**Tiếng Việt:**

**Giới thiệu tổng quan về dự án**

Dự án của bạn tập trung vào việc dự đoán giá cổ phiếu sử dụng nhiều mô hình học máy và học sâu khác nhau. Bạn đã xây dựng một lớp ModelTrainer trong tệp modeling.py để huấn luyện, dự đoán và đánh giá các mô hình này. Các mô hình bao gồm:

* Mạng nơ-ron hồi quy sâu (Deep Learning models):
  + LSTM (Long Short-Term Memory)
  + GRU (Gated Recurrent Unit)
  + BiLSTM (Bidirectional LSTM)
  + TCN (Temporal Convolutional Network)
  + Transformer
* Mô hình học máy truyền thống:
  + XGBoost
  + Random Forest
* Mô hình xếp chồng (Stacking Model), kết hợp các mô hình cơ sở để cải thiện hiệu suất.

**Chi tiết hoạt động của dự án**

**1. Khởi tạo và chuẩn bị dữ liệu**

* **Lớp** ModelTrainer: Đây là lớp chính chịu trách nhiệm cho việc huấn luyện và quản lý các mô hình. Khi khởi tạo, nó nhận vào các tham số:
  + model\_dir: Thư mục lưu trữ các mô hình đã được huấn luyện.
  + time\_step: Số lượng bước thời gian (lags) sử dụng trong mô hình thời gian.
  + num\_features: Số lượng đặc trưng (features) trong dữ liệu.
* **Chuẩn hóa dữ liệu**:
  + Sử dụng MinMaxScaler để chuẩn hóa dữ liệu về khoảng [0, 1]. Điều này giúp tăng hiệu suất và tốc độ huấn luyện cho các mô hình học sâu.
  + Dữ liệu sau khi chuẩn hóa sẽ được reshape để phù hợp với đầu vào của các mô hình.

**2. Huấn luyện các mô hình**

* **Phương thức** train\_models:
  + Duyệt qua danh sách các mô hình cần huấn luyện (model\_types).
  + Kiểm tra xem mô hình đã được lưu trữ trước đó hay chưa. Nếu có, tải mô hình và scaler tương ứng.
  + Nếu mô hình chưa tồn tại:
    - Đối với các mô hình học sâu:
      * Chuẩn hóa dữ liệu.
      * Chia dữ liệu thành tập huấn luyện và tập validation.
      * Gọi phương thức train\_nn\_model để huấn luyện mô hình với Optuna để tối ưu hóa siêu tham số.
    - Đối với các mô hình học máy:
      * Gọi phương thức train\_ml\_model để huấn luyện mô hình với Optuna.
    - Đối với mô hình Stacking:
      * Gọi phương thức train\_stacking\_model để huấn luyện mô hình xếp chồng.
  + Sau khi huấn luyện, lưu trữ mô hình và scaler (nếu cần).

**3. Dự đoán và đánh giá mô hình**

* **Dự đoán**:
  + Sử dụng phương thức predict để thực hiện dự đoán trên tập kiểm tra (X\_test).
  + Đối với các mô hình học sâu, dữ liệu đầu vào cần được chuẩn hóa trước khi dự đoán.
* **Đánh giá hiệu suất**:
  + Sử dụng phương thức evaluate\_performance để tính toán các chỉ số đánh giá như RMSE, MAE, MAPE, SMAPE, R², MedAE.
  + Lưu trữ kết quả đánh giá cho từng mô hình.
* **Ensemble Prediction**:
  + Tính trung bình dự đoán từ tất cả các mô hình để tạo ra dự đoán ensemble.
  + Đánh giá hiệu suất của dự đoán ensemble.

**4. Huấn luyện các mô hình học sâu**

* **Phương thức** train\_nn\_model:
  + Sử dụng **Optuna** để tối ưu hóa siêu tham số cho các mô hình.
  + Các mô hình bao gồm LSTM, GRU, BiLSTM, TCN, Transformer.
  + Định nghĩa các hàm xây dựng mô hình tương ứng với mỗi loại mô hình.
  + Sử dụng EarlyStopping và ReduceLROnPlateau để ngăn chặn quá khớp và điều chỉnh tốc độ học.
  + Huấn luyện mô hình với siêu tham số tốt nhất tìm được từ Optuna.

**5. Huấn luyện các mô hình học máy**

* **Phương thức** train\_ml\_model:
  + Sử dụng Optuna để tối ưu hóa siêu tham số cho XGBoost và Random Forest.
  + Sử dụng TimeSeriesSplit để thực hiện cross-validation phù hợp với dữ liệu chuỗi thời gian.
  + Chọn mô hình với hiệu suất tốt nhất dựa trên kết quả cross-validation.

**6. Huấn luyện mô hình xếp chồng (Stacking Model)**

* **Phương thức** train\_stacking\_model:
  + Tối ưu hóa và huấn luyện các mô hình cơ sở (XGBoost và Random Forest).
  + Tối ưu hóa mô hình meta (sử dụng XGBoost) để kết hợp các dự đoán từ các mô hình cơ sở.
  + Sử dụng cross-validation để đánh giá và chọn siêu tham số tốt nhất cho mô hình meta.

**7. Lưu trữ và tải mô hình**

* **Phương thức** save\_model:
  + Lưu trữ mô hình vào thư mục model\_dir.
  + Đối với mô hình học sâu, lưu trữ dưới định dạng .keras.
  + Đối với mô hình học máy, lưu trữ dưới định dạng .pkl sử dụng joblib.
* **Phương thức** load\_model:
  + Tải mô hình từ thư mục model\_dir.
  + Khôi phục scaler nếu cần thiết cho các mô hình học sâu.

**8. Kết quả và trực quan hóa**

* Lưu trữ kết quả đánh giá vào file CSV ({symbol}\_model\_comparison.csv).
* Vẽ biểu đồ so sánh RMSE giữa các mô hình và lưu trữ dưới dạng hình ảnh.
* Vẽ biểu đồ so sánh dự đoán của từng mô hình với giá trị thực tế.

**Kết luận**

Dự án của bạn là một hệ thống mạnh mẽ cho việc dự đoán giá cổ phiếu, kết hợp giữa các mô hình học sâu tiên tiến và các mô hình học máy truyền thống. Việc sử dụng Optuna giúp tối ưu hóa siêu tham số, cải thiện hiệu suất của các mô hình. Hệ thống được thiết kế linh hoạt, cho phép dễ dàng mở rộng và tùy chỉnh.

**English:**

**Project Overview**

Your project focuses on forecasting stock prices using various machine learning and deep learning models. You have developed a ModelTrainer class in the modeling.py file to train, predict, and evaluate these models. The models include:

* **Deep Learning Models**:
  + LSTM (Long Short-Term Memory)
  + GRU (Gated Recurrent Unit)
  + BiLSTM (Bidirectional LSTM)
  + TCN (Temporal Convolutional Network)
  + Transformer
* **Traditional Machine Learning Models**:
  + XGBoost
  + Random Forest
* **Stacking Model**, which combines base models to enhance performance.

**Detailed Operation of the Project**

**1. Initialization and Data Preparation**

* ModelTrainer **Class**: This is the main class responsible for training and managing models. Upon initialization, it receives parameters:
  + model\_dir: Directory to store the trained models.
  + time\_step: Number of time steps (lags) used in the time series model.
  + num\_features: Number of features in the data.
* **Data Normalization**:
  + Uses MinMaxScaler to normalize the data to the range [0, 1]. This improves the performance and training speed of deep learning models.
  + After normalization, the data is reshaped to fit the input requirements of the models.

**2. Training the Models**

* train\_models **Method**:
  + Iterates through a list of models to train (model\_types).
  + Checks if a model has been previously saved. If so, loads the model and corresponding scaler.
  + If the model doesn't exist:
    - For deep learning models:
      * Normalizes the data.
      * Splits the data into training and validation sets.
      * Calls the train\_nn\_model method to train the model, using Optuna for hyperparameter optimization.
    - For machine learning models:
      * Calls the train\_ml\_model method to train the model with Optuna.
    - For the stacking model:
      * Calls the train\_stacking\_model method to train the stacking model.
  + After training, saves the model and scaler (if necessary).

**3. Prediction and Model Evaluation**

* **Prediction**:
  + Uses the predict method to make predictions on the test set (X\_test).
  + For deep learning models, input data needs to be normalized before prediction.
* **Performance Evaluation**:
  + Uses the evaluate\_performance method to compute evaluation metrics such as RMSE, MAE, MAPE, SMAPE, R², MedAE.
  + Stores evaluation results for each model.
* **Ensemble Prediction**:
  + Calculates the average prediction from all models to create an ensemble prediction.
  + Evaluates the performance of the ensemble prediction.

**4. Training Deep Learning Models**

* train\_nn\_model **Method**:
  + Uses **Optuna** to optimize hyperparameters for the models.
  + Models include LSTM, GRU, BiLSTM, TCN, Transformer.
  + Defines model-building functions corresponding to each model type.
  + Uses EarlyStopping and ReduceLROnPlateau to prevent overfitting and adjust the learning rate.
  + Trains the model with the best hyperparameters found by Optuna.

**5. Training Machine Learning Models**

* train\_ml\_model **Method**:
  + Uses Optuna to optimize hyperparameters for XGBoost and Random Forest.
  + Uses TimeSeriesSplit for cross-validation suitable for time series data.
  + Selects the model with the best performance based on cross-validation results.

**6. Training the Stacking Model**

* train\_stacking\_model **Method**:
  + Optimizes and trains base models (XGBoost and Random Forest).
  + Optimizes the meta-model (using XGBoost) to combine predictions from base models.
  + Uses cross-validation to evaluate and select the best hyperparameters for the meta-model.

**7. Saving and Loading Models**

* save\_model **Method**:
  + Saves models to the model\_dir.
  + For deep learning models, saves in .keras format.
  + For machine learning models, saves in .pkl format using joblib.
* load\_model **Method**:
  + Loads models from the model\_dir.
  + Restores scaler if necessary for deep learning models.

**8. Results and Visualization**

* Stores evaluation results into a CSV file ({symbol}\_model\_comparison.csv).
* Plots RMSE comparison charts among models and saves them as images.
* Plots prediction comparison between each model and actual values.

**Conclusion**

Your project is a robust system for stock price prediction, integrating advanced deep learning models and traditional machine learning models. The use of Optuna for hyperparameter optimization enhances model performance. The system is designed to be flexible, allowing for easy expansion and customization.

**Tóm lại**, dự án của bạn thực hiện quy trình đầy đủ từ chuẩn bị dữ liệu, huấn luyện, tối ưu hóa, dự đoán, đánh giá và trực quan hóa kết quả cho bài toán dự đoán giá cổ phiếu. Việc kết hợp nhiều mô hình khác nhau giúp bạn so sánh và chọn ra mô hình tốt nhất cho dữ liệu của mình.