Project II
Bitcoin Close Price Prediction

Instructor: Prof. Le Tan Hung
Nguyen Ngoc Khanh 20204915

Bitcoin Close Price Prediction

1. Introduction
2. Dataset
3. Data Preprocessing
4. Long Short Term Memory(LSTM)
5. Gradient Boosting(XGBoost)
6. Final Result
7. Web Application
8. Demo

1. Introduction

- Cryptocurrencies have gained significant attention in recent years
- Bitcoin emerges as the leading and most well-known digital currency
- As the popularity of Bitcoin continues to grow, there is an increasing interest in predicting its price movements
- Objective: build a simple web application powered by machine learning algorithms that allows user to predict the bitcoin close price

2. Dataset

- Cryptocompare python api to get data
- Dataset contains 1001 bitcoin close prices from 2020-10-05 to 2023-07-02
- The dataset is divided into three sets: training set, validation set and test set with the ratio 60%, 20%, 20% respectively in the chronological order

3. Data Preprocessing

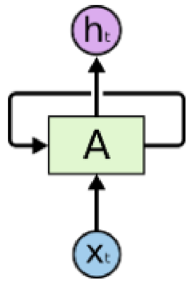
- Optimization algorithm is beneficial from the scaled data
- Before being fetched to the model to train, data is scaled into the range of [0, 1]

$$x_{scaled} = \frac{x - x_{max}}{x_{max} - x_{min}}$$

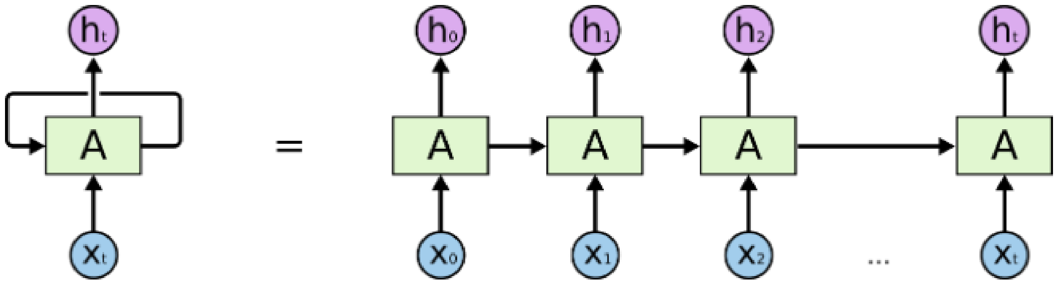
4. Long Short Term Memory(LSTM)

Recurrent Neural Network (RNN)

- RNN is a type of neural network that is capable of processing sequential input, including time-series data
- RNN contains loops that let data from earlier inputs be stored and used to affect the current output



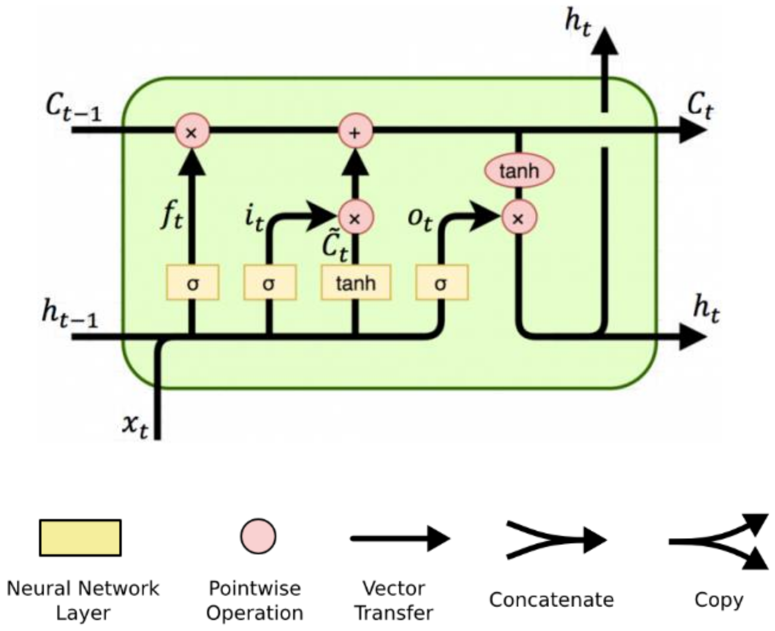
Recurrent Neural Networks have loops.



An unrolled recurrent neural network.

4. Long Short Term Memory(LSTM)

LSTM cell



- LSTM – a special type of RNN is developed to solve the problem of long-term dependencies that RNN fails to solve

- Outputs:

  Cell state C_t - long term memory
  Hidden state h_t - short term memory

- Three gates:

  Forget gate f_t decides which information to carry on
  Input gate i_t decides which information to be updated in the cell state
  Output gate decides which information to go out of a cell

Feed-Forward:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

$$h_t = o_t \odot \tanh(C_t)$$

4. Long Short Term Memory(LSTM)
Application

- Relu activation: guarantee positive bitcoin price
- Dropout: regularization
- Time step: 7
  the number of previous bitcoin close prices to predict the next close price
- Adam optimizer with learning rate = 1e-3

Layer	**Hyperparameters**
LSTM	units = 19, activation = relu
Dropout	dropout_rate = 0.2
Fully connected layer	units = 1

5. Gradient Boosting (XGBoost)

Tree ensemble

A tree emsemble model comprises of K additive functions and is defined as:

$$\hat{y}_i = \phi(\mathrm{x_i}) = \Sigma_{k=1}^{K} f_k(x_i), f_k \in F$$

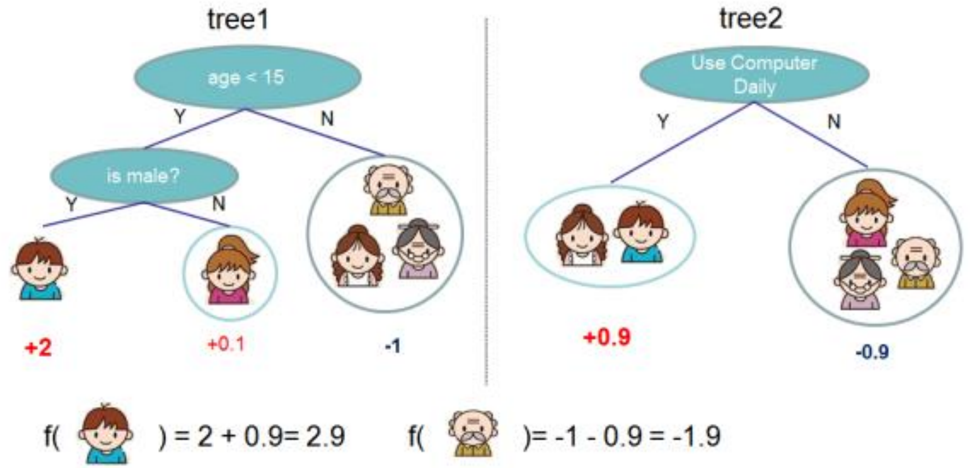
where:

$F = \{f(x) = w_{q(x)}\}(q : R^m \to T, w \in R^T)$ is the space of regression tree

q is the structure of the tree that maps a data point to a leaf index

T is the number of leaves in the tree

w is the leaf weights



Tree Ensemble Model. The final prediction for a given example is the sum of predictions from each tree.

5. Gradient Boosting (XGBoost)

Objective function

$$L(\phi) = \Sigma_i l(y_i, \hat{y}_i) + \Sigma_k \Omega(f_k) \ (1)$$

where

$\Omega(f) = \gamma T + \frac{1}{2}\lambda||w||^2$ that penalizes the complexity of the model

l is a differentiable convex loss function

5. Gradient Boosting (XGBoost)

Optimization

The tree ensemble model is optimized in an additive manner called gradient tree boosting. In the time step t, a new tree $f_t(x_i)$ is added to the model aiming to minimize the cost (1):

$$L^{(t)} = \Sigma_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t)$$

$$\approx \Sigma_{i=1}^{n}\left[l\left(y_i, \hat{y}_i^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t)$$

(second-order approximation)

where $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

$h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

Remove the constant operand $l\left(y_i, \hat{y}_i^{(t-1)}\right)$, minimize $L^{(t)}$ is equivalent to minimize

$$\widetilde{L}^{(t)} = \Sigma_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \gamma T + \frac{1}{2}\lambda \Sigma_{j=1}^{T} \mathrm{w}_\mathrm{j}^2$$

$$= \Sigma_{j=1}^{T}\left[\left(\Sigma_{i\in I_j} g_i\right) w_j + \frac{1}{2}\left(\Sigma_{i\in I_j} h_i + \lambda\right) w_j^2\right] + \gamma T$$

where $I_j = \{i | f_t(x_i) = w_j\}$ is the instance set of leaf j

5. Gradient Boosting (XGBoost)

Optimization

For a fixed tree structure $q(x)$, optimal value of leaf weights:

$$w_j^* = \frac{-\Sigma_{i \in I_j} g_i}{\Sigma_{i \in I_j} h_i + \lambda}$$

$\Rightarrow$ Find a tree structure q(x) that minimize

$$\widetilde{L}^{(t)} = -\frac{1}{2} \Sigma_{j=1}^{T} \frac{\left(\Sigma_{i \in I_j} g_i\right)^2}{\Sigma_{i \in I_j} h_i + \lambda} + \gamma T$$

5. Gradient Boosting (XGBoost)

Optimization

Greedy algorithm to find the tree structure:

Initial tree with a single node

Repeat until meet a stopping criterion:

Split a leaf that maximize this loss reduction

$$L_{split} = \frac{1}{2}\left[\frac{\left(\Sigma_{i\in I_L} g_i\right)^2}{\Sigma_{i\in I_L} h_i + \lambda} + \frac{\left(\Sigma_{i\in I_R} g_i\right)^2}{\Sigma_{i\in I_R} h_i + \lambda} - \frac{(\Sigma_{i\in I} g_i)^2}{\Sigma_{i\in I} h_i + \lambda}\right] - \gamma$$

where $I = I_L \; U \; I_R$

Stopping criterion: maximum depth of the tree

cannot find a split that lead to loss reduction

5. Gradient Boosting (XGBoost)

Hyperparameter

- Loss function $l(y_i, \hat{y}_i)$: mean squared error

$$l(y_i, \hat{y}_i) = \Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Number of trees: 1000

- Time step: 26
  the number of previous bitcoin close prices to predict the next close price

- All other hyperparameters: default values of XGBRegressor

6. Final Result
Evaluation Metrics

Mean Absolute Error(MAE)

$$MAE = \frac{1}{n}\Sigma_{i=1}^{n}|y_i - \hat{y}_i|$$

Model	**MAE**
LSTM	878.91
Gradient Boosting	821.32

6. Final Result
LSTM Predictions

Predictions of the next 100 days

6. Final Result
Gradient Boosting Predictions

Predictions of the next 100 days

7. Web Application

Dependencies

Frontend:

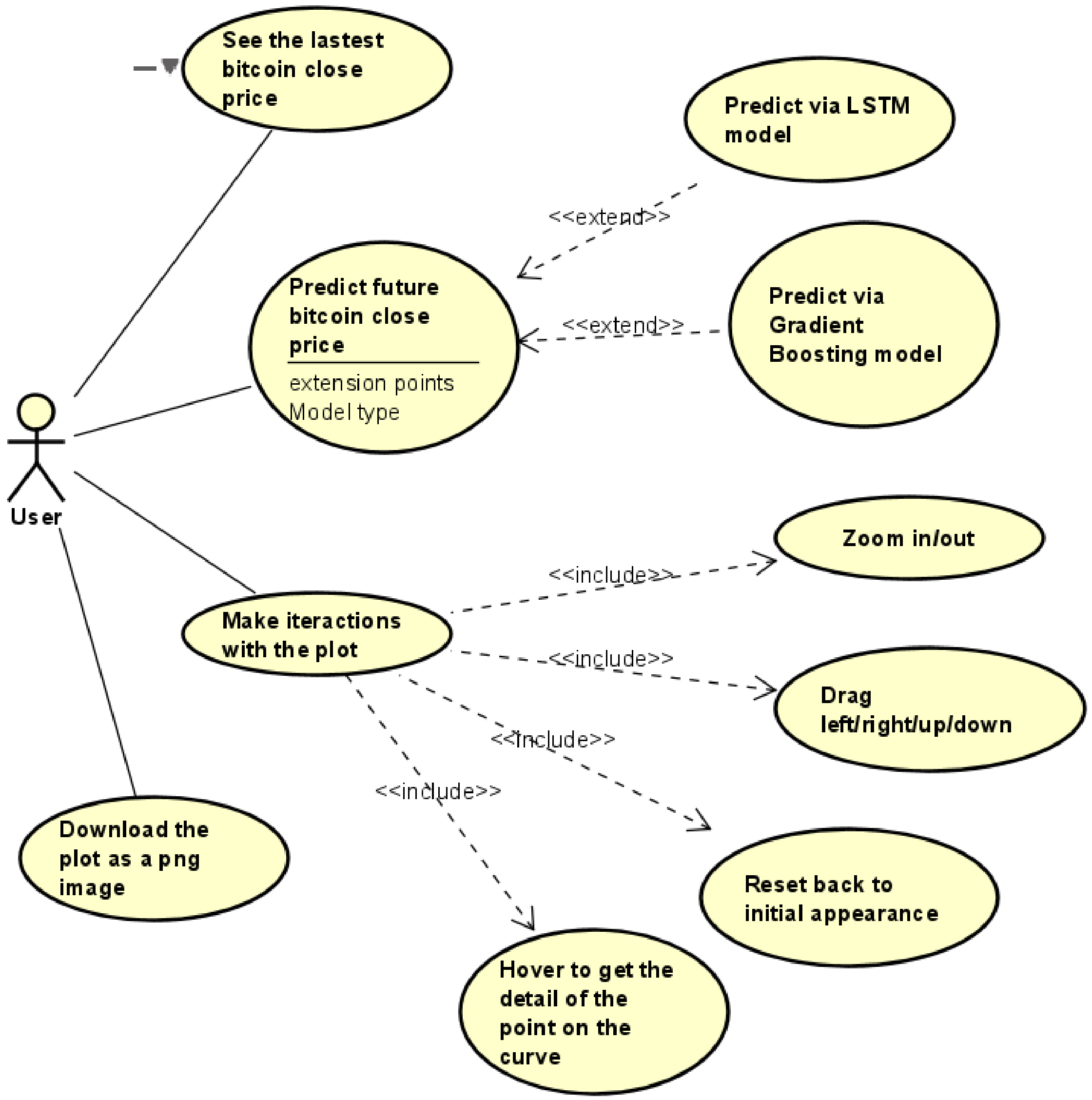
- HTML
- CSS
- Semantic UI

Backend:

- Programming language: Python
- Server-side framework: Django
- Interactive visualization: Plotly
- Machine learning packages: xgboost, tensorflow, sklearn

7. Web Application

UseCase diagram



8. Demo