CSCI 567 Final Tasks:

| **File** | **Owner** | **Purpose / Core Steps** | **Expected Inputs → Outputs** | **Tangible Result** |
| --- | --- | --- | --- | --- |
| **data\_pipeline.ipynb** | **Vikyath** | 1. Pull raw CSVs from Kaggle API and Yahoo Finance.  2. Convert CPI (monthly) to daily.  3. Align dates, forward-fill gaps. | **Inputs:** Kaggle URLs, Yahoo tickers.**Outputs:** raw.csv (≈ 3,800 rows × 5 cols). | Clean, reproducible dataset snapshot saved to Drive + committed. |
| **features.ipynb** | **Vikyath** | 1. Read raw.csv.  2. Add %return, MA5, MA20, rolling σ, RSI, MACD.  3. Min-max scale all columns.  4. Slice into 30-day windows. | **Inputs:** raw.csv.**Outputs:** train.npy, val.npy, feature list JSON. | Numpy arrays (≈ 3,040 train / 760 val windows × 30 × 15). |
| **bi\_gru.py** | **Vikyath** | 1. Define 1-layer Bi-GRU (32 units) in Keras.  2. Train 10 epochs with early stop.  3. Save best weights. | **Inputs:** train.npy, val.npy.**Outputs:** pred\_bi\_gru.csv, bgru\_weights.h5. | RMSE ≈ 9–11 on val. |
| **hf\_transformer.py** | **Abhinav** | 1. Wrap TimeSeriesTransformer-Tiny (HF).  2. Freeze encoder; train regressor head 4 epochs.  3. Log to W&B. | **Inputs:** same Numpy arrays.**Outputs:** pred\_transformer.csv, wandb\_run.json. | RMSE ≈ 8–10; attention weights stored. |
| **attention\_viz.ipynb** | **Abhinav** | Turn raw attention matrices into a heat-map PNG; show top 10 influential lags. | wandb\_run.json, model weights. | attention\_heatmap.png for report slide. |
| **tcn.py** | **Rodrigo** | 1. Build 3-block causal TCN in PyTorch-Forecasting.  2. Tune filters {32, 64}.  3. Save predictions. | Numpy arrays. | pred\_tcn.csv, tcn\_state.pth, RMSE ≈ 9–11. |
| **tcn\_viz.ipynb** | **Rodrigo** | Visualise first-layer filter activations; identify which lag segments the TCN fires on. | tcn\_state.pth. | tcn\_filters.png. |
| **ensemble.py** | **Rodrigo** | 1. Read three pred\_\*.csv files + val.npy targets.  2. Compute mean, RMSE-weighted, ridge meta.  3. Output best fusion. | Individual preds + y-val. | ensemble\_pred.csv, metrics.json (expect RMSE ≈ 7–8, DA ≥ 60 %). |
| **demo.ipynb** | **Vikyath** | 1. Pull last 30 days fresh prices via yfinance.2. Run all three models (load weights).3. Apply fusion weights → print tomorrow’s forecast. | Internet, model weights. | End-to-end demo (< 2 min runtime on T4). |
| **mini\_report.docx** | **All** | 2-page write-up: intro, data, methods, results. Each owner drafts their section. | PDFs and PNGs generated above. | Final report PDF. |
| **slides.pptx** | **All** | 6-slide deck: background | data | models |

Each Person’s Tasks:

1. **Vikyath – Bi-GRU Pipeline & Demo**• Owns data\_pipeline.ipynb, features.ipynb, bi\_gru.py, and the final demo.ipynb.  
   • Uses the merged daily dataset (raw.csv) built from Yahoo Finance (GC=F, DX-Y.NYB, CL=F) and FRED CPI. Then codes ensemble.py to fuse all three models.  
   • Trains a one-layer Bidirectional GRU on 30-day windows, exports pred\_bi\_gru.csv, and wires every model into the end-to-end demo.  
   • Success cue: val-set RMSE ≈ 9–11 and a demo notebook that prints tomorrow’s price in under two minutes.
2. **Abhinav – Hugging Face Transformer Track**• Works in hf\_transformer.py and attention\_viz.ipynb; logs experiments to Weights & Biases.  
   • Feeds the same pre-scaled 30 × 15 tensors from train.npy / val.npy into a TimeSeriesTransformer-Tiny, freezing most layers for speed.  
   • Saves pred\_transformer.csv plus an attention heat-map PNG that highlights which lags and features drive predictions.  
   • Target: Transformer val RMSE ≈ 8–10 and clear attention visuals for the report slide.
3. **Rodrigo – Temporal Convolutional Network & Ensemble**• Builds tcn.py and tcn\_viz.ipynb.  
   • Consumes the identical numpy windows; tunes a 3-block causal TCN (kernel 3, dilations 1-2-4).  
   • Outputs pred\_tcn.csv, a filter-activation PNG, and ensemble\_pred.csv after combining models via mean, RMSE-weighted, and ridge meta-learner.  
   • Pass criterion: individual TCN RMSE ≈ 9–11 and ensemble RMSE ≤ 8 with ≥ 60 % directional accuracy.
4. **ALL: Shared Dataset & Preprocessing**• Single source-of-truth file raw.csv (~3,800 rows, 5 original columns) built once in data\_pipeline.ipynb.  
   • Feature notebook adds returns, moving averages, volatility bands—yielding 15 continuous features and saves train.npy / val.npy.  
   • Every model reads these exact arrays, ensuring no leakage and perfectly aligned validation targets.
5. **ALL: Results & Deliverables to Watch For**• Three prediction CSVs (pred\_bi\_gru.csv, pred\_transformer.csv, pred\_tcn.csv) of length 760 each.  
   • metrics.json showing ensemble RMSE below the best single model.  
   • Visual assets: attention\_heatmap.png, tcn\_filters.png, loss curves from W&B.  
   • Final artifacts: Colab demo notebook, 2-page mini-report, 6-slide deck—all pushed to GitHub by the 48-hour mark.

What each teammate must produce (AFTER .npy)

| **Person** | **Model notebook** | **Mandatory output file(s)** | **Purpose in pipeline** |
| --- | --- | --- | --- |
| **Vikyath** | bi\_gru.ipynb / bi\_gru.py | pred\_bi\_gru.csv – a **single-column CSV** containing 760 numbers (your Bi-GRU’s predictions for every sample in val.npy) | Feeds the ensemble; also proves your model is working. |
| **Abhinav** | hf\_transformer.ipynb | pred\_transformer.csv – same shape: 760 rows, 1 column (Transformer predictions in the exact order of val.npy) | Second stream for the ensemble; attention plots optional. |
| **Rodrigo** | tcn.ipynb | pred\_tcn.csv – 760×1 predictions from the TCN | Third stream for the ensemble. |
| **Rodrigo (extra)** | ensemble.py | ensemble\_pred.csv – the fused forecast (simple mean, RMSE-weighted, and/or ridge meta-learner) plus metrics.json with RMSE & directional accuracy | Generates the team’s final “best guess” and evaluation numbers. |

**Note:** model-weight files (.h5, .pth) are nice to keep, but only the **prediction CSVs** are required for fusion.

**Summary:** Abhinav and Rodrigo will create their own Google Collab file for their model and then train on these 4 .npy files.

1. **Each creates a separate Colab notebook** (e.g., hf\_transformer.ipynb, tcn.ipynb).
2. **Mount Drive** → np.load('/content/drive/MyDrive/train.npy'), val.npy, y\_val.npy, y\_train.npy.
3. **Build and train** their model on train.npy and y\_train.npy; monitor performance on val.npy/y\_val.npy.
4. **Save predictions** for the 760-sample validation set (e.g., pred\_transformer.csv, pred\_tcn.csv).
5. **Use scaler.pkl** later to scale new data and inverse-scale forecasts.

Those three .npy files (plus scaler.pkl) are the only shared inputs they need.