**LSTM Model 1**

**A graph showing the price of bitcoin

AI-generated content may be incorrect.A graph showing the price of a stock market

AI-generated content may be incorrect.A graph showing the price of a stock market

AI-generated content may be incorrect.A graph showing the price of a stock market

AI-generated content may be incorrect.**

1. **Model Architecture:**  
   The model is built as a sequential neural network composed of three LSTM layers: The first layer is a Bidirectional LSTM with 64 units, which allows the model to capture patterns from both past and future contexts within each sequence. The second and third layers are standard LSTM layers, each with 64 units. Dropout layers with a rate of 0.3 follow each LSTM layer to help prevent overfitting by randomly disabling a fraction of neurons during training. Finally, a Dense output layer with one neuron provides the next-day predicted closing price. The model is compiled using the Adam optimizer with a low learning rate of 1e-4 and is optimized using the mean squared error (MSE) loss function.
2. **Feature Engineering:**  
   The model uses three primary input features: the cryptocurrency’s closing price, a Google Trends score for “bitcoin,” and a sentiment score generated by VADER. These raw features are normalized using a MinMaxScaler with a range of (-1, 1) to better preserve the data’s dynamic range. After scaling, sequences are created using a sliding window of 30 days—each sequence becomes the model’s input to predict the next day’s closing price. This sequential representation helps the LSTM capture the temporal dependencies and patterns present in the historical data.
3. **Training and Hyperparameter Tuning:**  
   For training, the data is split into 80% for training and 20% for testing. The model is trained for 50 epochs with a batch size of 32 and a 10% validation split during training. Although this version uses fixed hyperparameters, the choices (such as the number of LSTM units, dropout rate, and learning rate) were selected after careful experimentation to address issues like flatlining predictions. These hyperparameters can be further tuned using methods such as Bayesian optimization to explore a wider range of values and potentially improve performance further. However, in this version, the fixed settings aim to balance model complexity, training stability, and generalization.

**LSTM Model 2 using Bayesian optimization**

**A graph showing the price of a stock market

AI-generated content may be incorrect.A graph showing the price of a stock market

AI-generated content may be incorrect.A graph showing the price of a stock market

AI-generated content may be incorrect.A graph showing the price of a stock market

AI-generated content may be incorrect.**

1. **Model Architecture:** The LSTM model is built using a deep neural network architecture that begins with a Bidirectional LSTM layer containing 50 units, which helps capture patterns in both forward and backward directions across a 30-day sequence. This is followed by two additional LSTM layers (each with 50 units) interspersed with dropout layers set at a rate of 0.3 to reduce overfitting. The network concludes with a Dense layer consisting of a single neuron that outputs the predicted next day closing price.
2. **Training and Hyperparameter Tuning:** For training, the model uses a batch size of 32 and is allowed to train for up to 100 epochs with 10% of the training data reserved for validation. To optimize the learning process, EarlyStopping is employed to halt training if the validation loss does not improve for 10 consecutive epochs, ensuring the best model weights are restored. Additionally, a ReduceLROnPlateau callback is utilized, which reduces the learning rate by half if no improvement is observed for 5 epochs, helping the model converge more effectively on the optimal solution.
3. **Evaluation Metrics:** The model's performance is assessed using Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE). MAPE provides a relative error measure expressed as a percentage, indicating how far off predictions are on average, while RMSE offers an absolute error measurement in the same units as the cryptocurrency prices and R² which measures how well the model explains the variance in the data. For example, typical evaluation results might show a MAPE of approximately 18% and an RMSE around 19,184 USD for Bitcoin, with lower error metrics observed for other cryptocurrencies like ETH, XRP, and ADA.
4. **Bayesian Optimization Model:** Bayesian optimization is an efficient method for tuning hyperparameters. In our model, we use Keras Tuner's BayesianOptimization, where we first define a search space for key parameters like the number of LSTM units, dropout rates, and the learning rate. Instead of testing every possible combination, the tuner builds a simple model that predicts performance based on past trials. This surrogate model then suggests the most promising hyperparameter combinations by balancing between exploring new values and refining good ones. As a result, we quickly find an effective configuration that improves our LSTM model's performance while reducing the overall number of experiments compared to methods like grid search.

**Key differences in the Model.**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Model 1 (CSV + Fixed LSTM)** | **Model 2 (Yahoo Finance + Bayesian Optimization)** |
| Data Source | Loads cryptocurrency data from CSV files | Fetches cryptocurrency data using the Yahoo Finance API (yfinance) |
| Feature Engineering | Uses three features: crypto closing price, Google Trends score, and sentiment score | Uses the same three features and adds a 10-day Simple Moving Average (SMA) for additional trend smoothing |
| Model Architecture | Fixed architecture: 3 LSTM layers (50 units each) with dropout set at 0.2 | Hypermodel architecture incorporates a Bidirectional LSTM layer and tunes key parameters (e.g., LSTM units, dropout rates, learning rate) using Bayesian optimization |
| Hyperparameter Tuning | Uses fixed hyperparameters with a standard training loop | Uses Keras Tuner’s Bayesian Optimization with callbacks (EarlyStopping and ReduceLROnPlateau) to efficiently explore the hyperparameter space |
| Evaluation & Visualization | Computes evaluation metrics (MAPE and RMSE and R square) and plots the true vs. predicted prices without annotations | Computes the same evaluation metrics but also annotates the plot with the MAPE and RMSE and R square values for a more informative visualization |