

# Stock Market Trading with LSTM Analysis

Franklin Doane  
dept. of Computer Science  
Tennessee Technological University  
Cookeville, United States  
fgdoane42@tntech.edu

## I. DATASET CREATION

## II. TRAINING

### A. Config

- 1) *Load*: Load config as json object with path from positional arg
- 2) *Unpack*: Unpack config keys into variables

### B. Load dataset

- 1) *Init dataset object*: The dataset object is initialized and given config. Then the load dataset method is called which does the following.
- 2) *Verify files*: Loads all csvs in dataset and verifies a minimum length. A legacy feature at this point.
- 3) *Split files into task groups*: List of valid files is split into groups to have each group loaded with threads.
- 4) *Open dataset thread pool*: Open thread pool to load files.
- 5) *Load csv files*: Open every file from directory listed in config. Load as pandas dataframe with date index column and put them all in a list.
- 6) *Calculate indicators*: Calculate all additional features listed in configs. Cut out any data at the front of the dataframes that is missing these features (e.g. for rolling avgs).
- 7) *Create date feature*: 1hot encode day of the week and add the new weekday features to all dataframes.
- 8) *Slice data into training examples*: Use rolling window to cut each dataframe into training example dataframes with n day history. All examples get added to a list and labels are made into a dataframe of their own.
- 9) *Normalize examples*: Each dataframe training example is normalized using norm function from config.
- 10) *Create tensor input*: Each dataframe is converted to numpy then a tensor with the datatype and device listed in the config. All labels are converted a single tensor as well.

### C. Load Model

- 1) *Dynamically load model class*: The Python file containing the model class is loaded by name using importlib.
- 2) *Call init*: Init custom model class. The custom model class is passed a name, starting LR, and input feature count from dataset. The model inherits from torch.nn.Module which is also init.
- 3) *Setup pytorch objects*: Model class init sets up LSTM object and Sequential object for forward passes, a default optimizer, a null loss func and an empty history.

- 4) *Set optimizer*: Call model set optim method to initialize and save optimizer to model class.

- 5) *Send to device*: Set model to device specified in config.

### D. Scheduler

- 1) *Init scheduler*: Pass optimizer from model. Give it kwargs from config

### E. Dataloader

- 1) *Check for test set*: Check's to see if the model save config contains test set indices. For loading a partially trained model. Indices get set after init if none are there. Pretty much a vestigial feature.
- 2) *Call init*: Init object and pass in dataset object (and test set indices if applicable)
- 3) *Pick test indices*: If not test set indices were passed, create list of indices and either random sample indices (legacy) or slice chunk off the front to create test set indices.
- 4) *Make test set*: Use indices to make a mask and remove all test examples from training index set

## III. EVALUATION