PyTorch Financial Data Analysis Workflow

This project consists of several key files:

1. config.py - Configuration and hyperparameters
2. data\_loader.py - Data preprocessing and PyTorch datasets
3. model.py - Neural network architectures (LSTM and Transformer)
4. train\_simple.py - Training loop and model optimization
5. predict.py - Making predictions with trained models

WORKFLOW:

1. First, install PyTorch: pip install torch torchvision
2. Test data loading: python data\_loader.py  
   Test models: python model.py
3. Train a model: python train\_simple.py
4. Make predictions: python predict.py

WHAT EACH FILE DOES:

data\_loader.py:

* Uses Financial data - in this case from yahoo finance
* Creates PyTorch datasets and data loaders
* Handles data normalization and train/validation splits
* Converts time series into sequences for neural networks

model.py:

* Defines neural network architectures
* LSTM: Good for time series, captures temporal dependencies
* Transformer: More advanced, better for long sequences
* You can experiment with both and see which works better

train.py:

* Trains your model on the financial data
* Includes early stopping to prevent overfitting
* Saves the best model and training history
* Plots training curves to visualize learning

predict.py:

* Loads your trained model  
  Makes predictions on new data
* Calculates metrics (accuracy, error rates)
* Can predict future stock movements

Custom\_loss.py:

Using custom loss instead of MSE (Standard) Loss. Directional loss penalizes the wrong direction more, which is good. Huberloss is good for outliers. FocalLoss is good for imbalance classes, and Quantile loss is good for confidence [intervals.](http://intervals.so) So we have a class for each, depending on the situation.

Enhanced\_features.py - Data Preprocessing step.

* + Raw price data → Technical indicators (RSI, MACD, Bollinger Bands)
  + Returns calculation → Multiple timeframes (1d, 3d, 5d, 10d)
  + Moving averages → Price relative to SMA (5, 10, 20 day)
  + Volatility measures → Rolling standard deviation
  + Cross-asset features → Correlations between stocks

Also does Feature Selection step. Also does Normalization step.

Train\_classification.py - Feature Selection. 11 most correlated features.