HKUST CSE/CPEG FYP 2018-2019
Group RO1

Real-time Cryptocurrency Trading Suggestion System Using Machine Learning

Presented by RO1
FONG Chi Chung
SUEN Ka Chun
TANG Marco Kwan Ho

Today's Overview

Flow of the Presentation

Introduction, Design & Implementation

(15mins)

Mobile App Demonstration

(5mins)

Performance, Evaluation and Conclusion

(5mins)

Q&A Session

(10mins)

Introduction, Design and Implementation

Background

Cryptocurrency has become **popular** with:

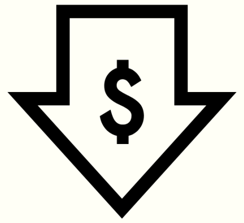
High Volatility

frequent changes in exchange rate provides opportunities to traders

24/7 Market

the trading period is not restricted

Low Entry Price



more flexibility in building portfolios

Background

But, there are **limitations** when human invest in cryptocurrency:

High Volatility

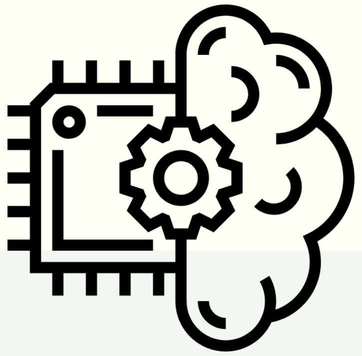
increase risk to investors

Human Emotion

hard to make correct decision

=> So, we need a **Trading Suggestion System**!

Our Final Year Project



Machine Learning

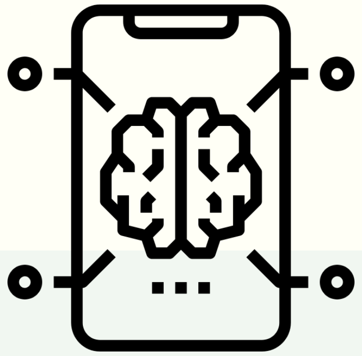
Cryptocurrency price prediction

+

Financial Model

Portfolio building using price prediction

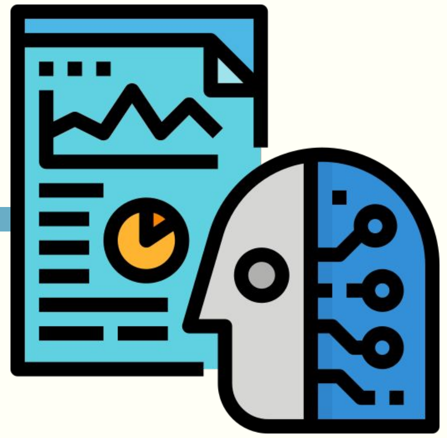
+



Mobile Application

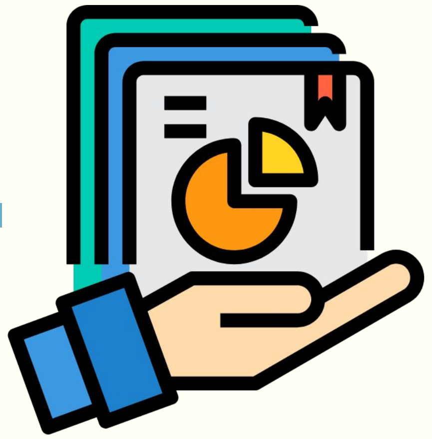
Platform for users to access our results

Objectives



Forecast

Experiment with several **machine learning** technique to see which ones perform best in forecasting cryptocurrency price and trend



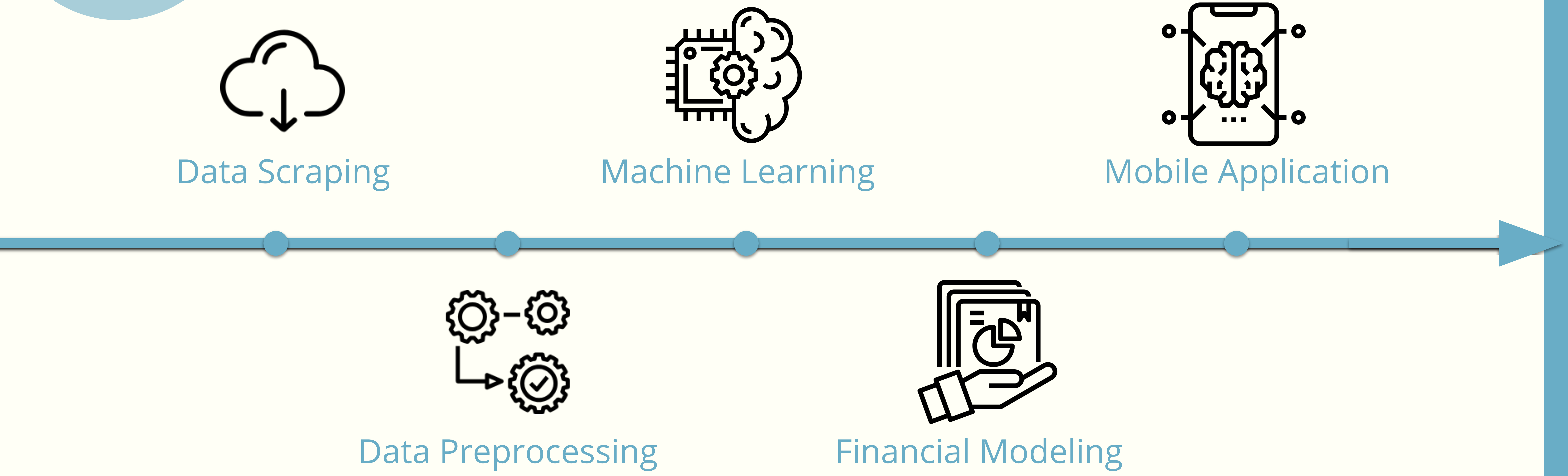
Build

Utilize **financial model** to make trading decision in the cryptocurrency market

Portal

Provide a user-friendly **application** for investors of cryptocurrency to look for trading suggestions

Project Design Flow



Data Scraping

Data Scraping

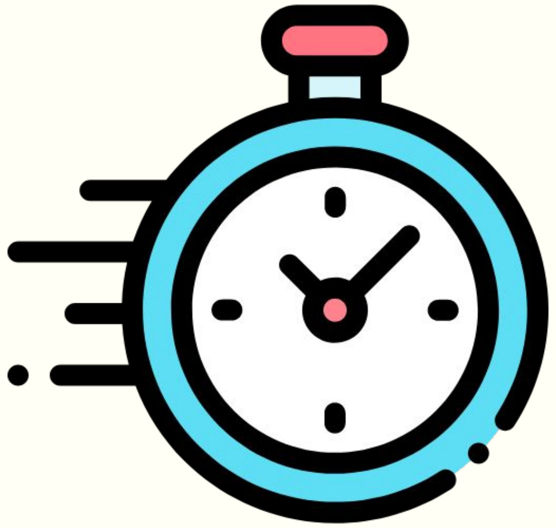
Dataset

2 Cryptocurrency

- Bitcoin
- Ripple

Scraped from 2 Online Exchange

- Bitfinex (historical data)
- Binance (latest data)



1-minute Interval

Data Scraping

MongoDB Database

- NoSQL Database
- Extendable
- JSON format

mongoDB®

Database & Data scraper Design

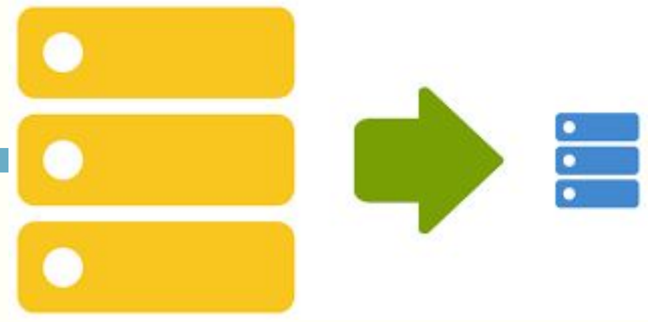
- implemented by python
- implement on Google Cloud

Data Preprocessing

Data preprocessing & Handling

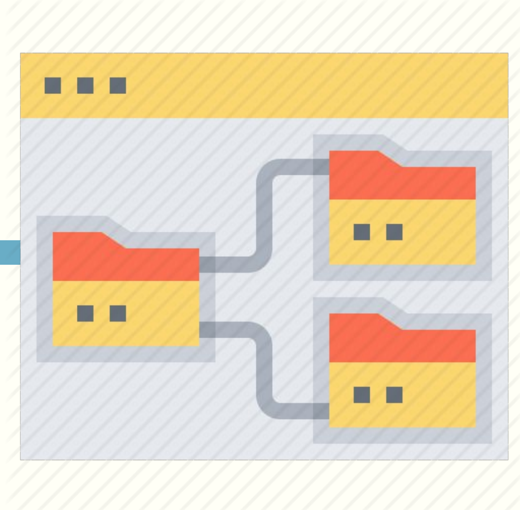
Technical indicator extraction

Extra feature for training the machine learning model



Data normalization

Map the raw data within the range 0 - 1



Dataset splitting

Split the dataset to train and test set

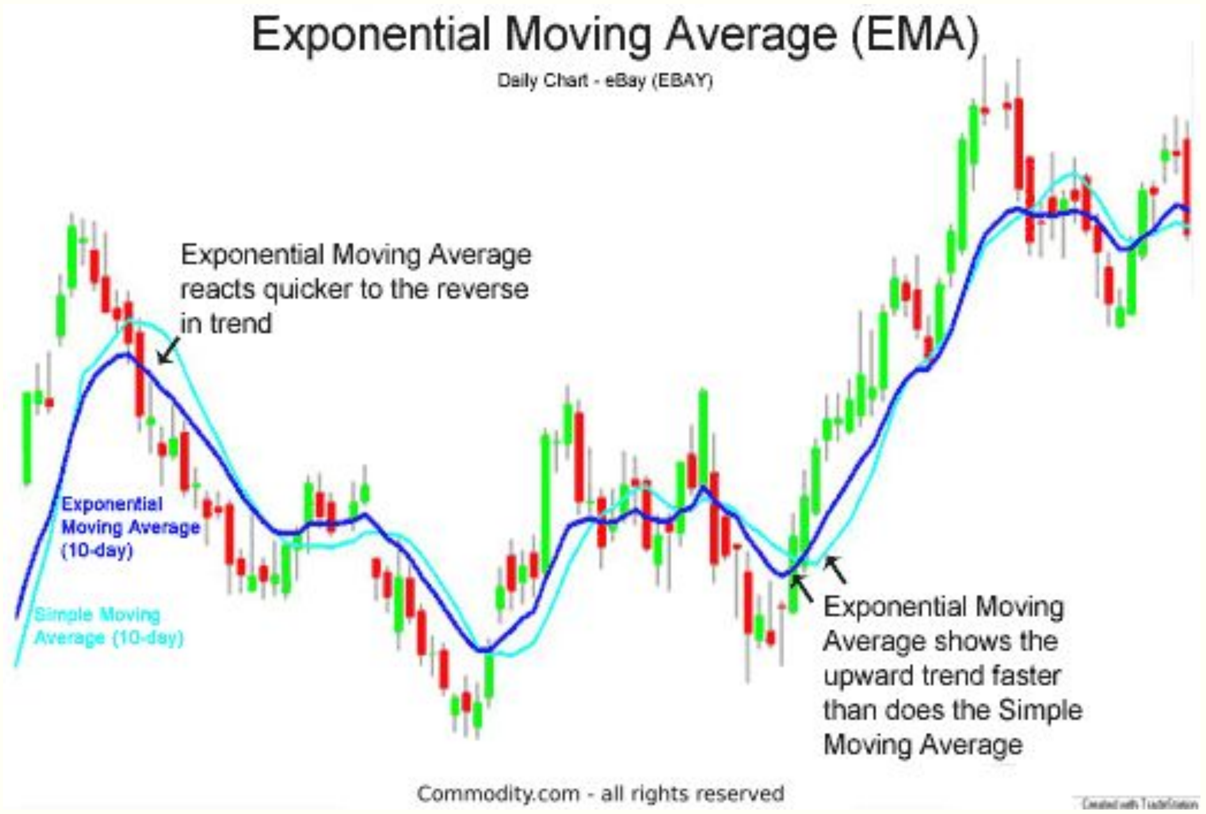
Technical indicator feature extraction

Research from MARA University of technology

Incorporating **MACD** and **EMAs** could increase the performance of machine learning models

MACD (12,26)

EMA 12,26

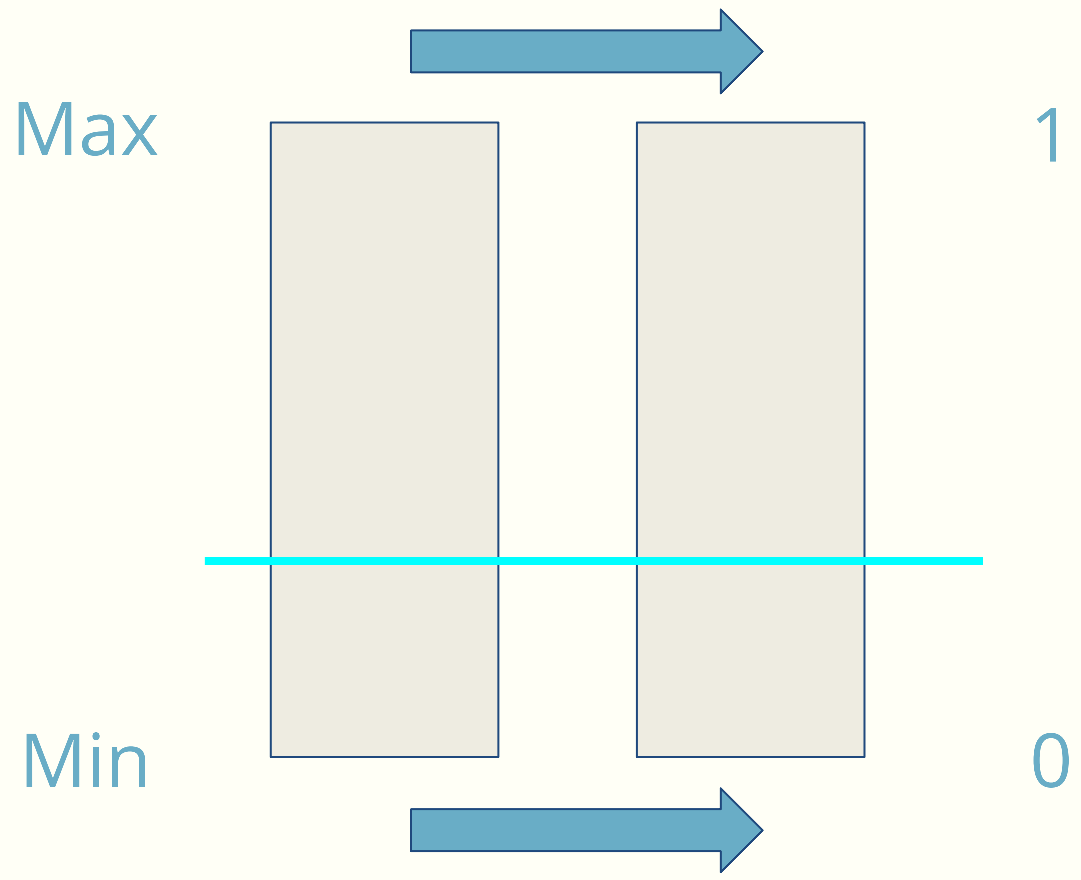


Python script to extract MACD and EMAs from our raw data.

Data ranging and normalization

Map data from 0 - 1 to handle extreme value. Avoid extreme value and weight stucking problem

ln(volume)

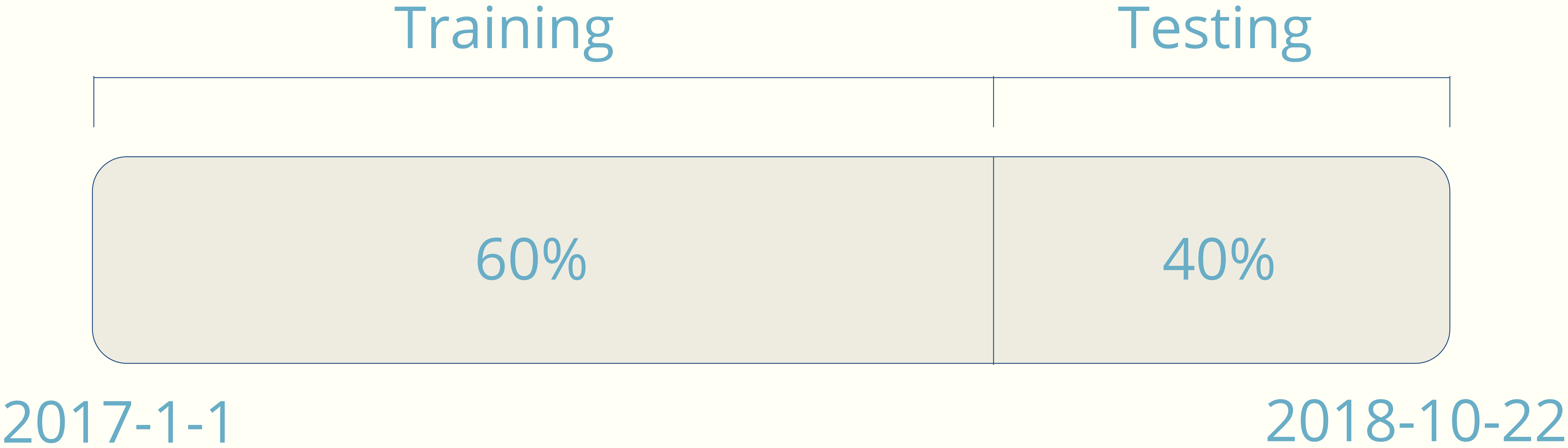


$$\text{encoded data} = \frac{\text{original data} - data_{min}}{data_{max} - data_{min}}$$

Volume

Normalization

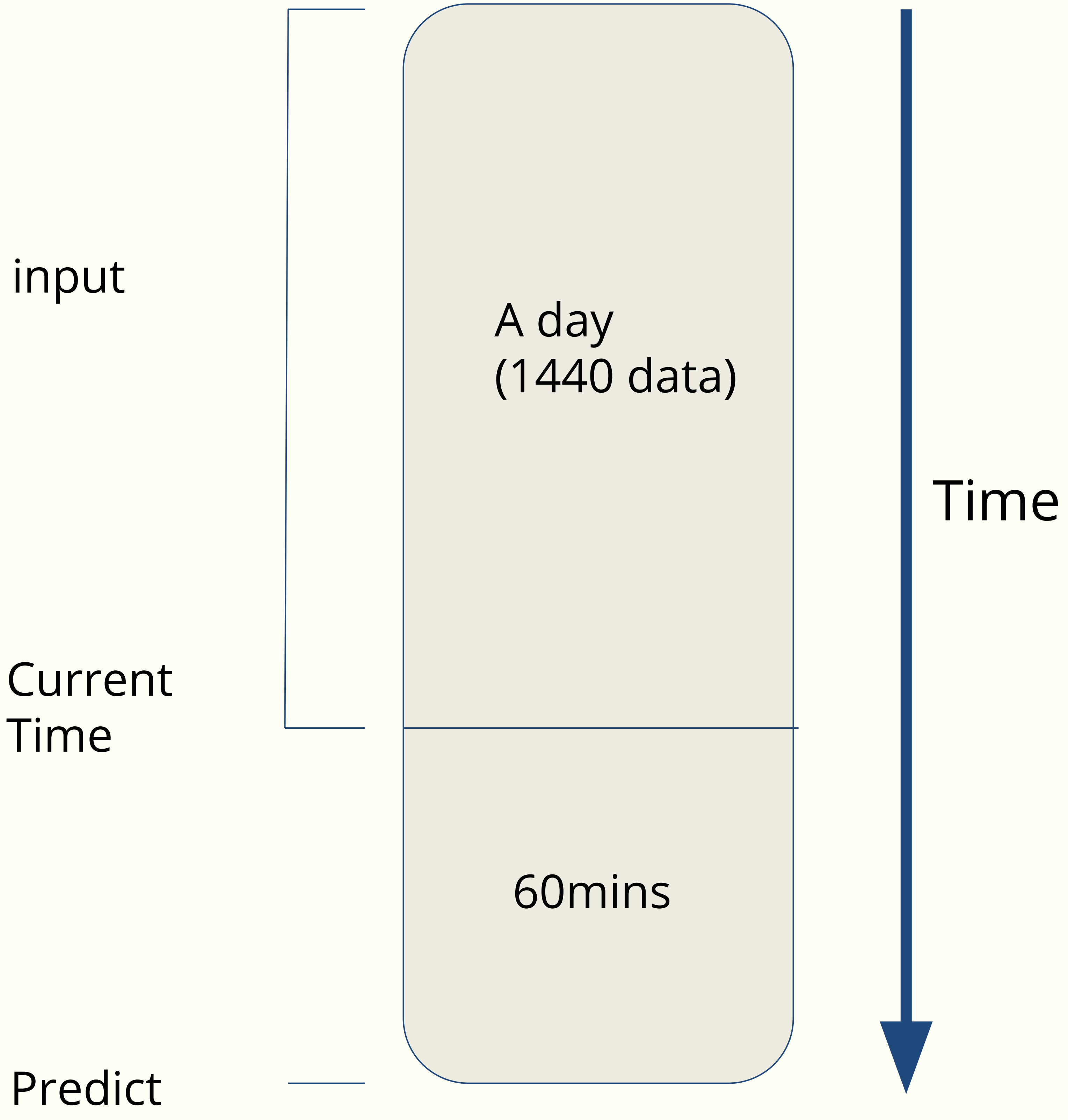
Dataset Splitting



- K-fold cross validation is not applicable in our experiment, Leave a bigger training set
- Trim the dataset before 2017, By experiment decrease MAE by ~0.5%.

** Ripple dataset ranging from 2017-05-19 to 2018-10-22. Also follows the 6:4 ratio

Machine Learning



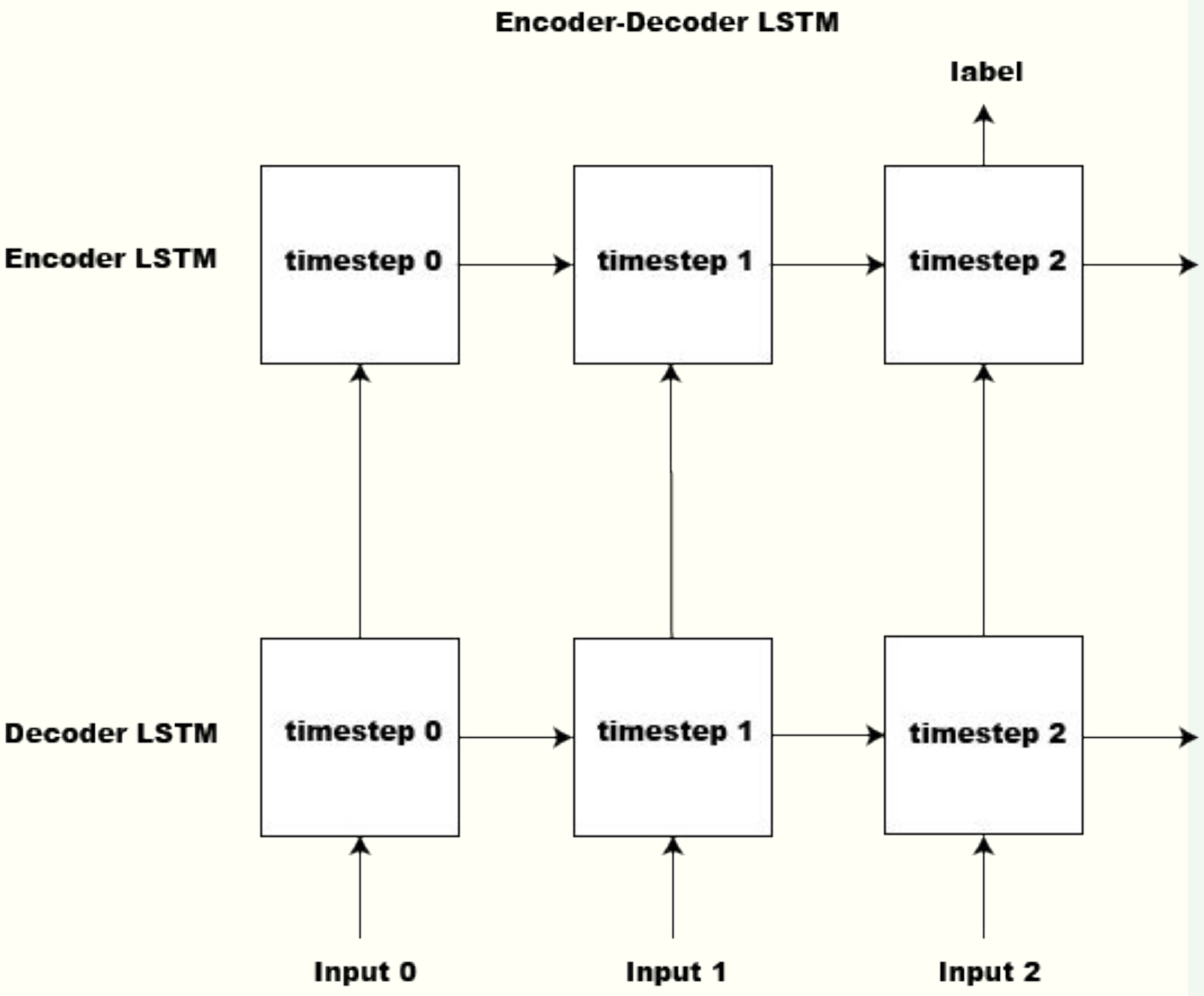
Machine learning models

3 LSTM models

Reading the data of previous Day and predict the price 60 minutes ahead

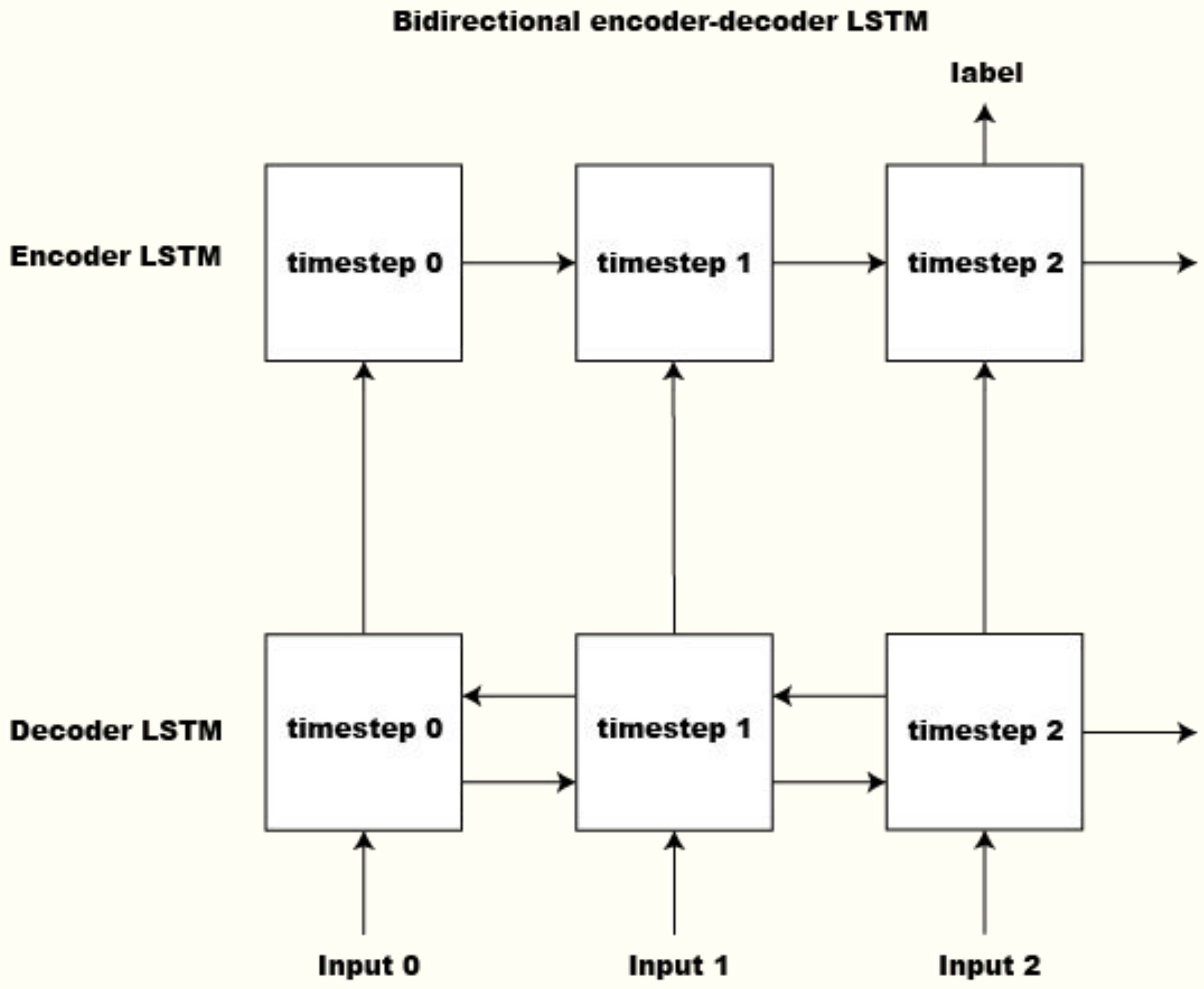
- limitation on hardware

Encoder-Decoder LSTM



- Decoder layer generate data abstraction
- Encoder layer memorize the abstraction from each timestep
- Generate prediction at the last timestep

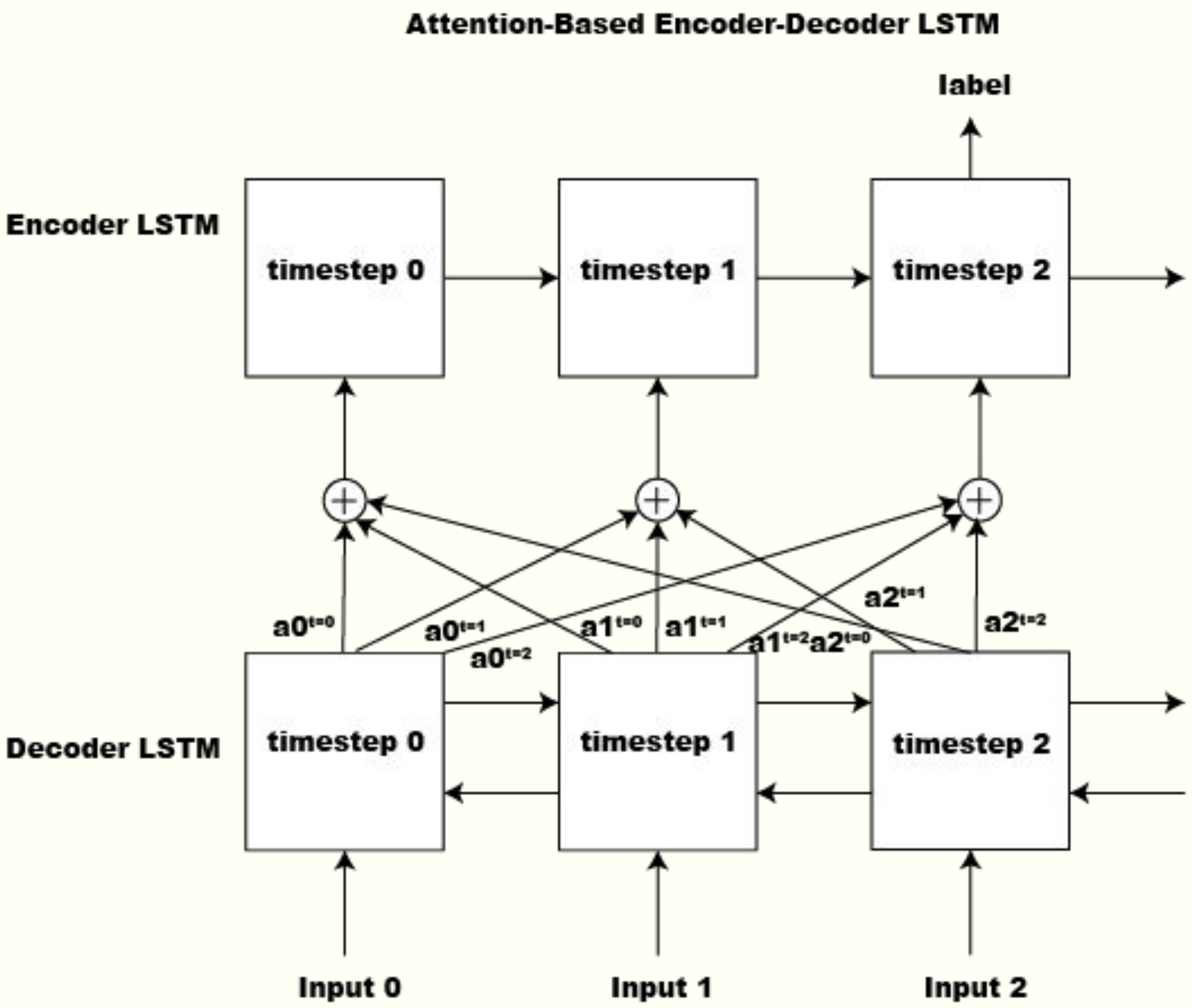
Bidirectional Encoder-Decoder LSTM



- Decoder layer pass it’s hidden state in a bidirectional way.

- bidirectional abstraction

Attention-Based Encoder-Decoder LSTM

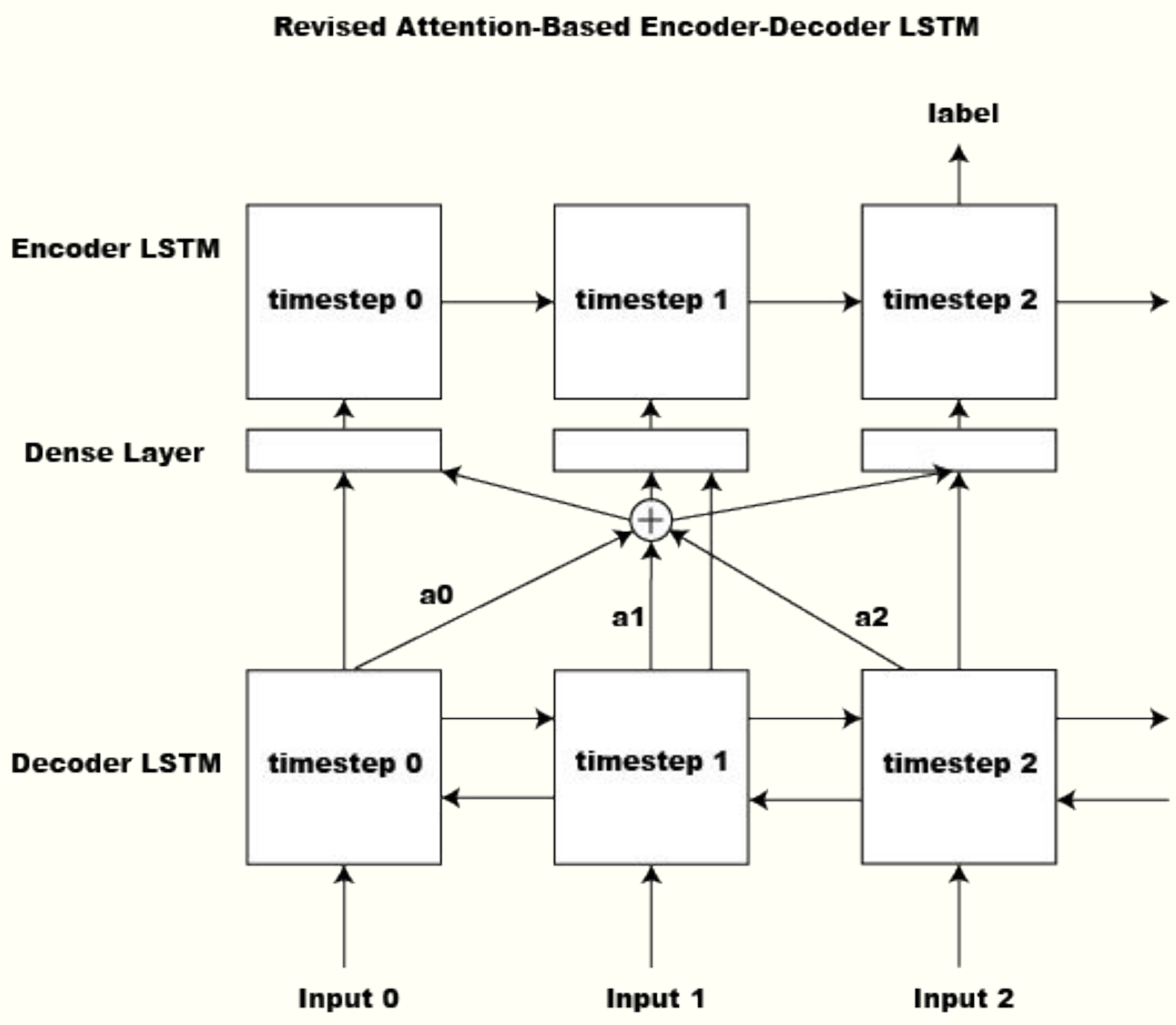


- Provides an attention vector for each timesteps

- Indicate how much attention the Encoder LSTM should pay for the abstraction of each time steps

- Attention Vectors for each time step
  **O(n^2) space complexity !

Revised Attention-Based Encoder-Decoder LSTM



- Provides **ONE** attention vector applies to **EVERY** timesteps
- Dense layer for the model to learn how important is the attention vector.
- Works like an extra summary of the decoded data
- Reduces the space complexity

Hyperparameter tuning & model selection

Scikit Optimize

Stats hyperparameter optimizer package.

50 rounds on:

- number of Decoder LSTM units
- number of Encoder LSTM units
- Learning rate
- number of Dense layers (FC layers)
- number of units of the Dense Layers
- Dense Layers activation functions

Pick the model with **lowest MAE** on price prediction for further evaluation

Financial Modeling

Portfolio Modeling Design

β Value

Volatility Measurement

CAPM

Correlation of Systematic Risk and Expected Returns

Portfolio Allocation

Generate Optimized Portfolios which allocate the portion of coins

β Value

β > 1 : More Volatile

β = 1 : Same Volatility

β < 1 : Less Volatile

$$\beta_i = \frac{Covariance(P_i,\ P_{BTC})}{Variance(P_{BTC})}$$

Bitcoin Price is set to be represent the market as bitcoin is the largest market capitalization coin and settlement currency for mainstream exchanges, meaning that the β value of BTC is 1.

CAPM

$$ER_i = R_f + \beta_i(ER_m - R_f)$$

ER_i	=	Expected return of the Coin
R_f	=	Risk-free rate (USDT)
β_i	=	Beta of the Coin
ER_m	=	Expected return of market
$(ER_m - R_f)$	=	Market risk premium

- Capital Asset Pricing Model
- Get Expected Return of Coin
- USDT is used as Risk-free indicator

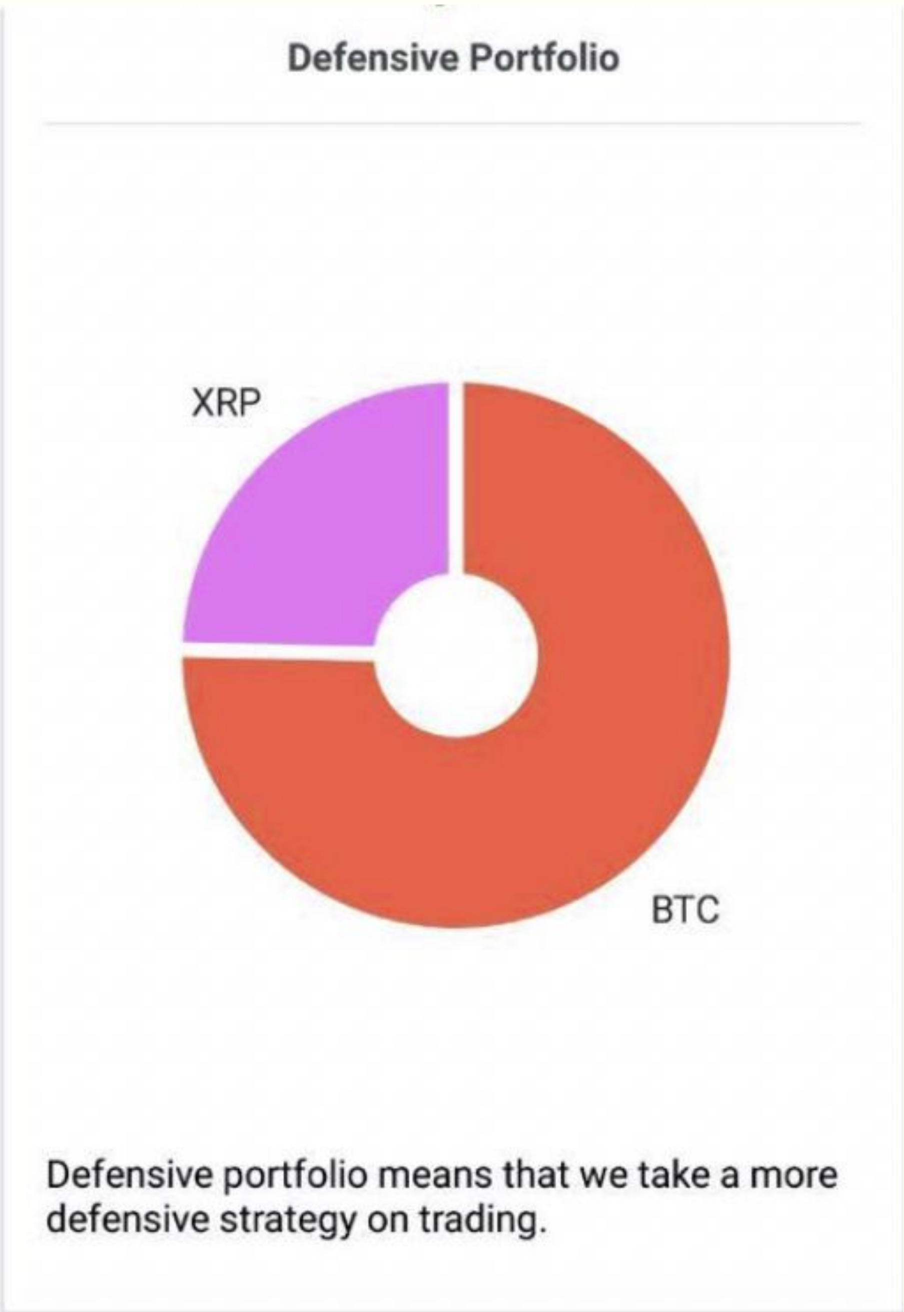
Modern Portfolio Theory

- Math Framework for Assembling Coin Portfolios
  -> Maximize Returns
  -> Choose Risks

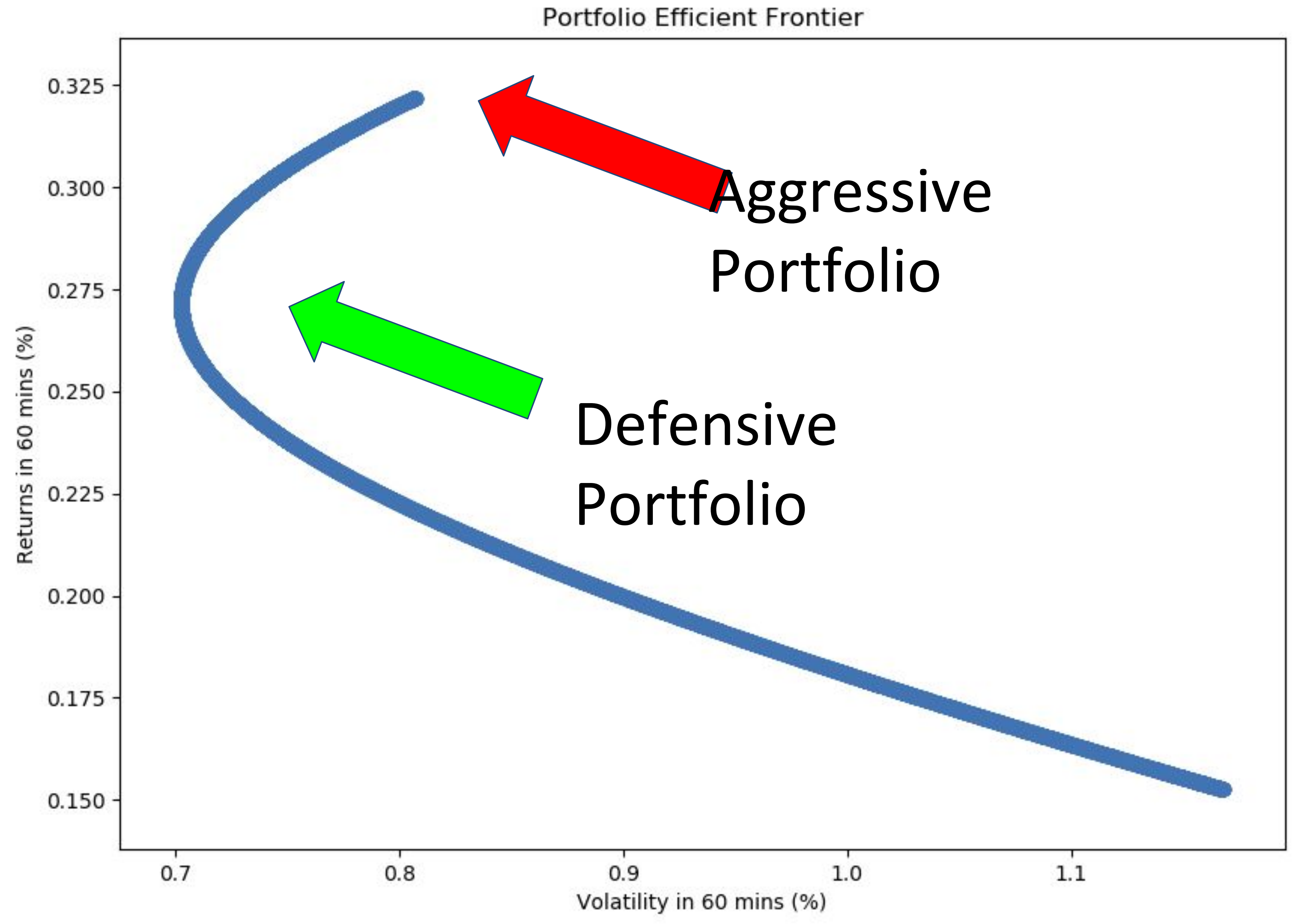
- Returns

$$Returns = \sum_{i=1}^{n} w * R$$

- Covariance of Coins
  - Evaluate the Risks of Coins



Portfolio Efficient Frontier



- 40000 Random Portfolios per tick

- Best Portfolios are on the Frontier

- Aggressive
  -> More Volatility
  -> More Return
  -> Risky

- Defensive
  -> Less Volatility
  -> Less Return
  -> Safer

Implementation

Python

Hosted on Google Cloud Compute Engine

Update Tick by Tick

Support Notification in Minute

Mobile Application

Why Mobile Application?

Download on the App Store

ANDROID APP ON Google play

Real Time Notification

24 Hour Ready

More Users

52.2 % and More

Network Usage Globally[1]

[1] https://www.statista.com/statistics/241462/global-mobile-phone-website-traffic-share/

Tools

React Native — Mobile Application Framework

Firebase — Real-time Database
Backend as a Service

Expo — React Native Toolchain

Mobile Application Service

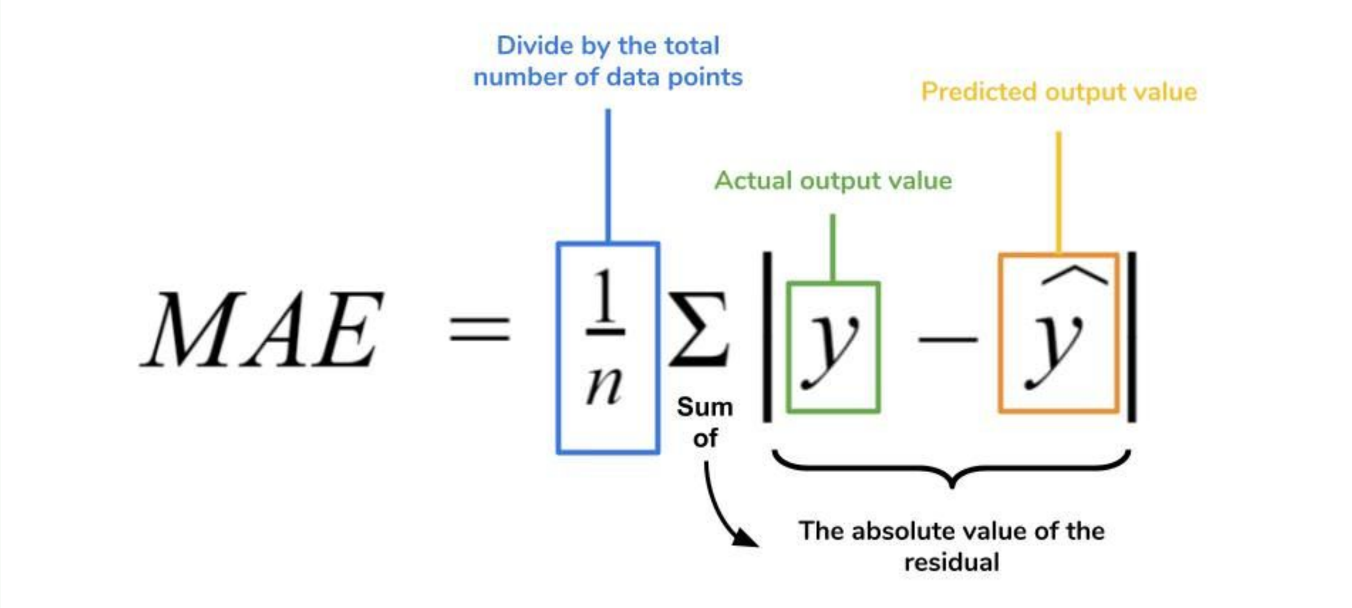
- *Authentication using Firebase*
- *Future 1 Hour Predicted Price*
- *Aggressive and Defensive Portfolio*
- *Push Notification for users*

Mobile Application Demonstration

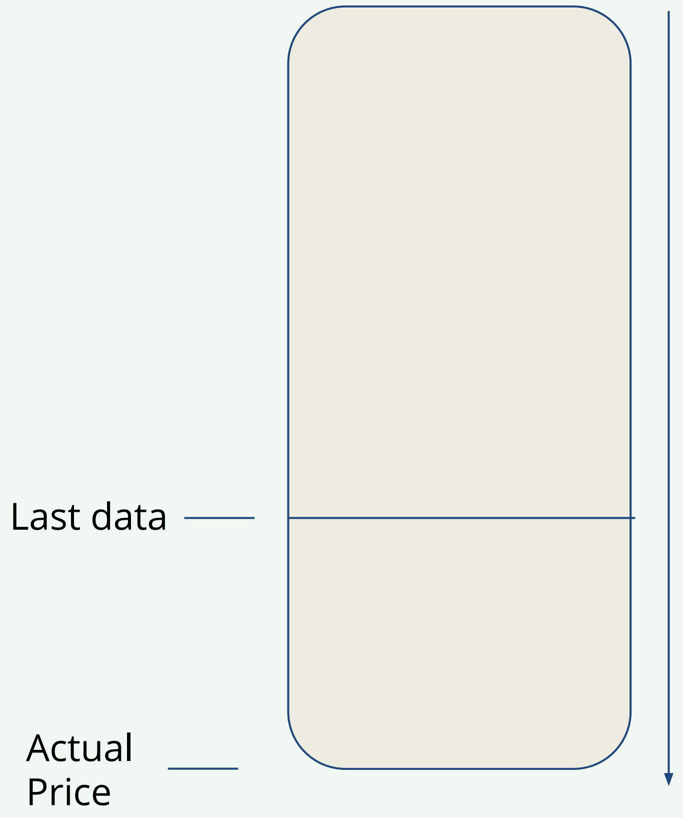
Evaluation, Discussion and Conclusion

Evaluation metrics

MEAN absolute Error



Directional Accuracy



Last data of the input series as reference

Correct prediction:

Actual price & predicted price

- Both larger
- Both Smaller

Than the reference price

$$Directional\ Acc = \frac{num\ correct\ predictions}{num\ samples}$$

Evaluation of Machine learning Models

	Mean absolute error (MAE) on price prediction	Directional Accuracy
Encoder-Decoder LSTM	0.77 %	50.6%
Bidirectional Encoder-Decoder LSTM	0.71%	50.3%
Attention-Based Encoder-Decoder LSTM	1.37%	49.8%

Table 2: Models' Performance on Bitcoin

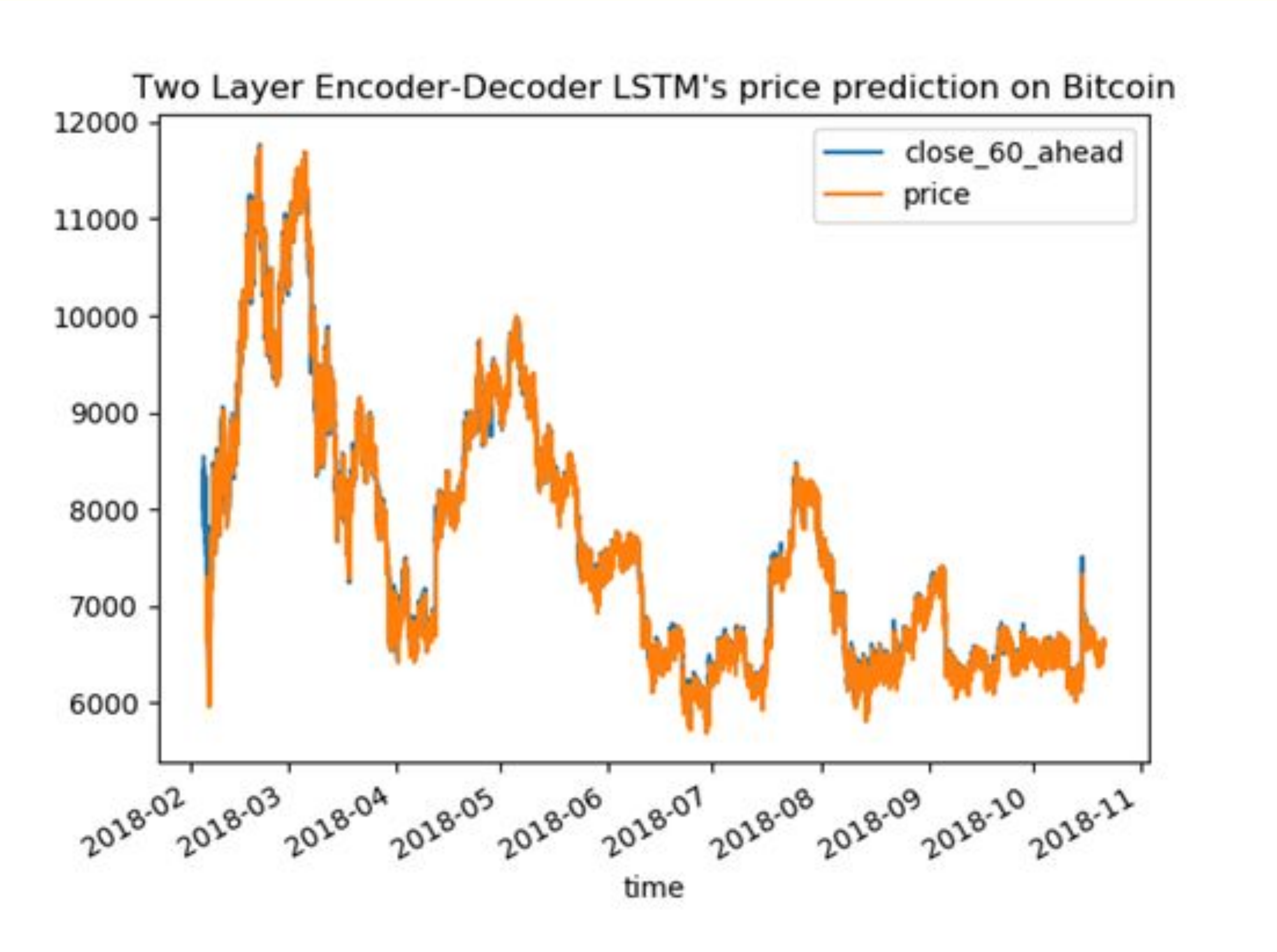
	Mean absolute error (MAE) on price prediction	Directional Accuracy
Encoder-Decoder LSTM	1.36%	50.9%
Bidirectional Encoder-Decoder LSTM	1.91%	50.8%
Attention-Based Encoder-Decoder LSTM	2.29%	50.7%

Table 1: Models' Performance on Ripple

Testset data (370132, 250678) samples in total.

- Encoder-Decoder LSTM is the best.
- Good at predicting the actual price
- Not giving high Directional Accuracy
  - Minute-wise data is noisy
  - Models themselves are regression models
  - Sample size is huge
- Attention & bidirectional architecture do not improve the performance of the LSTM models.
  - Assets type
  - task complexity

Results



Two Layer Encoder-Decoder LSTM's price prediction on Bitcoin
close_60_ahead
price
12000
11000
10000
9000
8000
7000
6000
2018-02
2018-03
2018-04
2018-05
2018-06
2018-07
2018-08
2018-09
2018-10
2018-11
time

Two Layer Encoder-Decoder LSTM's price prediction on Ripple
close_60_ahead
price
0.9
0.8
0.7
0.6
0.5
0.4
0.3
2018-05
2018-06
2018-07
2018-08
2018-09
2018-10
time

Conclusion

- Developed Machine learning models which capable of predict the Future Price.
- Evaluated LSTM ,bidirectional and attention architecture's performance
- Developed Mobile portal to provide real time suggestions.
- Use different dataset (Forex, indexes)
- Use time interval of data
- Combine those result, generate all rounded suggestion system.

Q&A Session

Appendix: Application Evaluation

- JEST

  ->Javascript Testing Framework

SnapShot Test

```
PASS  __tests__/HomeSnapShot.test.js (28.733s)
  <Home />
    ✓ Renders Correctly (111ms)

 › 1 snapshot written.
Snapshot Summary
 › 1 snapshot written from 1 test suite.
 › 1 snapshot file obsolete from 1 test suite. To remove it, run `npm test -- -u`.

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   1 file obsolete, 1 written, 1 total
Time:        29.336s
```

Unit Test

```
PASS  __tests__/Home.test.js (7.43s)
  <Home />
    ✓ has correct child (98ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        7.785s, estimated 11s
```