# Project_M

# 스터디 보고서

2025 / 02 / 26

박정하

# 목차

# 1. LSTM 주가 예측 프로그램

# LSTM 주가 예측 프로그램

- **종목 : Tesla**
  - **변동성이 큰 주식이므로, 인공지능 모델이 이러한 큰 변동성을 효과적으로 예측할 수 있는지 검증해보고자 함.**



- **학습 데이터**
  - 종가(Close Price) – 단순한 시계열 예측
  - 종가 + 거래량(Volume)
  - 종가 + 이동평균선 + RSI + 거래량
  - 종가 + 시가 + 저가 + 거래량 + 이동평균선 + RSI + 볼린저 밴드

- **이동평균선(MACD) : 일정 기간동안 주가를 평균하여 선으로 나타낸 지표**
- **상대강도지수(RSI) : 가격 움직임을 분석하는 기술적 지표**
- **볼린저 밴드 : 주가의 변동성을 분석하는 지표**

# LSTM 주가 예측 프로그램

- Step 1. 주가 데이터 불러오기

```python
import yfinance as yf
import pandas as pd

ticker = "TSLA"
start_date = "2023-02-25"
end_date = "2024-02-25"

data = yf.download(ticker, start=start_date, end=end_date)
data = data[['Close']]  #종가 선정
```

- yfinance 모듈을 이용해 주가 데이터 불러옴.
- 기간은 23-02-25 ~ 25-02-25
- 종가에 대한 데이터만 추출

```
----------------------------------------
Price           Close
Ticker           TSLA
Date
2023-02-27  207.630005
2023-02-28  205.710007
2023-03-01  202.770004
2023-03-02  190.899994
2023-03-03  197.789993
----------------------------------------
(250, 1)
----------------------------------------
 Max : Price  Ticker
Close  TSLA      293.339996
dtype: float64
----------------------------------------
 Min : Price  Ticker
Close  TSLA      153.75
dtype: float64
```

- 총 250일 치에 대한 주가 데이터를 가져왔고
- 최고가일때 주가 확인
- 최저가일때 주가 확인

# LSTM 주가 예측 프로그램

- Step 2. 데이터 정규화
  - 데이터를 0~1 사이의 값으로 변환하는 과정
  - 입력데이터가 너무 크거나 차이가 클 경우 학습이 어려울 수 있음
  - 더 효율적인 학습 가능
  - Min-Max Scaling

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

  - 파이썬에서는 sklearn모듈을 이용해 쉽게 정규화 가능

```
1 from sklearn.preprocessing import MinMaxScaler
2 scaler = MinMaxScaler(feature_range=(0, 1))
3 data_scaled = scaler.fit_transform(data)
```

```
[[0.38598758]
 [0.37223303]
 [0.35117133]
 [0.26613651]
 [0.31549534]]
--------------------------------------
Max : 1.0
--------------------------------------
Min : 0.0
```

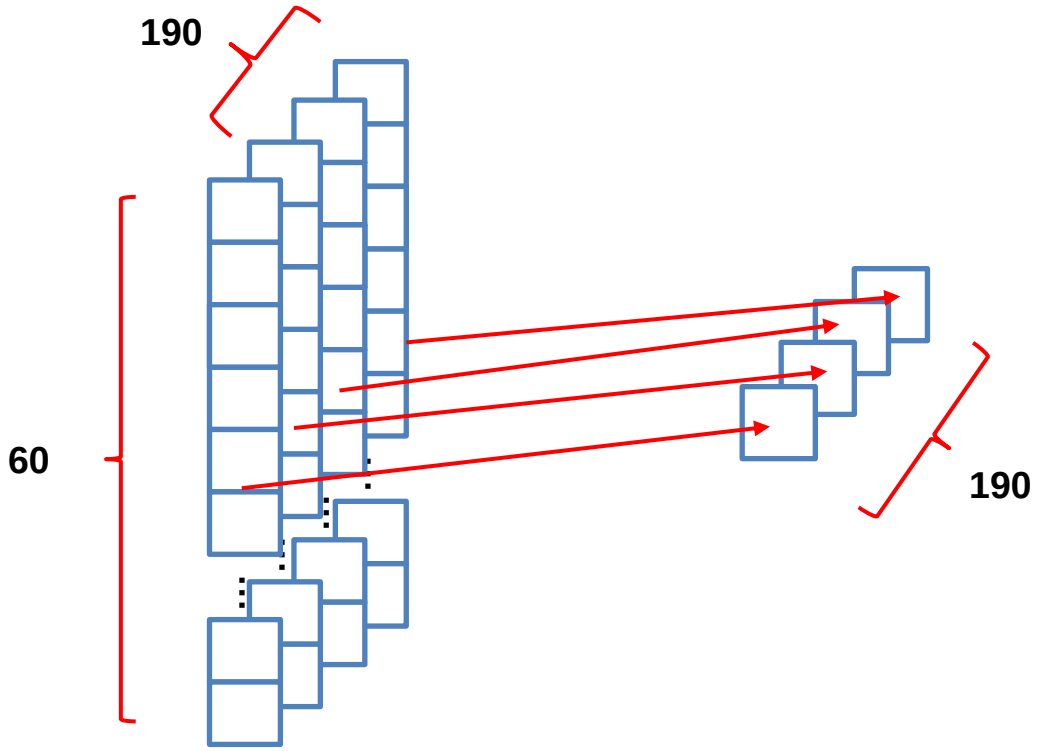
# LSTM 주가 예측 프로그램

- Step 2. 데이터 분할
  - X 에는 60일치에 대한 주가 데이터
  - Y에는 다음날에 대한 주가 데이터
  - EX) for문 안에서 I =1 일때 X에는 1~60일에 해당하는 주가, y에는 61일에 해당하는 주가
  - For문 안에서 총 250-60 = 190개의 데이터 생성

```python
def create_sequences(data, seq_length):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i+seq_length])
        y.append(data[i+seq_length])
        # print(i)
    return np.array(X), np.array(y)
#sep_length는 과거 몇일치에 대한 데이터로 다음 날 주가를 예측할지 결정
seq_length = 60  #60일치를 이용해 주가 예측
X, y = create_sequences(data_scaled, seq_length)
```

```python
print(X.shape)
print(y.shape)
```

```
(190, 60, 1)
(190, 1)
```



# LSTM 주가 예측 프로그램

- Step 2. 데이터 분할
  - 훈련 데이터 80%, 테스트 데이터 20%로 분할

```
split = int(len(X) * 0.8)
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(X_test.shape)
```

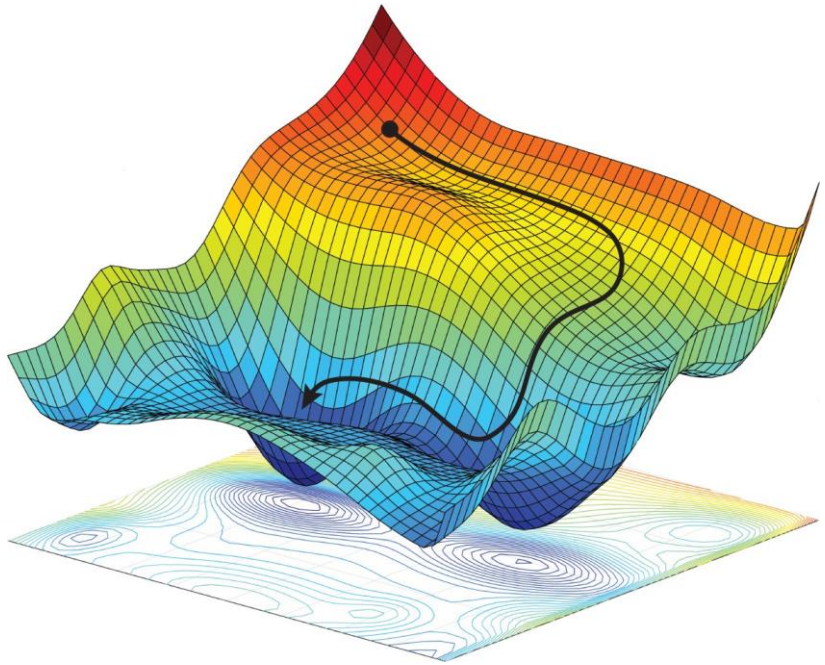
```
(152, 60, 1)
(38, 60, 1)
(152, 1)
(38, 60, 1)
```

# LSTM 주가 예측 프로그램

- Step 3. LSTM 모델 구축
  - 최적화 기법 : Adam : SGD(경사 하강법)을 개선한 알고리즘
  - 손실함수 : MSE(평균 제곱 오차) - 주가는 연속적인 변수이며 이를 예측하는 것은 회귀 문제이기 때문에 MSE가 적합

- 회귀예측 (Regression)
  - MSE

$$J(\theta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$



# 2. 향후 계획

# 향후 계획

- LSTM 에 대한 심층적인 이해 필요

- 종가만을 이용해 모델을 만들고 하이퍼파라미터 변경을 통해 성능 개선
  (히든층의 개수, 레이어의 노드, dropout 등 변경)

- 입력 데이터의 종류를 늘려 여러 데이터의 조합으로 학습한 모델의 성능 비교

# 3. 코딩테스트 문제 선별

# 코딩테스트 문제 선별 - 링크

## 그리디 알고리즘 (탐욕법)

`지금 당장 가장 좋은 것`을 고르는 알고리즘
-> 그리디 알고리즘이 출제되면 **그리디로 얻은 답이 최적의 결과**를 가짐

### 설탕 배달 다국어

| 시간 제한 | 메모리 제한 | 제출 | 정답 | 맞힌 사람 | 정답 비율 |
|---|---|---|---|---|---|
| 1 초 | 128 MB | 362924 | 140728 | 104138 | 38.099% |

### 문제

상근이는 요즘 설탕공장에서 설탕을 배달하고 있다. 상근이는 지금 사탕가게에 설탕을 정확하게 N킬로그램을 배달해야 한다. 설탕공장에서 만드는 설탕은 봉지에 담겨져 있다. 봉지는 3킬로그램 봉지와 5킬로그램 봉지가 있다.

상근이는 귀찮기 때문에, 최대한 적은 봉지를 들고 가려고 한다. 예를 들어, 18킬로그램 설탕을 배달해야 할 때, 3킬로그램 봉지 6개를 가져가도 되지만, 5킬로그램 3개와 3킬로그램 1개를 배달하면, 더 적은 개수의 봉지를 배달할 수 있다.

상근이가 설탕을 정확하게 N킬로그램 배달해야 할 때, 봉지 몇 개를 가져가면 되는지 그 수를 구하는 프로그램을 작성하시오.

# 코딩테스트 문제 선별 - 링크

## ATM

| 시간 제한 | 메모리 제한 | 제출 | 정답 | 맞힌 사람 | 정답 비율 |
|---|---|---|---|---|---|
| 1 초 | 256 MB | 129815 | 89717 | 70913 | 69.463% |

### 문제

인하은행에는 ATM이 1대밖에 없다. 지금 이 ATM앞에 N명의 사람들이 줄을 서있다. 사람은 1번부터 N번까지 번호가 매겨져 있으며, i번 사람이 돈을 인출하는데 걸리는 시간은 $P_i$분이다.

사람들이 줄을 서는 순서에 따라서, 돈을 인출하는데 필요한 시간의 합이 달라지게 된다. 예를 들어, 총 5명이 있고, $P_1 = 3$, $P_2 = 1$, $P_3 = 4$, $P_4 = 3$, $P_5 = 2$ 인 경우를 생각해보자. [1, 2, 3, 4, 5] 순서로 줄을 선다면, 1번 사람은 3분만에 돈을 뽑을 수 있다. 2번 사람은 1번 사람이 돈을 뽑을 때 까지 기다려야 하기 때문에, 3+1 = 4분이 걸리게 된다. 3번 사람은 1번, 2번 사람이 돈을 뽑을 때까지 기다려야 하기 때문에, 총 3+1+4 = 8분이 필요하게 된다. 4번 사람은 3+1+4+3 = 11분, 5번 사람은 3+1+4+3+2 = 13분이 걸리게 된다. 이 경우에 각 사람이 돈을 인출하는데 필요한 시간의 합은 3+4+8+11+13 = 39분이 된다.

줄을 [2, 5, 1, 4, 3] 순서로 줄을 서면, 2번 사람은 1분만에, 5번 사람은 1+2 = 3분, 1번 사람은 1+2+3 = 6분, 4번 사람은 1+2+3+3 = 9분, 3번 사람은 1+2+3+3+4 = 13분이 걸리게 된다. 각 사람이 돈을 인출하는데 필요한 시간의 합은 1+3+6+9+13 = 32분이다. 이 방법보다 더 필요한 시간의 합을 최소로 만들 수는 없다.

줄을 서 있는 사람의 수 N과 각 사람이 돈을 인출하는데 걸리는 시간 $P_i$가 주어졌을 때, 각 사람이 돈을 인출하는데 필요한 시간의 합의 최솟값을 구하는 프로그램을 작성하시오.

# 감사합니다