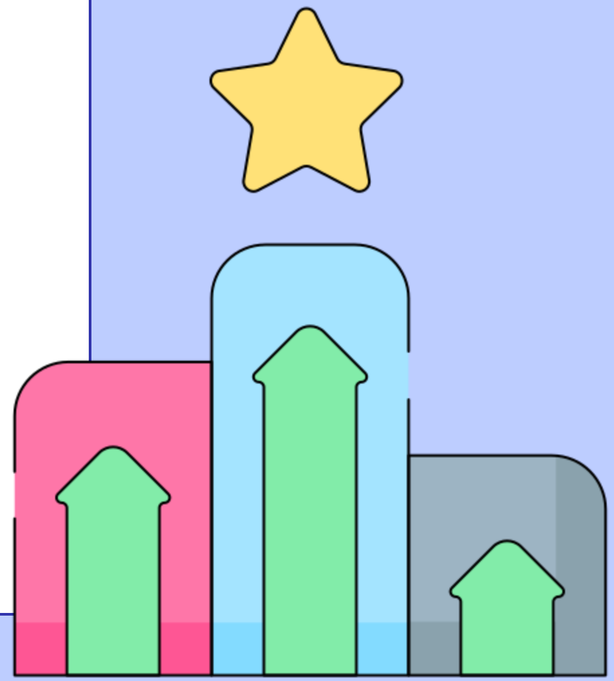
MLP, LSTM, Transformer를 활용한

주가 예측 프로젝트

Transformer와 기존 모델의 주가 예측 비교



팀장 : 이지아

팀원 : 1명(이름 미공개)

목차

연구 배경

Artificial Intelligence

주식 시장의 변화와 딥러닝의 필요성

- 2030 세대의 투자 증가와 증시 불안정성으로 인해 주가 예측의 중요성 부각
- 복잡한 패턴 처리와 비선형 문제 해결에 강점을 지닌 딥러닝 기술 주목

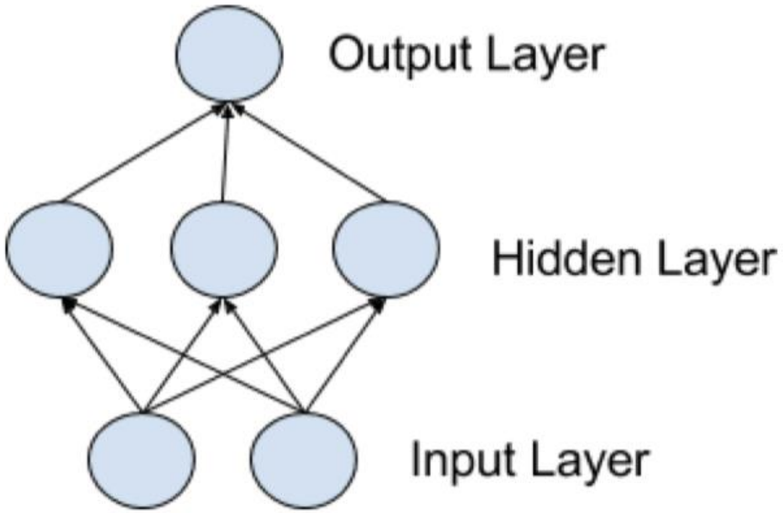
#머신러닝 #딥러닝 #주가예측

모델 개요

MLP

다층 신경망으로 비선형 문제 해결

- 구조 : 입력층과 출력층 그리고 은닉층으로 구성된 기본 신경망
- 활용 : 다양한 예측 문제에서 입력과 출력의 노드 수와 은닉층 구조 조정 가능
- 학습 : 오차 역전파를 통해 가중치를 조정하여 최적화된 모델 도출



#Backpropagation #MLP

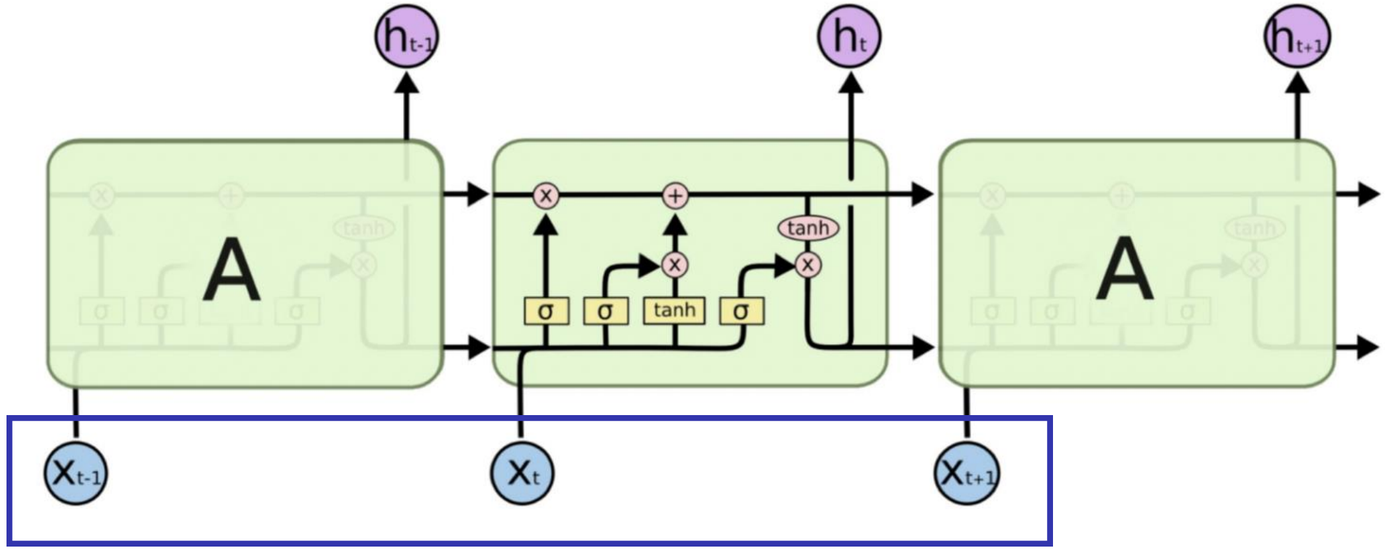
모델 개요

Artificial Intelligence

LSTM

장기 의존성 문제 해결을 위한 RNN 구조

- 구조 : Forget, Input, Output 게이트로 선택적 정보 저장과 갱신
- 특징 : 과거 데이터를 장기적으로 기억해 시계열 데이터 처리에 강점
- 학습 : 오차 역전파와 시간에 따른 역전파를 통해 과거와 현재 정보를 학습하고 장기 의존성을 고려해 가중치 조정

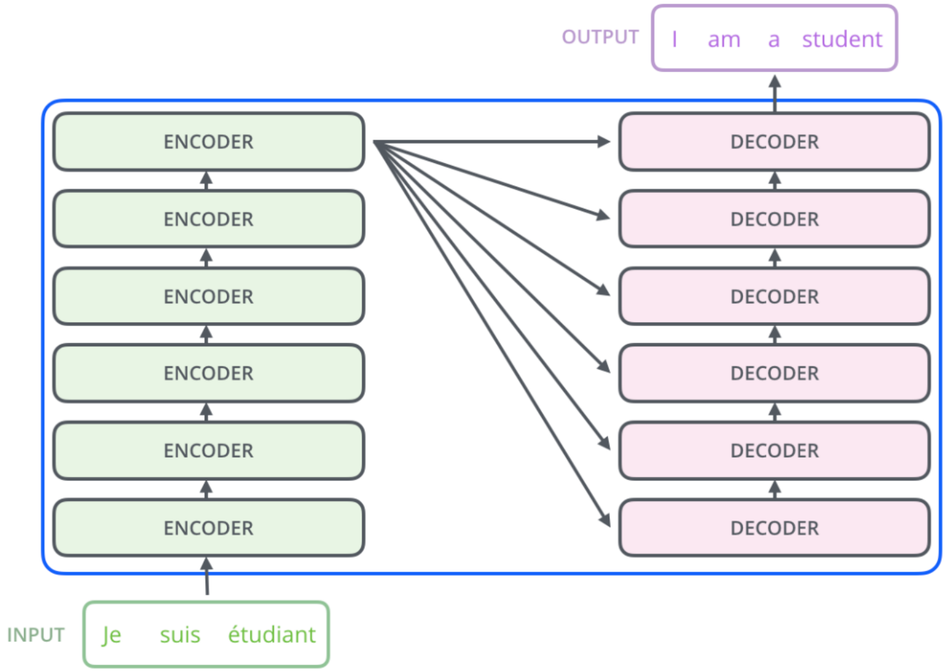


모델 개요

Transformer

자연어 처리에서 시작해 다양한 예측 문제에 적용

- **특징 : Self-Attention과 병렬 처리로 긴 시퀀스 데이터에 강점**
- **활용 : 주가 예측과 같은 복잡한 패턴을 요구하는 문제에서 뛰어난 성능**

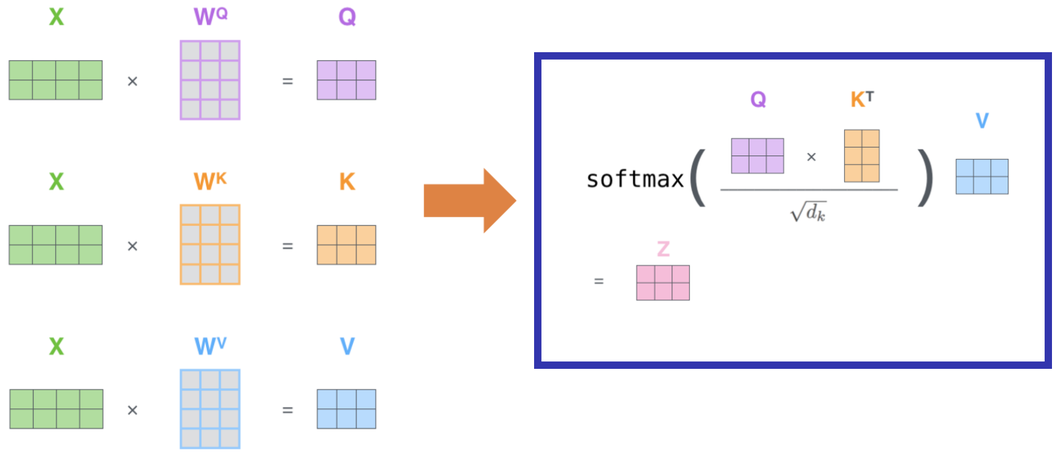


모델 개요

Transformer’s Self-Attention

데이터 간의 관계를 고려한 가중치 부여

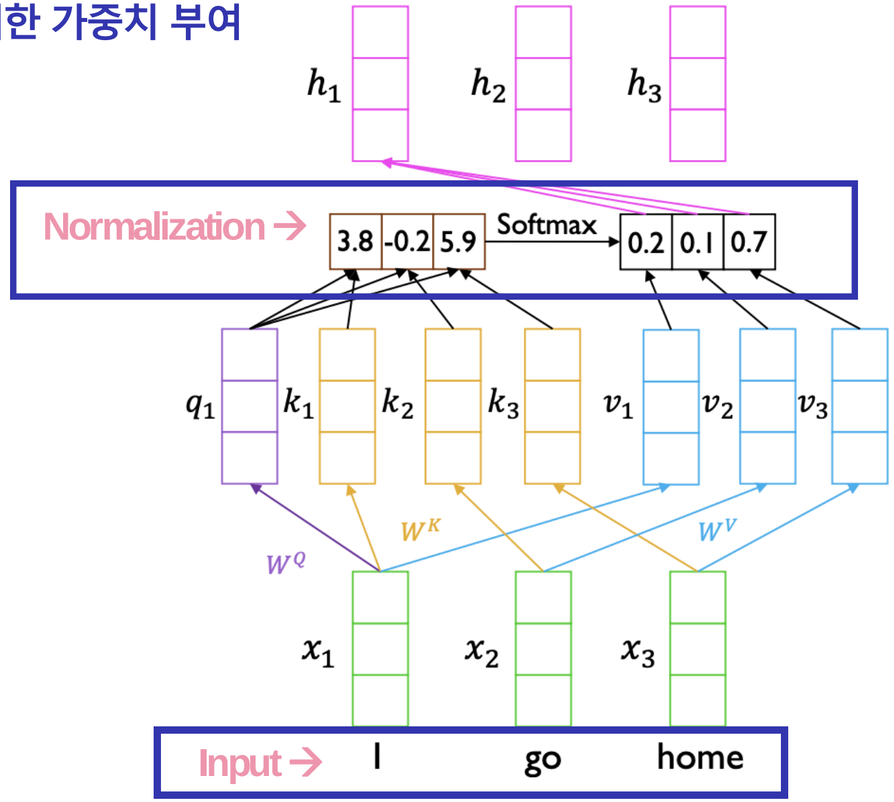
- 쿼리(Q), 키(K), 벨류(V) : 입력 간의 유사도 계산에 사용되는 벡터
- 가중치 계산 : SoftMax 함수를 통해 각 요소의 중요도 반영
- 장점 : 모든 시점의 정보를 동시에 처리하여 효율적인 데이터 학습

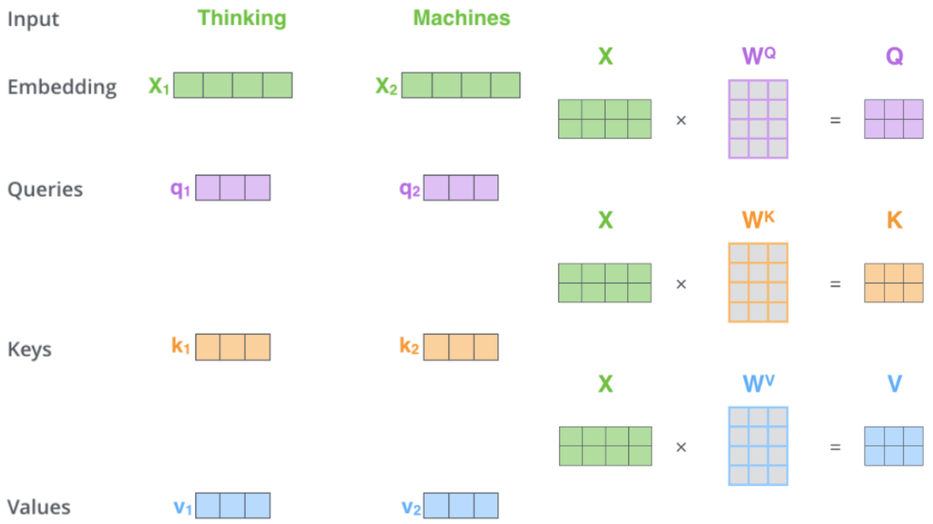


모델 개요

Transformer's Self-Attention

데이터 간의 관계를 고려한 가중치 부여



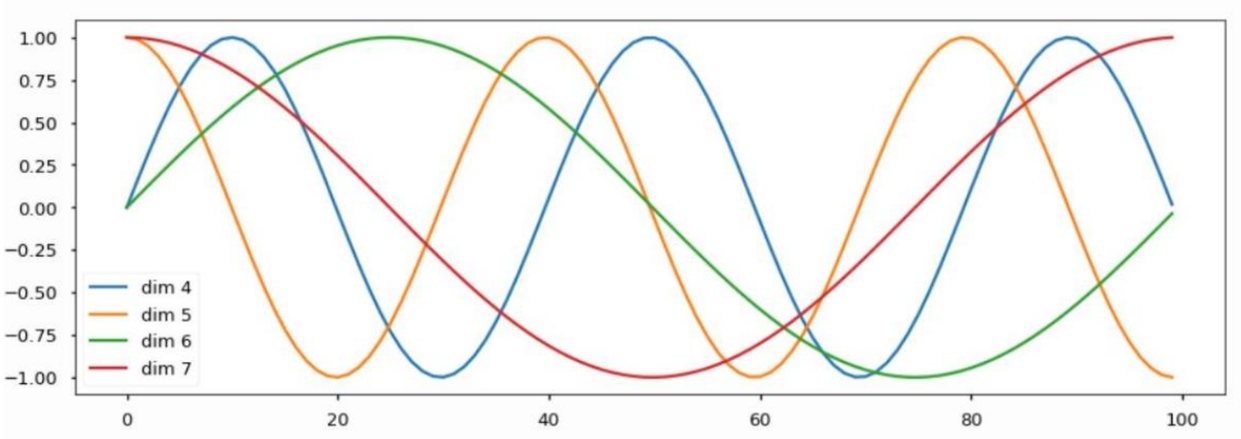


모델 개요

Transformer’s Self-Attention

데이터 간의 관계를 고려한 가중치 부여

- **Self-Attention은 순서를 고려하지 않기 때문에, Positional Encoding을 사용해 시퀀스 내 순서 정보를 보완**

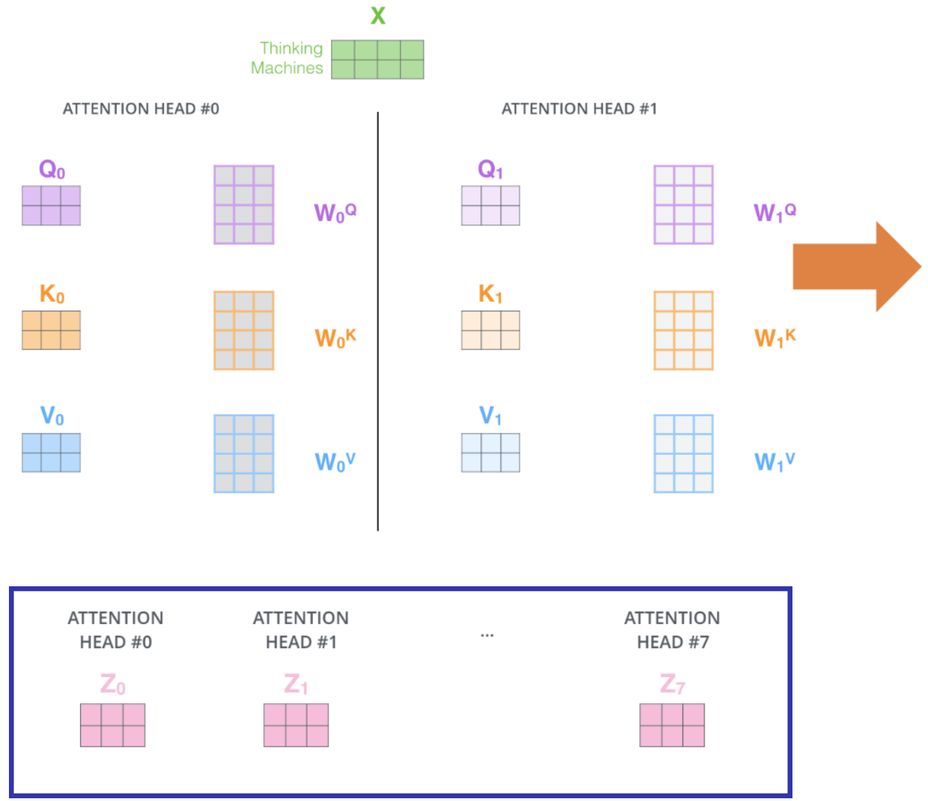


모델 개요

Transformer’s Multi-Head Attention

다양한 정보 패턴을 동시에 학습

- 다중 Attention Head : 각 Head는 다른 정보에 집중해 종합적으로 학습
- 정보 통합 : 각 Head별 출력을 결합해 다양한 패턴을 반영
- 장점 : 단일 Attention보다 풍부한 표현력 제공

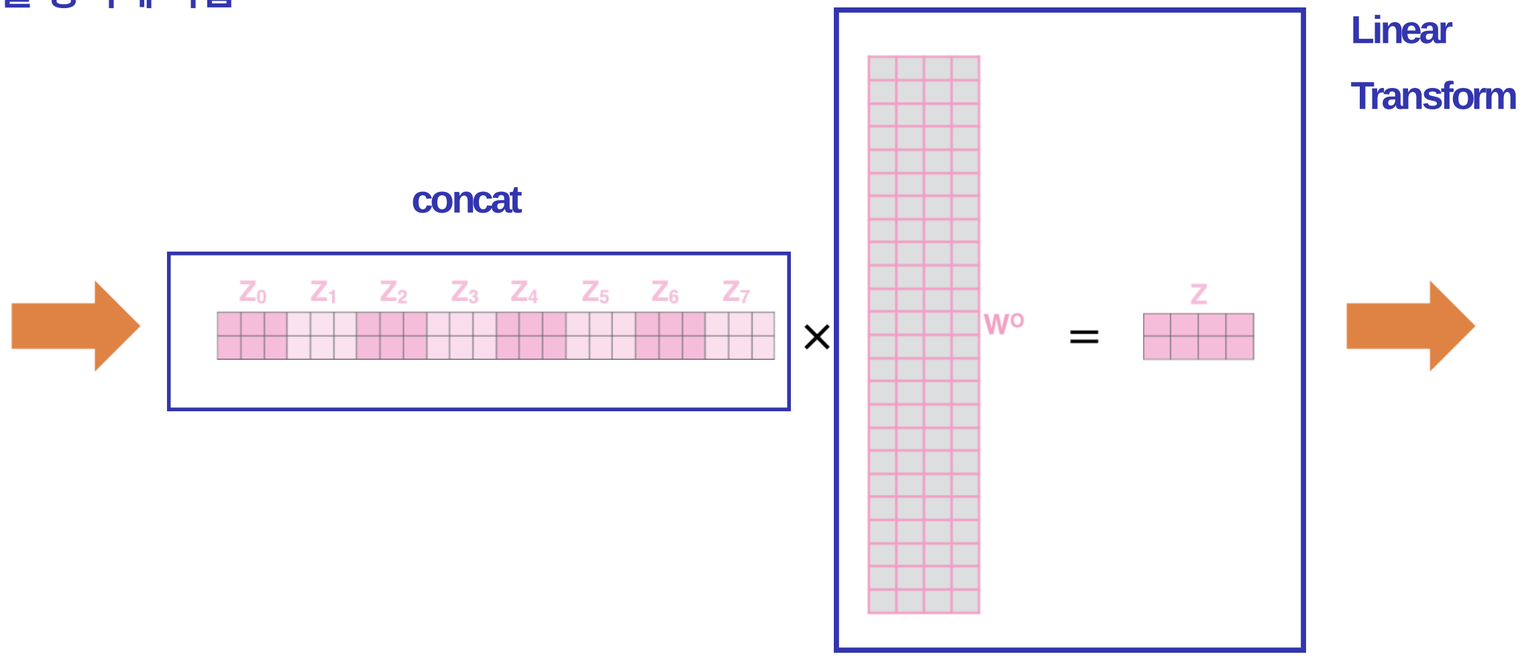


모델 개요

Artificial Intelligence

Transformer’s Multi-Head Attention

다양한 정보 패턴을 동시에 학습



모델 개요

Transformer's 주가 예측에서의 장점

기존 모델 대비 성능 우수

- 병렬 처리로 학습 속도 향상 : MLP, RNN, LSTM에 비해 효율적인 학습
- 긴 시퀀스에서 우수한 성능 : 장기 의존성 문제 해결
- 주가 예측에서 활용 : 복잡한 패턴과 변동성을 효과적으로 학습

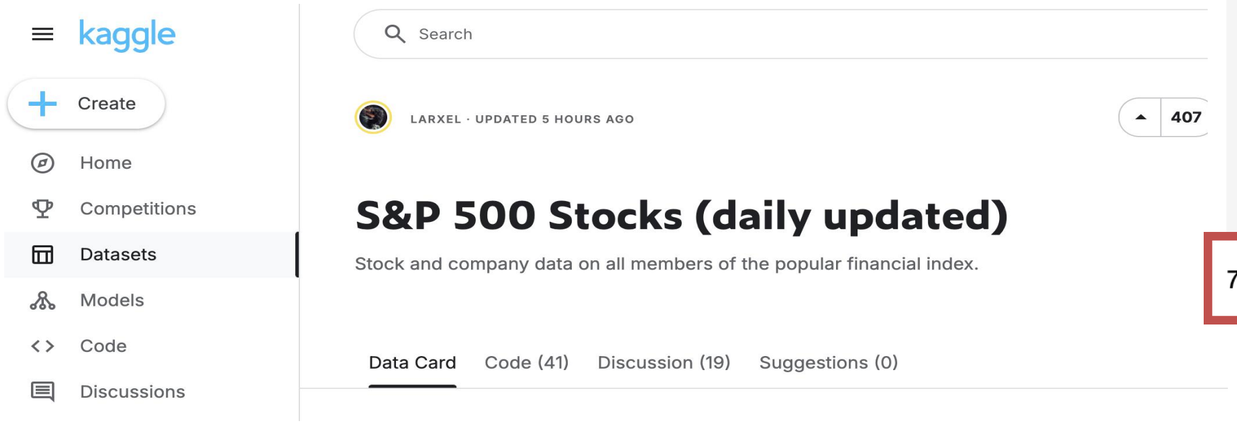
#Transformer #장기의존성

실험 설정

Artificial Intelligence

데이터셋

- 데이터 출처 : kaggle의 S&P 500 데이터와 yfinance 수치 데이터
- 데이터 설정 : 과거 10일 데이터를 기반으로 예측 입력 설정
- 기술적 지표 : MACD, RSI, BB, PSAR, ROC



name	date	adj_close	close	high	low	open	volume
A	2018-01-02	64.401237	67.599998	67.889999	67.339996	67.419998	1047800.0
	2018-01-03	66.039841	69.320000	69.489998	67.599998	67.620003	1698900.0
	2018-01-04	65.544449	68.800003	69.820000	68.779999	69.540001	2230700.0
	2018-01-05	66.592400	69.900002	70.099998	68.730003	68.730003	1632500.0
	2018-01-08	66.735283	70.050003	70.330002	69.550003	69.730003	1613400.0
...	...	...	...	...	...	...	...
ZTS	2023-12-22	193.067841	194.979996	195.910004	192.740005	195.320007	1548400.0
	2023-12-26	193.582718	195.500000	196.339996	194.089996	194.880005	814600.0
	2023-12-27	194.968979	196.899994	197.009995	194.740005	195.410004	766400.0
	2023-12-28	195.226440	197.160004	198.600006	196.529999	197.619995	880100.0
	2023-12-29	195.434387	197.369995	198.009995	196.250000	196.679993	1007200.0

744175 rows × 6 columns

실험 설정

Artificial Intelligence

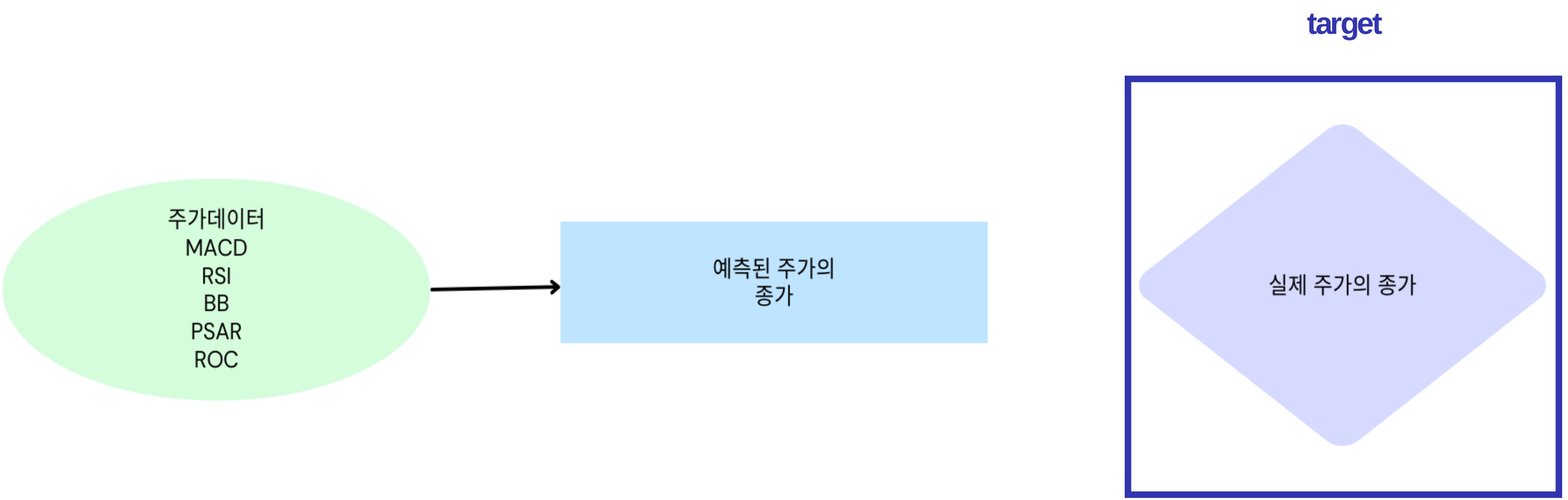
데이터셋

```python
# MACD
def calculate_moving_average(df, window=14):
    return df['close'].rolling(window=window).mean()

# RSI
def calculate_rsi(df, window=14):
    delta = df['close'].diff(1)
    gain = delta.where(delta > 0, 0)
    loss = -delta.where(delta < 0, 0)

    avg_gain = gain.rolling(window=window).mean()
    avg_loss = loss.rolling(window=window).mean()

    rs = avg_gain / avg_loss
    rsi = 100 - (100 / (1 + rs))
    return rsi

# Bollinger Bands
def calculate_bollinger_bands(df, window=20, num_std_dev=2):
    rolling_mean = df['close'].rolling(window=window).mean()
    rolling_std = df['close'].rolling(window=window).std()
    upper_band = rolling_mean + (rolling_std * num_std_dev)
    lower_band = rolling_mean - (rolling_std * num_std_dev)
    return upper_band, lower_band
```

실험 설정

Artificial Intelligence

데이터셋

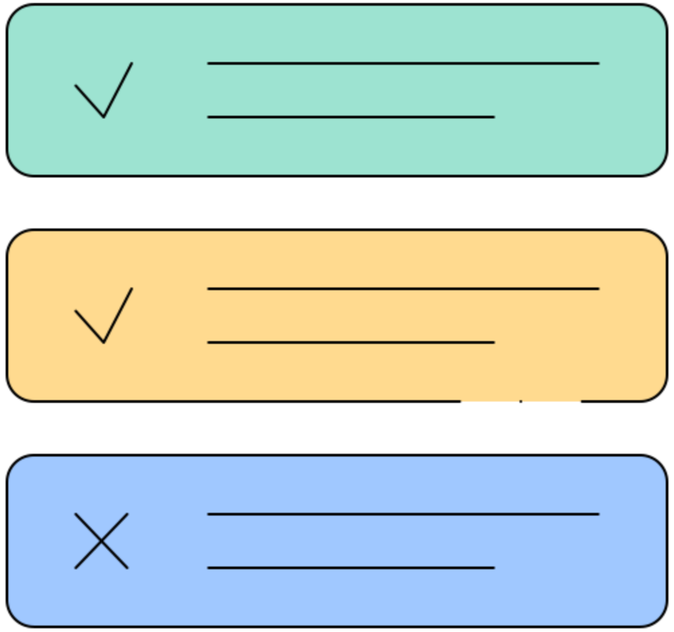


실험 설정

Artificial Intelligence

모델 학습 설정

- 동일한 학습률, 에포크 수, 배치 크기 설정
- Adam Optimizer 사용으로 일관된 최적화 수행
- PyTorch로 모델 구현 및 학습 진행



실험 설정

Artificial Intelligence

모델 학습 설정

```
# MLP 모델 정의
model = MLP_N(input_size, hidden_size, num_layers, output_s
loss_func = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
```

```
# LSTM 모델 정의
model = LSTM_N(input_size, hidden_size, num_layers, dropout
loss_func = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
```

```
# Transformer 모델 정의
model = TransformerModel(input_size, d_model, nhead, num_er
loss_func = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
```

```
minibatch_size = 128
train_batches = DataLoa
val_batches = DataLoade

nb_epochs = 50
progress_interval = 3
early_stop = 30
```

실험 설정

Artificial Intelligence

모델 성능 평가 지표

- **RMSE : 예측 값과 실제 값 사이의 평균 제곱근 오차**

  -> 작을수록 예측 정확도 높음

- **MDA : 예측 방향과 실제 방향이 일치하는 정도를 평가하는 지표**

  -> 높을수록 예측 정확도 높음

  -> 금융 분야에서 방향성 측정하기 위해 사용

```
Epoch 3: Train Loss = 0.0153, Validation Loss = 0.0112
Epoch 6: Train Loss = 0.0061, Validation Loss = 0.0055
Epoch 9: Train Loss = 0.0049, Validation Loss = 0.0029
Epoch 12: Train Loss = 0.0037, Validation Loss = 0.0018
Epoch 15: Train Loss = 0.0032, Validation Loss = 0.0018
Epoch 18: Train Loss = 0.0030, Validation Loss = 0.0015
Epoch 21: Train Loss = 0.0026, Validation Loss = 0.0019
Epoch 24: Train Loss = 0.0027, Validation Loss = 0.0022
Epoch 27: Train Loss = 0.0028, Validation Loss = 0.0029
Epoch 30: Train Loss = 0.0018, Validation Loss = 0.0017
Epoch 33: Train Loss = 0.0018, Validation Loss = 0.0016
Epoch 36: Train Loss = 0.0023, Validation Loss = 0.0019
Epoch 39: Train Loss = 0.0020, Validation Loss = 0.0029
Epoch 42: Train Loss = 0.0017, Validation Loss = 0.0023
Epoch 45: Train Loss = 0.0016, Validation Loss = 0.0019
Epoch 48: Train Loss = 0.0016, Validation Loss = 0.0017
RMSE for AAPL: 2.59518806393462
MDA for AAPL: 0.49
```

결과 분석

Artificial Intelligence

예측 성능 비교

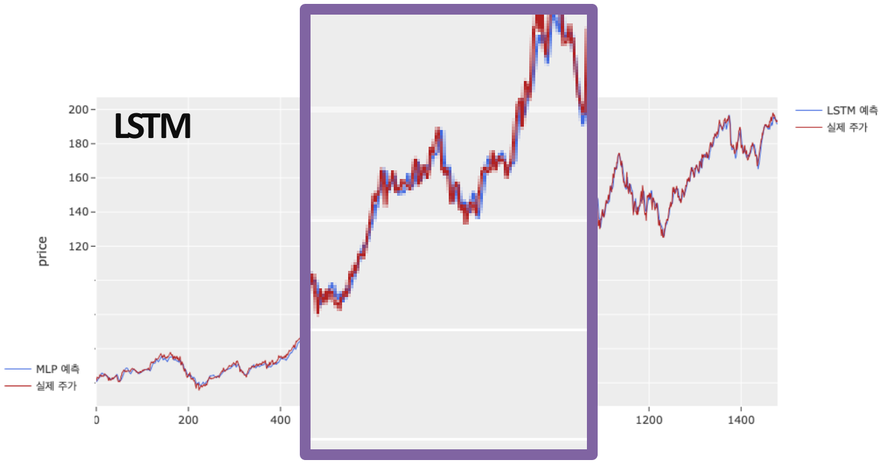
종목	모델	RMSE	MDA
AAPL	MLP	2.627	0.49
	LSTM	2.787	0.50
	Transformer	2.750	0.49
MSFT	MLP	4.471	0.49
	LSTM	4.603	0.49
	Transformer	4.252	0.49
NVDA	MLP	0.746	0.51
	LSTM	0.763	0.50
	Transformer	0.674	0.51

결과 분석

Artificial Intelligence

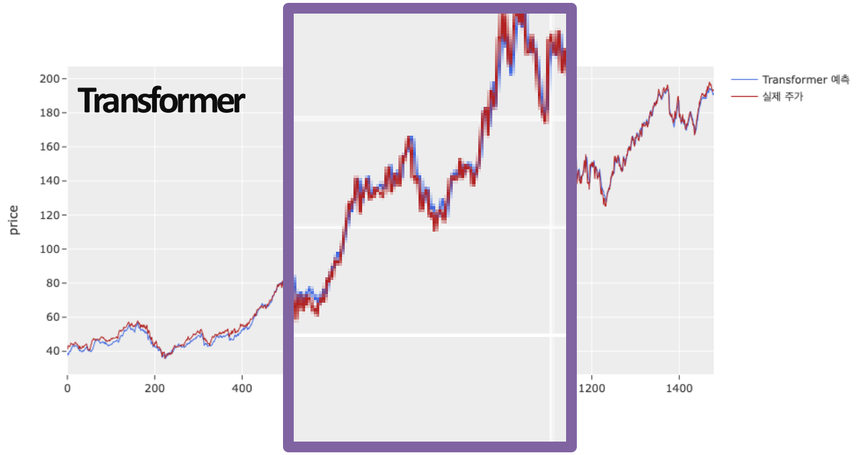
예측 성능 비교

AAPL 종목 예측



예측 주가

실제 주가

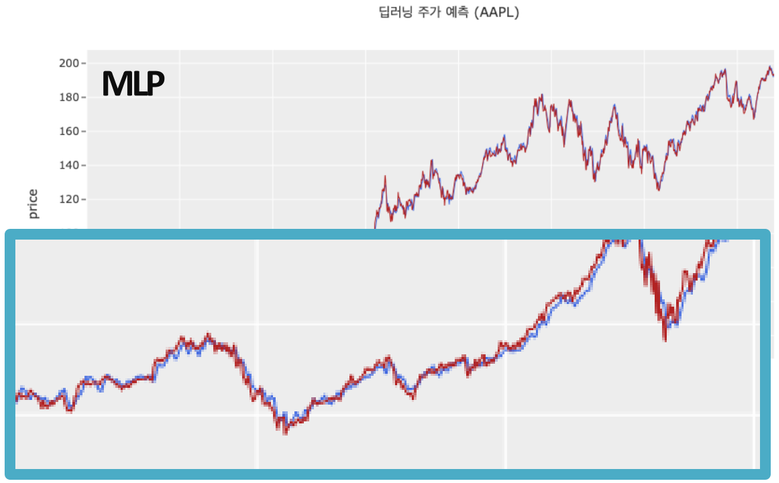


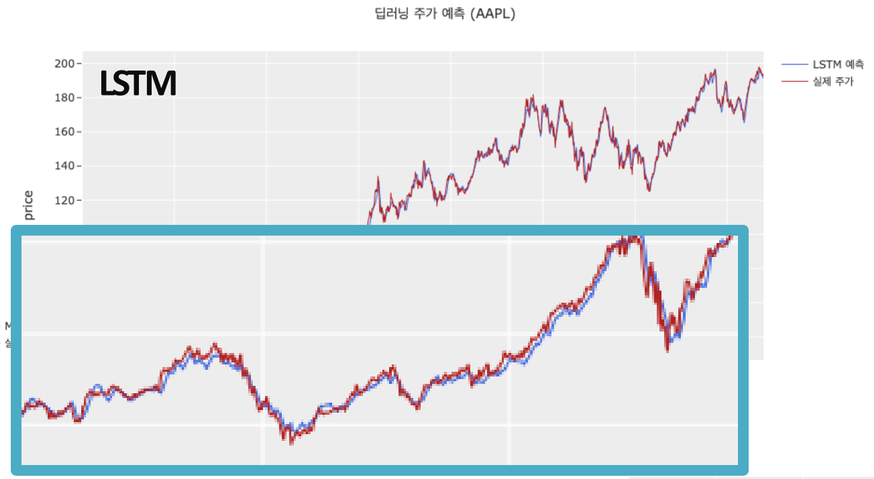
결과 분석

Artificial Intelligence

예측 성능 비교

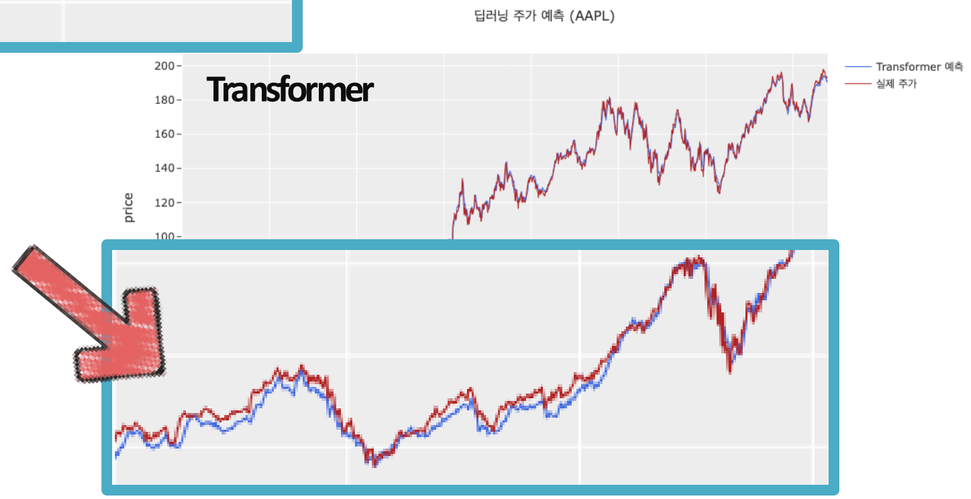
AAPL 종목 예측





예측 주가

실제 주가

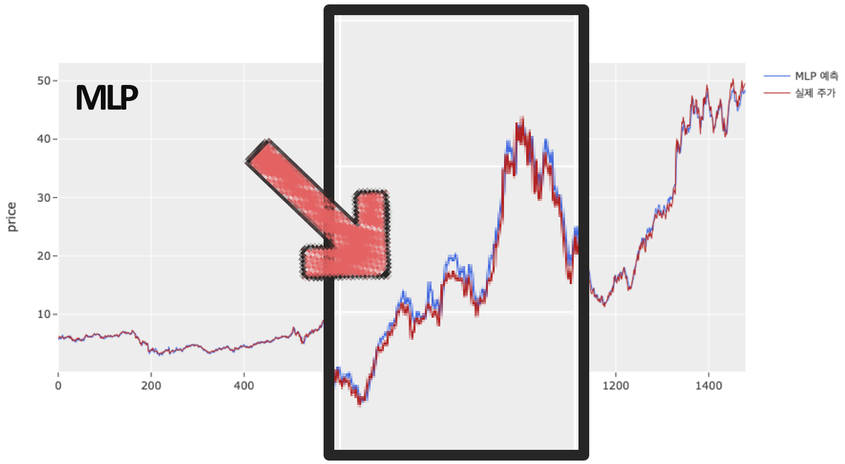


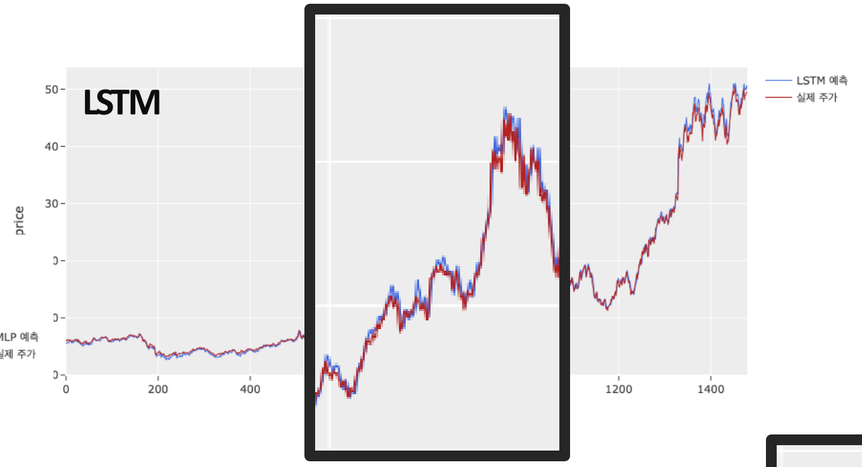
결과 분석

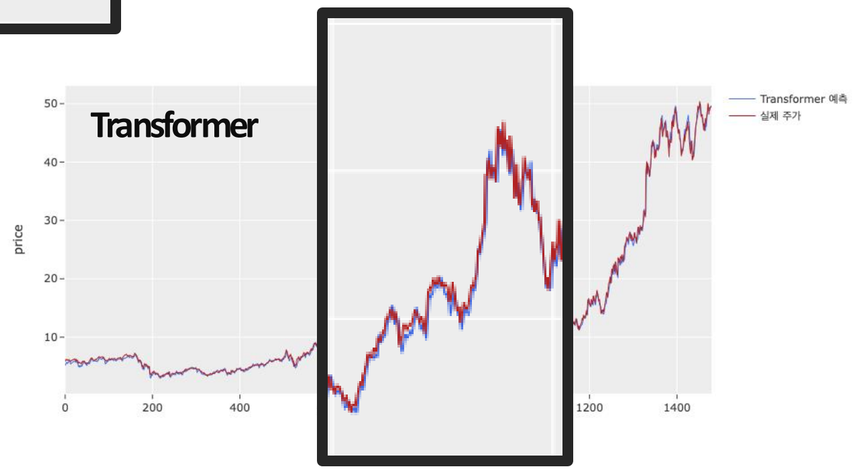
Artificial Intelligence

예측 성능 비교

NVDA 종목 예측







예측 주가

실제 주가

결론과 활용 방안

Artificial Intelligence

결론

MLP	특정 종목에서 비교적 낮은 RMSE를 기록해 간단한 패턴 학습에서 강점
LSTM	시계열 데이터 처리에서 일부 장기 패턴을 잘 학습했으나, 변동성이 큰 구간에서는 Transformer보다 낮은 성능
Transformer	대부분의 종목에서 가장 낮은 RMSE를 보여 복잡한 패턴과 변동성을 잘 학습

결론과 활용 방안

Artificial Intelligence

한계와 개선 방안

DATA 관점

- 데이터셋 크기가 충분하지 않아 모델 간 성능 차이에 제한이 있음
- 외부 변수 반영 부족

HOW?

- 더 큰 데이터셋을 확보하여 학습과 평가에 활용
- 주가 변동에 영향을 미치는 외부 데이터를 추가하여 Feature 생성
- ex) 정책 변화, 경제 지표와 같은 이벤트 데이터

MODELING 관점

- 각 모델별로 하이퍼파라미터 최적화가 필요함

HOW?

- 각 모델에 적합한 하이퍼파라미터를 탐색하고 조정하여 성능 극대화
- ex) 학습률, 배치 크기, 에포크 수 등을 모델 특성에 맞게 최적화

결론과 활용 방안

Artificial Intelligence

주가 예측 결과의 실용적 활용

금융 및 투자 리포트 작성	예측 데이터를 기반으로 맞춤형 리포트 제공
AI 기반 투자 조언	생성형 AI 챗봇에 활용하여 실시간 조언 제공
리스크 관리	시장 변동 리스크 최소화 및 데이터 기반 투자 전략 수립에 기여

감사합니다

www.reallygreatsite.com