perplexity

Detailed Explanation of the Provided Code

The code implements a **stock index prediction web application** using Streamlit, TensorFlow, and various data science libraries. It is designed to predict the closing price of stock indices using a deep learning model (specifically, a hybrid bidirectional LSTM architecture), with features engineered from historical price data. Below is a breakdown of the main components and their roles[1].

1. Imports and Setup

- **Libraries**: Imports standard libraries (`os`, `re`, `sys`, `time`, `warnings`), data processing (`numpy`, `pandas`), deep learning (`tensorflow`, `keras`), and statistical tools (`statsmodels`, `river` for drift detection).
- **GPU Configuration**: Checks for GPU availability and configures TensorFlow to use it if present for faster model training and inference.

2. TradingView Data Integration

- **find_tradingview_data_path()**: Locates the directory containing TradingView data, either in a default path or relative to the script.
- **TradingViewData Import**: Imports classes for fetching historical stock/index data from TradingView.

3. Model Definition: HBLSTMModel

This class encapsulates the hybrid bidirectional LSTM model and related preprocessing.

Constructor (`__init__`)

- Initializes:
  - Model architecture (`_build_model`)
  - Feature and target scalers (`MinMaxScaler`)
  - Drift detector (`ADWIN` from `river`)
  - Window size for time-series data (3 by default)

_build_model()

- **Architecture**:
  - Two Bidirectional LSTM layers (64 and 128 units)
  - Dropout layers (to prevent overfitting)
  - Dense layers for final prediction
- **Loss**: Mean Squared Error (MSE)
- **Optimizer**: Adam

_create_dataset()

- Converts raw tabular data into sequences suitable for LSTM input (sliding window approach).

initial_train()

- Scales features and targets, creates time-series datasets, and trains the model with early stopping.

incremental_update()

- Updates the model incrementally with new data.
- Uses drift detection to decide if full retraining is needed.

predict()

- Generates predictions for new input data, handling scaling and sequence creation.

4. Feature Engineering: `calculate_stock_features()`

- **Calculates technical indicators**:
  - **EMA** (Exponential Moving Average) with various spans
  - **RSI** (Relative Strength Index)
  - **MACD** (Moving Average Convergence Divergence) and its signal line
  - **WMA** (Weighted Moving Average)
  - **Lagged features** (previous values of close, EMA, WMA)
  - **ACF/PACF** (Auto/Partial autocorrelation features)
  - **Correlations** (e.g., close vs EMA5)
  - **Price ratios and returns** (e.g., close/open, percent change)
  - **Volatility** (rolling standard deviation)
- Drops rows with missing values and ensures enough data remains after feature creation.

5. Feature Selection: `load_selected_features()`

- Loads a CSV file mapping datasets to the best feature sets (selected via prior analysis).
- Parses the feature list for each dataset.

6. Prediction and Update Workflow: `predict_and_update()`

- **Data Preparation**:
  - Loads existing and new data for the symbol.
  - Applies feature engineering.
  - Selects features as per the loaded mapping.
- **Model Loading and Prediction**:
  - Loads the pre-trained model.
  - Checks feature compatibility.
  - Scales data and prepares it for LSTM input.
  - Predicts closing prices for the new data.
- **Result Handling**:
  - Inverse transforms predictions to original scale.
  - Stores prediction latency for each input.
  - Appends predictions to the results DataFrame.
- **Incremental Update**:
  - Optionally updates the model with new data.
  - If drift is detected, triggers full retraining.
  - Saves updated model.
- **Returns**: DataFrame with time, actual close, predicted close, and latency.

7. Periodic Update: `periodic_predict_and_update()`

- Runs the prediction and update cycle in a loop, sleeping for a specified interval (default: 15 minutes).

8. Model and Symbol Mappings

- **model_file_mapping**: Maps model filenames to their corresponding CSV data files.
- **symbol_lookup**: Maps dataset keys to (symbol, exchange) pairs for TradingView.

9. Streamlit Web App

- **Title and Sidebar**: Sets the app title and displays model information in the sidebar.
- **Feature File Handling**:
  - Converts Excel features file to CSV if needed.
  - Loads selected features and displays them.
- **Prediction Button**:
  - On click, runs `predict_and_update()` for the selected model.
  - Displays results, including latest actual and predicted close prices.
  - Allows the user to download the latest prediction as a CSV.

Summary Table: Main Components

Component	Purpose
HBLSTMModel	Defines and manages the LSTM-based prediction model, including training, updating, and prediction
calculate_stock_features	Performs feature engineering on raw price data
load_selected_features	Loads feature selection mapping from CSV
predict_and_update	Orchestrates data fetching, feature engineering, prediction, and model updating
Streamlit UI	Provides a user interface for running predictions and viewing results

Key Concepts Used

- **Bidirectional LSTM**: Captures both past and future dependencies in time-series data for better predictive accuracy.
- **Feature Engineering**: Extracts meaningful signals from raw price data to improve model performance.
- **Drift Detection**: Monitors prediction errors to detect changes in data distribution, triggering retraining if needed.
- **Incremental Learning**: Updates the model with new data without full retraining unless drift is detected.
- **Scalability**: Designed to handle multiple indices and models via mapping dictionaries.

How It All Works Together

1. **User loads the app** and selects the model (e.g., NIFTY 15-min).
2. **Features are loaded** from a configuration file.
3. **On prediction request**:

- New data is fetched and engineered.
- The model predicts future close prices.
- Results are displayed and can be downloaded.
- Optionally, the model is updated with the latest data, ensuring it adapts to new trends.

This architecture allows for robust, adaptive, and user-friendly time-series forecasting for financial indices[1].

Reference:[1] See attached code in paste.txt.

⁂

1. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/30719968/c3a266ad-c417-4227-b206-cac7605bf945/paste.txt