주식투자 예측 프로그램

202104005 김예나
201804008 김우원
202207018 장정규

목차

- 프로그램 필요성
- 프로그램 소개
- 데이터 분석
- 사용 알고리즘
- 순서도 및 구현
- 성능 평가
- 결론

프로그램 필요성

- 데이터를 기반으로 패턴을 학습하고 미래를 예측하는 것이 핵심이라고 생각했기 때문에 주가와 관련있는 데이터들을 선정한 후 학습 데이터셋과 테스트 데이터셋으로 분리 후 모델을 학습시켜 테스트 데이터로 성능을 평가하는 방법을 사용한다.
- 기존에 나와있는 주식예측 프로그램의 데이터 셋보다 더 많은 데이터셋을 확보하기 위해, 증권사 HTS를 사용하여 데이터셋에 다양한 주식 관련 항목들을 추가시켜 좀 더 정확한 예측 모델을 만들고자 했다.

프로그램 소개

- 제목 : 주식예측 프로그램
- 목적 : 주식 가격 움직임을 예측하여 예측한 방향성을 토대로, 투자자들이 더 나은 투자 전략을 가지고 주식 투자 결정을 할 수 있도록 돕는 것이다.
- 요약 : 종가, 시가, 고가, 저가등의 정보가 포함된 10년간의 주식 시계열 데이터를 입력하여 학습시킨 후, 다음 날의 종가를 예측한다.
- 기대효과 : 투자자들의 리스크를 줄여 투자 의사 결정을 돕고 투자 전략을 테스트할 수 있게 하는 것을 기대효과로 가진다.

데이터 분석

- 유진투자 증권 HTS (https://www.eugenefn.com/main.do)

인덱스	이름	데이터타입	값	설명
0	Date	범주형	숫자	주식거래일자
1	Open	연속형	숫자	해당 거래일의 최초 거래 시점에 형성된 가격
2	High	연속형	숫자	해당 거래일 동안 기록된 가장 높은 가격
3	Low	연속형	숫자	해당 거래일 동안 기록된 가장 낮은 가격
4	Close	연속형	숫자	거래일의 마지막 거래 시점에 형성된 가격
5	Volume	연속형	숫자	해당 거래일 동안 거래된 주식의 총량을 나타내는 값
6	RSI	연속형	숫자	주식의 과매수 상태를 나타내는 값
7	RSISignal	연속형	숫자	RSI 지표에서 파생되는 신호로 70을 돌파하여 하락하는 경우, 판매 신호로 간주
8	CCI	연속형	숫자	양수일수록 상승추세, 음수일수록 하락추세인 값

데이터 분석

인덱스	이름	데이터타입	값	설명
9	CCISignal	연속형	숫자	값이 +100 이상인 경우 과매수 상태, -100 이하인 경우 과매도 상태
10	UpDI	연속형	숫자	DMI중 하나로, 가격이 이전 거래일보다 상승한 경우, 증가하는 값
11	DownDI	연속형	숫자	DMI중 하나로, 가격이 이전 거래일보다 하락한 경우 증가하는 값
12	ADX	연속형	숫자	현재 주식 추세의 강도를 나타내는 값
13	PDI	연속형	숫자	DMI중 하나로, 상승 추세의 강도가 클수록 높은 값
14	MDI	연속형	숫자	DMI중 하나로, 하락 추세의 강도가 클수록 높은 값
15	WilliamsR	연속형	숫자	과매수 또는 과매도 상태에 있는지를 판단하는 값으로 -20 이하면 과매도
16	WilliamsD	연속형	숫자	WilliamsR 지표를 기반으로 한 주가의 상대강도를 나타내는 지표값
17	MACD	연속형	숫자	가격의 상승 또는 하락 추세가 강할수록 증하가는 값
18	MACDSignal	연속형	숫자	주식 가격의 상승이나 강세가 클수록 증가하는 값

데이터 분석

인덱스	이름	데이터타입	값	설명
19	MACDOscillator	연속형	숫자	값이 크고 양수인 경우 강한 상승 추세, 값이 크고 음수인 경우 강한 하락 추세
20	Close5ma	연속형	숫자	종가 5일 이동평균 값
21	Close10ma	연속형	숫자	종가 10일 이동평균 값
22	Close20ma	연속형	숫자	종가 20일 이동평균 값
23	Close60ma	연속형	숫자	종가 60일 이동평균 값
24	INDNB	연속형	숫자	지정 종목의 개인순매수 값
25	INSNB	연속형	숫자	지정 종목의 기관순매수 값
26	FNB	연속형	숫자	외국인 순매수/순매도 값
27	Call	연속형	숫자	금리
29	EX	연속형	숫자	환율(원/달러)
30	SSR	연속형	숫자	공매도 비율

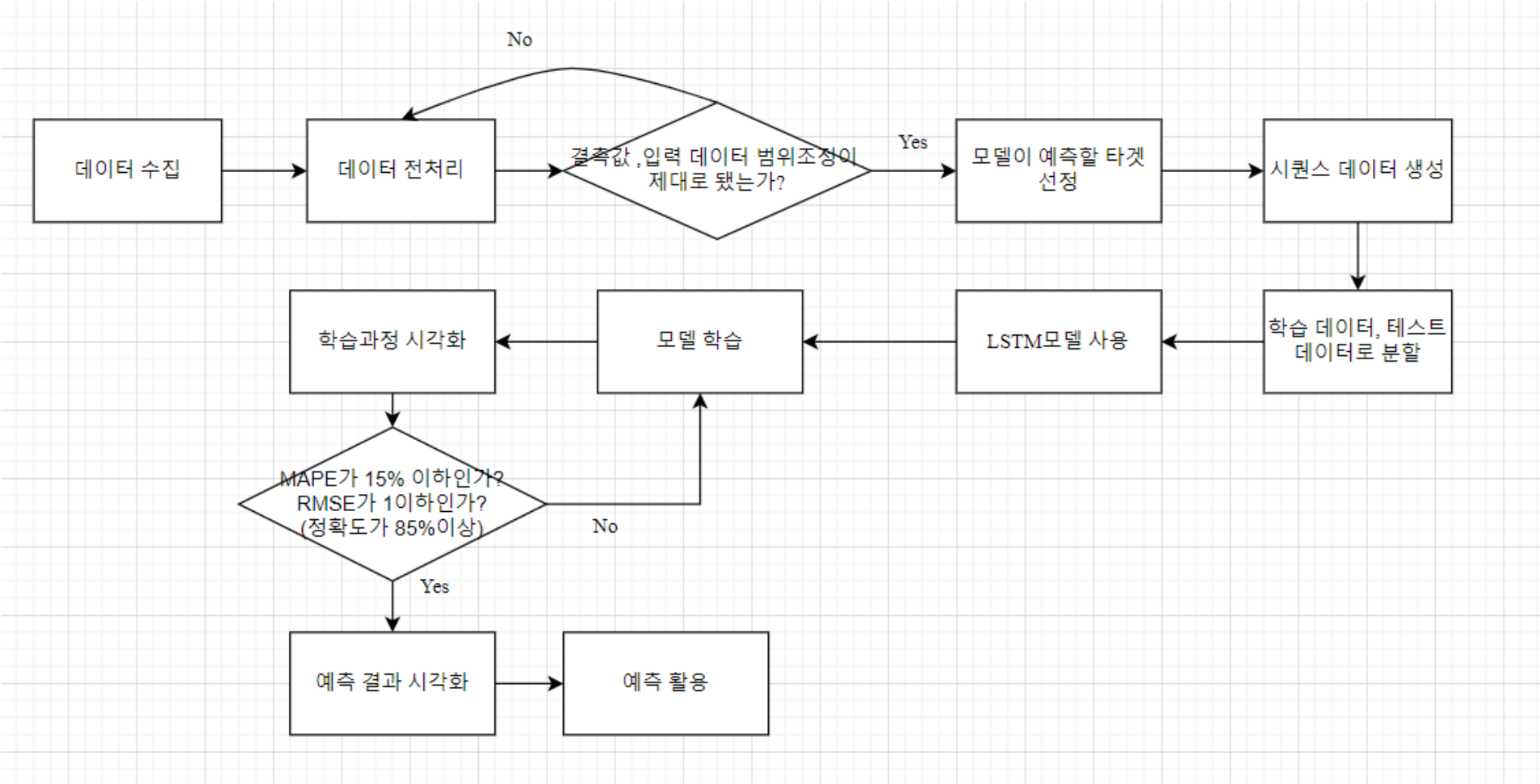
사용 알고리즘 : LSTM

- RNN의 한 종류로, 시간적 순서를 가진 시계열 데이터와 같은 처리하는 데 사용되는 신경망 구조
- 과거의 데이터를 토대로 현재(미래)를 예측하는 곳에 사용
- RNN은 반복 모듈을 통해 시간에 따른 의존성을 학습하지만 장기 의존성 문제로 인해 긴 시퀀스 데이터에서 오랜 기간의 의존성을 제대로 학습하기 어려운 문제를 가짐
- 과거의 정보를 기억하고 유지하는 역할인 기억 셀(Memory Cell), 시그모이드 활성화 함수와 곱셈 연산을 통해 기억 셀의 동작을 조절하는 게이트 메커니즘을 도입

  ➡ RNN의 장기 의존성 문제를 해결

- 장점 : 시계열 데이터에서 더 긴 의존성을 파악할 수 있고, 선택적으로 정보를 저장하고 삭제하면서 필요한 정보만을 유지하고 전달
- 단점 : 상대적으로 처리 속도가 느리고 과적합 가능성이 높아짐

전체 순서도


No
데이터 수집
데이터 전처리
결측값 ,입력 데이터 범위조정이 제대로 됐는가?
Yes
모델이 예측할 타겟 선정
시퀀스 데이터 생성
학습 데이터, 테스트 데이터로 분할
LSTM모델 사용
모델 학습
학습과정 시각화
MAPE가 15% 이하인가?
RMSE가 1이하인가?
(정확도가 85%이상)
No
Yes
예측 결과 시각화
예측 활용

전처리

```
price_indicator = df.loc[:,'Open':'Close'].values[1:].astype(float) # 가격 관련 지표
volume_indicator = df.loc[:,'Volume':'Volume'].values[1:].astype(float) # 거래량 관련 지표
etc_indicator = df.loc[:,'RSI':'SSR'].values[1:].astype(float) # 추세 또는 거래량 활용 지표

scaler = MinMaxScaler(feature_range=(0, 1)) # 0~1 값으로 스케일링
scaler_etc = MinMaxScaler(feature_range=(-1, 1)) # -1~1 값으로 스케일링

scaled_price_indicator = scaler.fit_transform(price_indicator) # 가격 관련 지표에 스케일링
scaled_volume_indicator = scaler.fit_transform(volume_indicator) # 거래량 관련 지표에 스케일링
scaled_etc_indicator = scaler_etc.fit_transform(etc_indicator) # 추세 또는 거래량 활용 지표에 스케일링
price_indicator
```

```
array([[ 26900.,  27100.,  26400.,  26650.],
       [ 26800.,  26900.,  26200.,  26350.],
       [ 26400.,  26450.,  25650.,  25900.],
       ...,
       [106000., 110500., 106000., 110300.],
       [111000., 112700., 109200., 110300.],
       [110100., 110500., 108500., 108700.]])
```

모델학습

```
input_shape = (seq_length, trainX.shape[2])

model = Sequential()
model.add(LSTM(64, input_shape=input_shape, return_sequences=True))
model.add(Dropout(0.001))
model.add(LSTM(64))
model.add(Dropout(0.001))
model.add(Dense(1))
model.add(Activation('softsign')) #linear / softsign
print(input_shape)
```

```
(29, 29)
```

```
adam = optimizers.Adam(learning_rate=0.005)
model.compile(loss='mean_squared_error', optimizer=adam)
model.summary()

model_checkpoint = ModelCheckpoint('best_model.h5', monitor='val_loss', save_best_only=True)


# 모델 학습
history = model.fit(trainX, trainY, epochs=10, batch_size=64,
                    verbose=1, validation_data=(testX, testY))
```

구현 : 사용자 입력

```
df= pd.read_csv(os.path.join("C:₩ml_workspace₩SK하이닉스 13~23.csv"), encoding='utf8')
df.describe()
```

Out[48]:

	Open	Higt	Low	Close	Volume	RSI	RSISignal	CCI	CCISignal	UpDI	...	C
count	2566.000000	2566.000000	2566.000000	2566.000000	2.566000e+03	2566.000000	2566.000000	2566.000000	2566.000000	2566.000000	...	256
mean	68822.798129	69714.127046	67836.730320	68737.996882	3.723025e+06	0.519573	0.519210	8.346485	8.160978	24.632529	...	10
std	30681.834072	31095.626517	30178.490951	30629.049394	1.960378e+06	0.159519	0.131400	107.891457	77.176467	10.247633	...	
min	23850.000000	24200.000000	23600.000000	23700.000000	8.425620e+05	0.066300	0.137700	-276.060000	-158.160000	1.430000	...	8
25%	41000.000000	41350.000000	40350.000000	40900.000000	2.442582e+06	0.405450	0.425975	-76.547500	-59.485000	17.350000	...	9
50%	71800.000000	73000.000000	70800.000000	71750.000000	3.258910e+06	0.515400	0.513750	13.300000	10.525000	23.590000	...	10
75%	87900.000000	89000.000000	86500.000000	87875.000000	4.465971e+06	0.634575	0.614100	90.610000	75.535000	30.812500	...	10
max	149000.000000	150500.000000	142500.000000	148500.000000	2.101944e+07	0.972600	0.902800	361.630000	168.680000	68.210000	...	11

8 rows × 29 columns

구현 : 자격증명 확인

In [69]:
```
open_value_test = df.loc[df['Open'] > 40000]
open_value_test
```

Out[69]:

	Date	Open	Higt	Low	Close	Volume	RSI	RSISignal	CCI	CCISignal	...	Close5ma	Close10ma	Close20ma	Close60ma	INDNB	
324	2014-04-23	40200	40850	40050	40750	6302966	0.8088	0.6411	170.66	137.01	...	103.32	106.15	108.10	106.87	-3253028	8
325	2014-04-24	40850	40900	40150	40900	4202633	0.7920	0.6713	134.90	136.59	...	102.22	105.66	108.04	107.07	-1374110	1
326	2014-04-25	41100	41350	40650	40750	6413507	0.8049	0.6980	120.38	133.34	...	101.07	104.27	107.12	106.55	-1091322	-18
327	2014-04-28	40450	40800	39900	40500	2962593	0.8049	0.7194	80.43	122.76	...	99.90	102.69	105.84	105.77	-299651	
328	2014-04-29	40250	40500	39600	39700	2864162	0.6694	0.7094	49.35	108.08	...	97.98	100.08	103.39	103.54	442888	-3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2561	2023-05-30	113200	113400	109500	110300	9058509	0.9167	0.7677	160.56	137.10	...	106.34	111.41	117.90	123.97	-625685	-6

성능평가 : RMSE

```
: from sklearn.metrics import mean_squared_error, mean_absolute_error

# 예측 수행
y_pred = model.predict(testX)

# RMSE 계산
rmse = np.sqrt(mean_squared_error(testY, y_pred))
print("Root Mean Squared Error (RMSE):", rmse)

# 평균 제곱근 오차 (RMSE)예측값과 실제값의 차이의 크기
```

```
24/24 [==============================] - 1s 21ms/step
Root Mean Squared Error (RMSE): 0.1183605930153349
```

성능평가 : MAPE

```python
def calculate_mape(testY, y_pred):
    actual_values = np.array(testY)
    predicted_values = np.array(y_pred)

    # Calculate absolute percentage errors
    absolute_errors = np.abs((testY - y_pred) / testY)

    # Calculate mean absolute percentage error
    mape = np.mean(absolute_errors) * 100

    return mape

mape = calculate_mape(testY, y_pred)
print(f"MAPE: {mape}%")

#예측 값과 실제 값 사이의 절대 오차의 평균을 계산
#정확도는 87.45043106277341%
```

```
MAPE: 12.746287935897483%
```

결론

- 요약 : LSTM 알고리즘을 사용하여 더 다양한 종류의 주식 관련 키워드 뿐만 아니라 금리나 환율과 같은 주식 가격을 결정하는 외부 키워드 또한 학습시켜 정확도를 높였다.
- 극복한 점 : 초반에 학습을 시키면서 과소적합문제가 계속 발생하여 고통을 받았으나, 학습할 데이터양이 너무 적은 것 같아 최근 10년으로 범위를 늘린 결과 문제를 해결했다. 또한 이 다음에는 과적합 문제가 발생해서 또 다시 고통이 찾아왔지만 적절한 에포크 횟수 조절과 Dropout 레이어를 추가하여 해결할 수 있었다. 수업 시간에 배운 기본이 정말 중요하고 포기하지 않고 계속해서 여러 시도를 하면서 이겨내려는 자세라면 못할 것이 없다는 것을 깨달았다.
- Future work
  - 금리나 환율 이외에도 전날 뉴스에서 많이 언급된 단어, 전날 나스닥지수 증감량, 관련 업종 주가 추이 등과 같은 키워드들도 더 많이 수집하여 추가할 것이다.
  - Transformer, GANs과 같은 다른 모델도 사용해서 LSTM모델과 성능이 얼마나 차이나는지 비교하여 주식예측과 관련된 더 적합하고 정확한 모델이 무엇인지 알아볼 것이다.