an experiment tracker.
* Doesn’t produce plots of residuals/error distribution → consider adding for diagnostic teaching.

**Appendix — Key constants & paths (for quick reference)**

A screenshot of a computer program

AI-generated content may be incorrect.