2021 Spring ML Term Project

Final report

비트코인 차트 분석 프로그램

< 5조 >

201823787 공민정

201723277 김소이

201823776 김한성

201823780 정석화

201720752 최윤지

< 프로젝트 최종 보고서 >

프로젝트 명	비트코인 차트 분석 프로그램
프로젝트 주제	비트코인 차트를 분석하여 코인 가격 예측

■ Introduction

요즘 “비트코인”이란 단어를 중심으로 가상화폐에 대한 이야기가 열풍이다. 그렇지만 주위 사람들이 돈을 벌었다는 이야기를 듣고 코인 투자를 시작하는 사람들이 대부분이다. 주식과는 다르게 코인 시장은 장이 마감하지 않기 때문에 일상생활에 방해가 될 정도로 많은 사람들이 코인시장에 뛰어든 상황이다.

주식과 마찬가지로 코인시장도 공부와 노력이 필요한 분야임에도 불구하고 많은 사람들이 제대로 알지 못하고 시작하는 사람들이 많다. 기본적으로 차트 분석, 호재와 악재 등을 하나도 분석하지 않고 흔히 말하는 “이름 이쁜 것을 사면 돈을 번다.”라는 말이 나올 정도로 많은 사람들이 무작위로 코인 시장에 뛰어들어 투자가 아닌 투기를 하고 있다. 이런 상황으로 인해 “벼락 거지”가 되는 상황이 생겨 책임지지 못할 상황이 생겨나고 있다.

투자의 모든 책임은 본인에게 있기에 이를 돕기 위해 차트 분석 프로그램을 개발하고자 한다. 가격 변동 요인이 많은 비트코인이기 때문에 비트코인 가격에 영향을 주는 요인을 선정하여 모델 학습에 포함시켜 모델의 정확성을 비교해보고 최종적으로 feature를 선정하여 정확도가 높은 모델을 만들도록 할 것이다. 그렇지만 이 프로그램도 어디까지나 투자를 도와줄 보조 역할일 뿐 맹신해서는 안 된다.

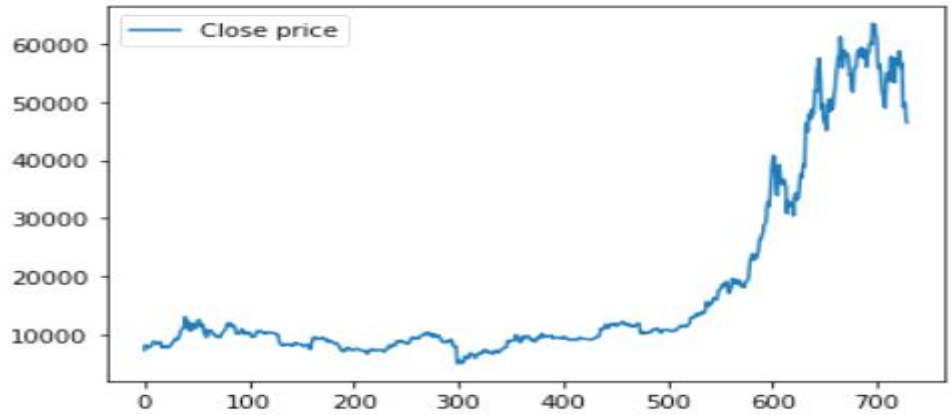
모델 개발은 Google에서 제공하는 Colab을 이용하여 진행하였고 필요한 데이터로는 과거 비트코인 가격에 대한 데이터, 비트코인 관련 주식 데이터, 비트코인 관련 SNS 게시글의 긍부정 데이터가 있다. 모델 개발에 있어서의 성공 지표는 자체적으로 기준을 정하여 정확도가 90%가 넘는다면 투자에 도움이 될 만한 지표라고 정하였다.

최근 일론 머스크와 같은 영향력이 큰 셀럽들이 SNS에 글을 게시하여 가격 변동에 큰 여파를 주는 상황들이 많이 나타나고 있다. 이런 상황에도 이 프로그램이 정확한 예측을 할 수 있는 지에 대한 위험요소가 있을 것 같다.

■ Data Preparation

모델 개발에 필요한 데이터로는 과거 비트코인 가격에 대한 데이터, 비트코인 관련 주식 데이터, 비트코인 관련 SNS 게시 글의 긍부정 데이터가 필요하다.

주식, 비트코인 데이터 셋에는 다양한 feature들이 존재하지만 volume(거래량), Market Cap(시가 총액)은 가격을 예측하는 feature로 사용되기 적합하지 않다고 파악하여 제거하고 사용하였다. 최종적으로 사용한 데이터 셋은 비트코인의 시가, 고가, 저가, 종가, 비트코인 관련 주식들의 종가, 트위터 긍부정 평가지표를 이용하였다. 기본적으로 가격 예측에 대해 사용할 것은 종가 기준으로 진행을 하였다. 다음은 종가 데이터를 가지고 시각화 한 데이터이다.

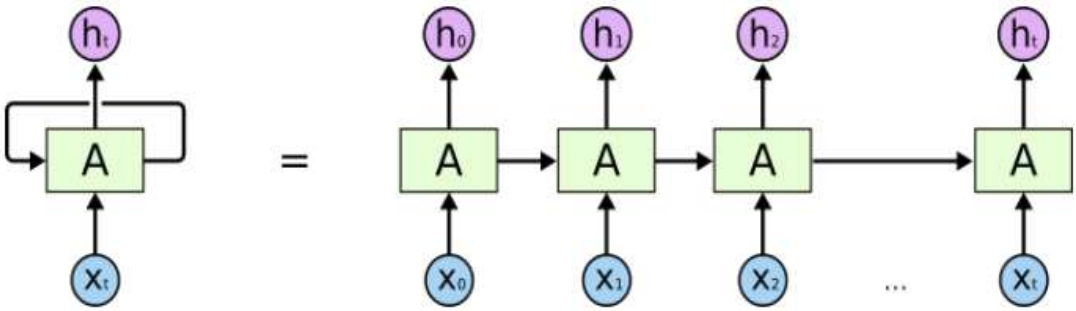


주식과 비트코인 가격에 대한 데이터 셋은 2019년 5월 18일부터 2021년 5월 16일까지의 2년 간의 데이터를 사용했다. 비트코인 거래소와 달리 주식 시장은 주말과 휴일에 열리지 않기 때문에 휴일 날의 주식 데이터에 결측값이 발생한다. 이러한 결측값들은 이전 날의 값을 그대로 사용하는 방식으로 처리했다. 트위터 긍부정 데이터셋은 2021년 4월부터 5월까지의 결측값이 발생했는데 결측값은 데이터 상의 최신 데이터로 결측값을 채웠다. 2년간의 데이터셋 중 680일을 train set, 50일을 test set으로 이용하였다. 그러고 각 데이터마다 데이터의 편차가 크기 때문에 정규화를 진행한 뒤 사용하였다.

< 데이터 출처 >
비트코인 데이터
https://coinmarketcap.com/currencies/bitcoin/historical-data/
Paypal (비트코인 관련 주식)
https://www.marketwatch.com/investing/stock/pypl/download-data
MircroStrategy (비트코인 관련 주식)
https://www.marketwatch.com/investing/stock/mstr/download-data?mod=mw_quote_tab
Vidente (비트코인 관련 주식)
https://www.marketwatch.com/investing/stock/121800/download-data?countrycode=kr&mod=mw_quote_tab
트위터 긍부정 데이터
https://www.kaggle.com/rogerho/ssentiment-analysis-on-5yrs-of-tweets-about-btc

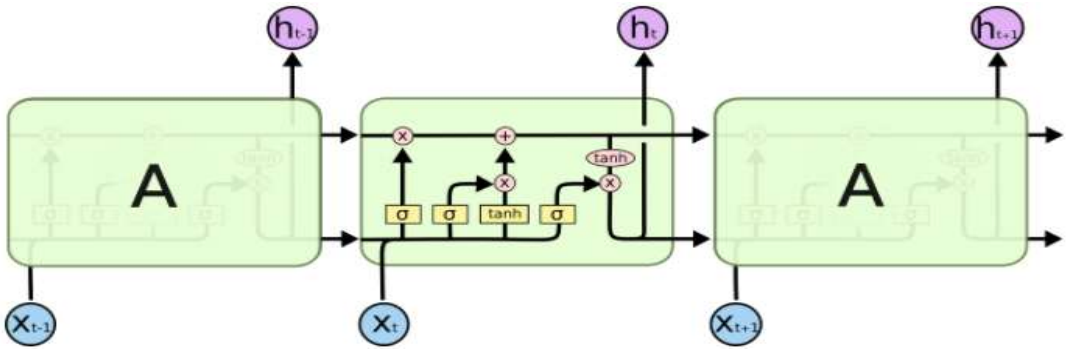
■ Modeling

LSTM은 Recurrent Neural Networks(이하 RNN)의 한 종류이며 순서가 중요한 경우 예를 들면 단어가 문장 안에서 순서가 중요한 경우나, 주가와 같은 시계열 데이터셋에서 효과적인 모델이다. 최근에는 자연어 처리 분야에서 활발하게 사용되어지는 알고리즘이다.
먼저, RNN은 스스로 반복을 진행하면서 이전 단계에서 얻은 정보들을 지속하게 한다.

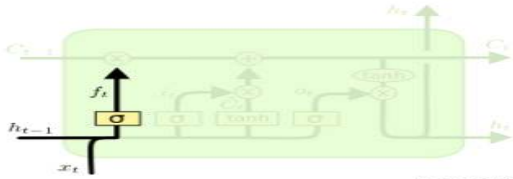


다음과 같은 형태로 하나의 네트워크를 반복구조를 통해 이전의 정보를 넘겨주는 식으로 진행된다. 이런 형태는 곧 sequence, list로 이어지는 것을 알 수 있고 그러한 데이터를 다루기에 최적화된 구조임을 알 수 있다. 그렇지만 길게 이어지는 형태에서 예측해야하는 데이터와 정보가 되는 데이터의 시간격차가 큰 경우 학습 정보를 이어가기가 힘들어진다.

이런 “긴 기간의 의존성(long-term dependencies)”를 해결할 수 있는 것이 LSTM이다.



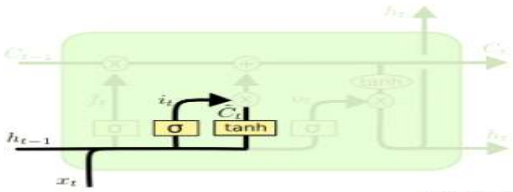
크게 보면 똑같은 구조를 이루지만 각각 모듈을 살펴보면 단순한 neural network는 layer가 한 층이지만, 4개의 layer로 특별한 방식으로 정보를 주고받는다. LSTM의 핵심은 cell state인데 컨테이너 벨트 식으로 되어 있으며 무언가를 더하고 없앨 수 있는 기능이 있는데 이 기능은 gate에 의해서 제어된다. gate는 정보가 전달할 수 있는 추가적인 방법으로 sigmoid layer와 pointwise 곱셈으로 이루어져있다. 어떤 데이터를 버리고, 저장하고, 새로 cell state를 업데이트하고 output으로 내보내는 이 4단계 과정을 통해 하나의 모듈을 지나며 학습을 진행한다.



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

LSTM의 forget gate layer

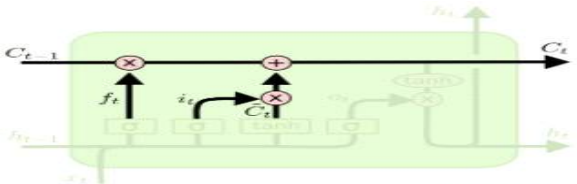
첫 단계로는 cell state로부터 어떤 정보를 버릴 것인지를 정하는 것으로, sigmoid layer에 의해 결정된다. 그래서 이 단계의 gate를 "forget gate layer"라고 부른다. 이 단계에서는 h_{t-1}과 x_t를 받아서 0과 1사이의 값을 C_{t-1}에 보내준다. 그 값이 1이면 모든 정보를 보존하고, 0이면 버리게 된다.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

LSTM의 input gate layer

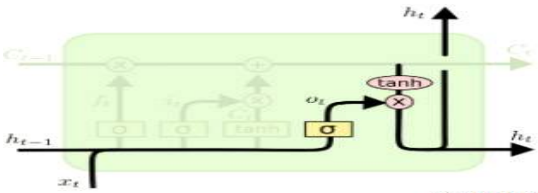
다음 단계는 앞으로 들어오는 새로운 정보 중 어떤 것을 cell state에 저장할 것인지를 정한다. 먼저, "input gate layer"라고 불리는 sigmoid layer가 어떤 값을 업데이트할지 정한다. 그 다음 tanh layer가 새로운 후보 값들인 $\overline{c_t}$라는 vector를 만들고, cell state에 더할 준비를 한다. 이렇게 두 단계에서 나온 정보를 합쳐서 state를 업데이트할 재료를 만들게 된다.



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

LSTM의 cell state 업데이트

세 번째 단계는 과거 state인 C_{t-1}를 업데이트해서 새로운 cell state인 f_t를 만든다. 이전 단계에서 어떤 값을 얼마나 업데이트해야 할지 다 정해놨으므로 실행만 하면 된다. 우선 이전 state에 f_t를 곱해서 가장 첫 단계에서 잊어버리기로 정했던 것들을 지운다. 그 후 $i_t * \overline{C_t}$를 더한다. 이 더하는 값은 두 번째 단계에서 업데이트하기로 한 값을 얼마나 업데이트할 지 정한 만큼 scale한 값이 된다.



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM의 output gate layer

마지막으로 무엇을 output으로 내보낼 지 정하는 일이 남았다. 이 output은 cell state를 바탕으로 필터된 값이 된다. sigmoid layer에 input 데이터를 태워서 cell state의 어느 부분을 output으로 내보낼 지를 정한다. 그 후 cell state를 tanh layer에 태워서 -1과 1 사이의 값을 받은 뒤에 전에 계산한 sigmoid gate의 output과 곱한다. 그렇게 하면 output으로 보내고자 하는 부분만 내보낼 수 있게 된다.

모델 구현은 다음과 같다.

```
class MYLSTM(nn.Module):
  def __init__(self, input_size, hidden_size, num_layers):
    super(MYLSTM, self).__init__()
    self.num_layers = num_layers #number of layers
    self.input_size = input_size #input size
    self.hidden_size = hidden_size #hidden state

    self.lstm = nn.LSTM(input_size=input_size, hidden_size=hidden_size, num_layers=num_layers, batch_first=True) #lstm
    self.fc1 =  nn.Linear(hidden_size, 256) #fully connected
    self.fc2 = nn.Linear(256, 1) #fully connected last layer
    self.relu = nn.ReLU()
```

```
  def forward(self,x):
    h_0 = Variable(torch.zeros(self.num_layers, x.size(0), self.hidden_size)) #hidden state
    c_0 = Variable(torch.zeros(self.num_layers, x.size(0), self.hidden_size)) #internal state
    # Propagate input through LSTM

    output, (hn, cn) = self.lstm(x, (h_0, c_0)) #lstm with input, hidden, and internal state

    hn = hn.view(-1, self.hidden_size) #reshaping the data for Dense layer next
    out = self.relu(hn)
    out = self.fc1(out) #first Dense
    out = self.relu(out) #relu
    out = self.fc2(out) #Final Output

    return out
```

파라미터 값을 바꾸어가며 학습을 여러 번 진행해보면서 최적의 파라미터 값을 찾을 수 있었다. 손실함수로는 regression에서 주로 사용되는 MSE Loss를 사용하였고 Optimizer는 Adam을 이용하여 구현하였다. 학습을 진행하며 loss가 overfitting이 나지 않도록 3000 epoch만 학습을 진행하였다.

```
num_epochs = 3000
learning_rate = 0.00001

input_size = 8 #number of features
hidden_size = 30 #number of features in hidden state
num_layers = 1 #number of stacked lstm layers

lstm = MYLSTM(input_size, hidden_size, num_layers)
loss_function = torch.nn.MSELoss()    # mean-squared error for regression
optimizer = torch.optim.Adam(lstm.parameters(), lr=learning_rate)  # adam optimizer
```

■ Evaluation

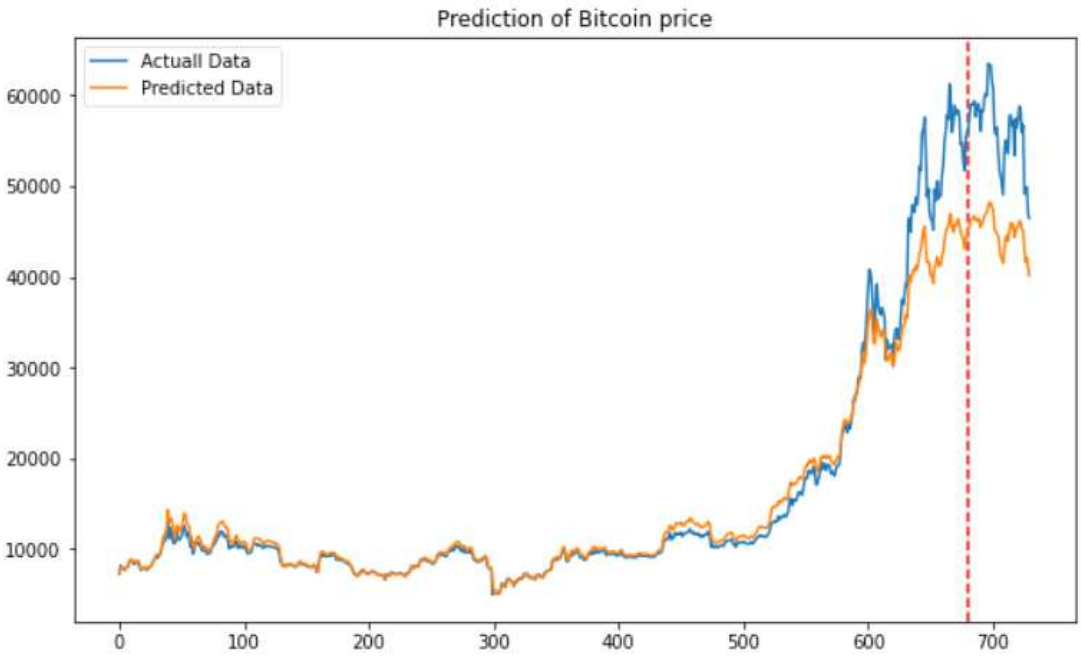
모델의 성능을 평가하기 위한 평가지표는 자체적으로 기준을 다음과 같이 정하였다.

정확도 : 100 - (|실제가 - 예측가 | / 실제가) * 100

regression 모델에는 다양한 evaluation 방법은 있지만 백분율로 정확도를 표기하기 위해 팀원들과 상의하여 기준을 세웠다.

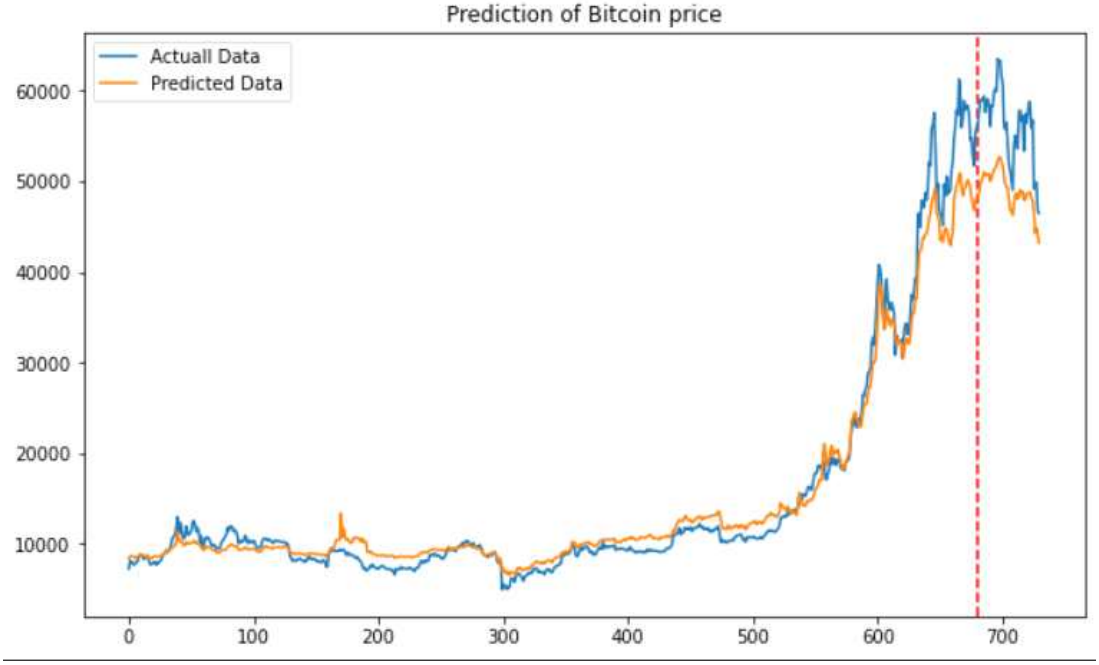
최적의 모델을 구현하기 위해 기준점으로 잡은 큰 특징은 비트코인 가격으로만 예측, 주식 관련 데이터를 추가하여 예측, 트위터 긍부정 데이터를 추가하여 예측을 진행하기로 하였다. 그래프 상 파란색은 실제 데이터를 의미하고 주황색은 예측 데이터를 의미한다. 빨간 점선 기점으로 왼쪽은 학습한 상황이고 오른쪽은 테스트를 진행했을 때의 모습을 나타낸다.

*** 비트코인 가격으로만 예측**



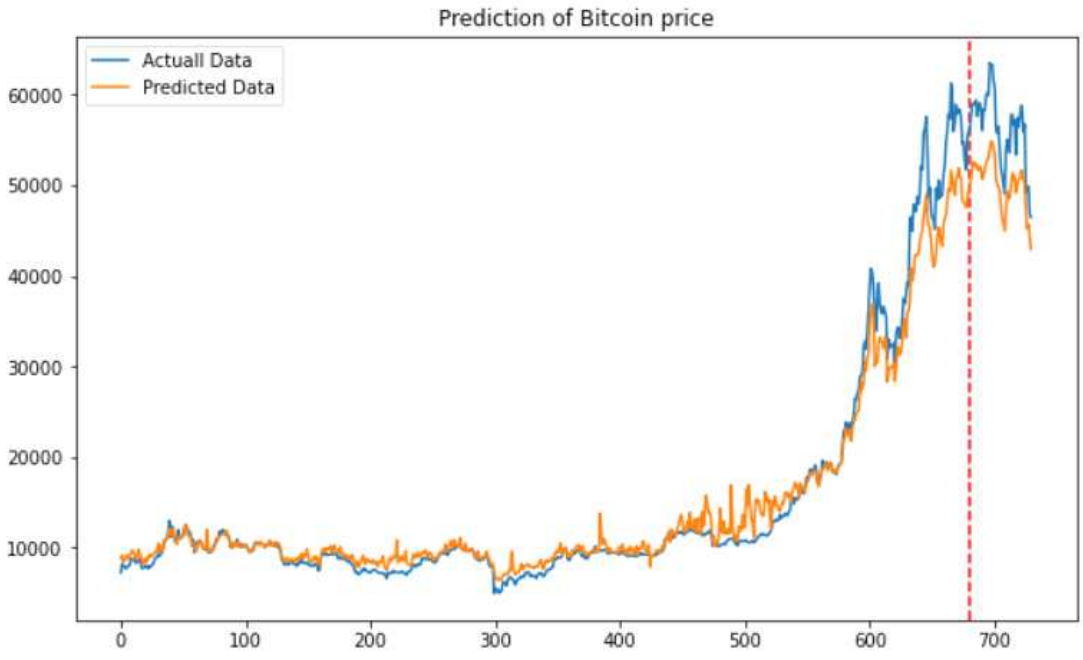
정확도 : 80.31%

* 주식 관련 데이터 추가하여 예측



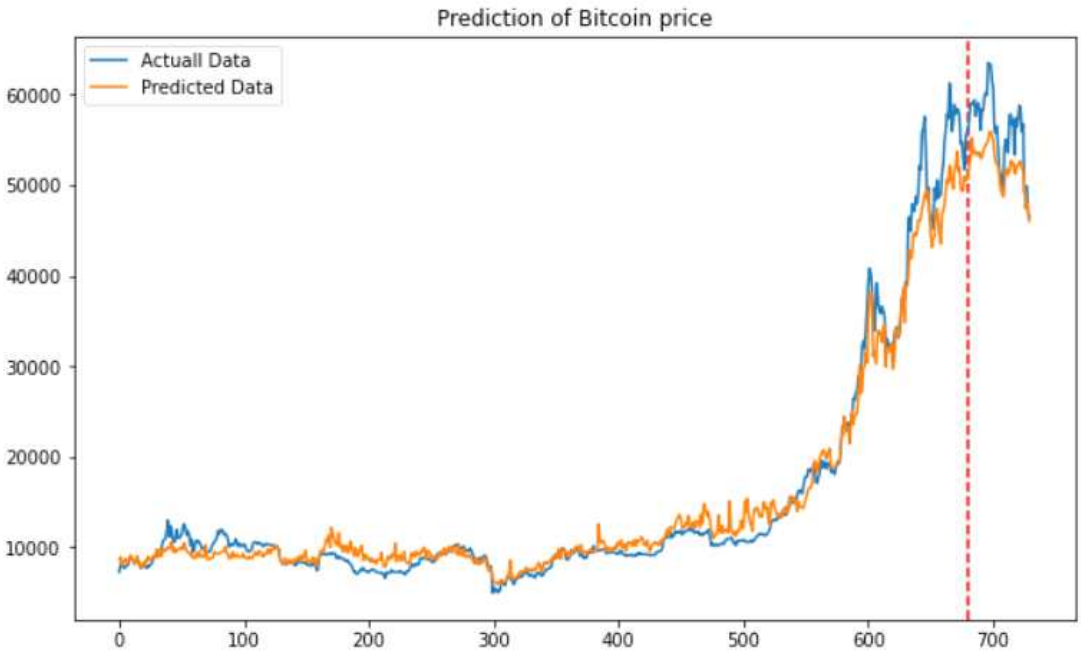
정확도 : 87.27%

* 트위터 긍부정 데이터 추가하여 예측



정확도 : 89.24%

*** 주식 관련 데이터, 트위터 긍부정 데이터 추가하여 예측**



정확도 : 92.75%

모델 구현 전 비트코인 관련 주식과 SNS관련 게시 글의 영향이 비트코인 가격 예측에 영향을 주는지에 대한 의문이 생겼지만 다음과 같이 4차례에 걸쳐서 특징들을 각각 추가해 보고 학습을 진행해본 결과 비트코인 관련 주식, SNS 긍부정 데이터 셋 모두 비트코인 가격 예측에 영향을 주는 것을 확인할 수 있었다. 최종적으로도 90% 이상의 정확도를 넘어서 모델이 나타내주는 데이터의 신뢰성을 높일 수 있었다.

■ Conclusion

기존 비트코인 가격 데이터에 추가적으로 넣는 feature를 4가지 단계로 각각 학습시켰다.

① 비트코인
② 비트코인 + 주식데이터(PayPal, MicroStrategy, Vidente)
③ 비트코인 + 트위터 긍부정 데이터
④ 비트코인 + 주식데이터(PayPal, MicroStrategy, Vidente) + 트위터 긍부정 데이터

각 모델의 학습결과 정확도가 높은 순: ④ > ③ > ② > ①

기존의 가격 예측 프로그램은 단순히 해당하는 가격만을 가지고 학습시킨 모델이었다면 해당 프로젝트에서는 외부적인 요인을 feature로 추가하여 여러 요인을 반영하여 학습할 수 있도록 하였다. 추가적인 변동 요인으로 PayPal과 MicroStrategy 그리고 Vidente의 주식 데이터와 트위터에서 올라온 비트코인의 게시 글에 대한 긍부정 요인을 분석한 데이터를 추가하였다. 그 결과, 비트코인 데이터로만 학습한 예측 모델보다 더욱 높은 정확도가 나왔기 때문에 더욱 놓은 정확도를 가진 예측 프로그램을 만들 수 있었다.

데이터 수집에 대한 한계점으로 인해 더욱 다양한 경우의 수를 확인해보지 못하였다. 프로젝트에서 활용한 데이터의 기간이 대략 2년 정도였다. 이 데이터 양 만으로도 충분히 높은 정확도를 가진 모델을 만들 수 있었지만 더 많은 기간의 데이터를 학습에 활용한다면 어떤 결과가 나올지 비교할 수 없었다는 아쉬움이 있었다. 또한 처음에 명시하였던 일론머스크와 같은 영향력이 큰 셀럽들이 SNS에 글을 게시할 때의 비트코인 가격의 영향에 대한 위험요소를 언급하였는데 이는 셀럽들이 글을 언제 게시하고, 어떠한 글을 게시하는지에 대한 예측이 불가하여 그 여파에 대한 예측은 하지 못할 것으로 결론을 내렸다.

비트코인 가격 예측 프로그램을 구현해보았는데 최종적으로 성능은 92.75%로 꽤 좋은 성능이 나왔으며 이 프로그램을 이용하여 많은 사람들이 투기가 아닌 투자를 하여 안전한 자산 투자가 되었으면 한다.