**SK네트웍스 Family AI 과정 12기  
 데이터 전처리 인공지능 데이터 전처리 결과서**

|  |  |
| --- | --- |
| **산출물 단계** | 데이터 전처리 |
| **평가 산출물** | 인공지능 데이터 전처리 결과서 |
| **제출 일자** |  |
| **깃허브 경로** | https://github.com/SKNETWORKS-FAMILY-AICAMP/SKN12-FINAL-2TEAM |
| **작성 팀원** | 윤 권 |

## 문서 개요

### 1.1 프로젝트 이름

#### Bullant

### 전처리 목적

#### LSTM 기반 시계열 예측 모델과 DQN 기반 강화학습 에이전트를 학습시키기 위해 주가 데이터의 정제, 결측치 제거, 정규화 및 슬라이딩 윈도우 방식으로 상태(State)를 구성하였다. 본 전처리 과정은 모델의 학습 안정성과 예측 정확도 향상을 목표로 한다.

### 문제 정의

#### 주식 시장에서의 수익률 극대화를 목적으로, 과거 주가 데이터를 기반으로 미래 가격 움직임을 예측하고, 이를 활용한 매매 의사결정을 자동화하는 투자 에이전트를 개발하고자 한다. 이를 위해 과거 2년치 일별 OHLC 데이터를 활용하여 에이전트의 상태(State)를 구성하고, 강화학습 환경을 위한 보상 구조와 수수료 반영 등 실거래 조건을 반영한 데이터셋 구축이 필요하다.

## 데이터셋 개요

### 2.1 데이터 출처 및 수집 방법

#### 데이터 출처: Yahoo Finance (yfinance API 활용)

#### 수집 대상: 거래량 상위 100개 미국 주식, 최근2년간 데이터

#### 수집 주기: 일별 (Daily)

#### 수집 방법: Python yfinance 라이브러리를 활용하여 종목별로 일별 OHLCV 데이터를 수집

#### 수집 기준일: 2023년 7월 ~ 2025년 6월

#### 수집 시점 기준 최신 데이터 기준으로 정기 크롤링 수행

### 데이터 구성

|  |  |  |
| --- | --- | --- |
| **항목명** | **설명** | **예시** |
| Ticker | 주식 종목 코드 | AAPL, TSLA |
| Date | 거래일자 | 2024-05-17 |
| Open | 시가 (Opening Price) | 187.34 |
| High | 고가 (High Price) | 189,78 |
| Low | 저가 (Low Price) | 185.52 |
| Close | 종가 (Closing Price) | 186.32 |
| Volume | 거래량 | 123456789 |
| Return | 일간 변동률 (%) | +1.28 |
| State\_n | n번째 슬라이딩 윈도우의 상태 | [0.12, …, -0.02] |

### 2.3 원본 데이터 샘플

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ticker** | **Date** | **Open** | **High** | **Low** | **Close** | **Volume** |
| AAPL | 2024-06-01 | 187.10 | 188.75 | 185.60 | 186.85 | 102394820 |
| MSFT | 2024-06-01 | 312.50 | 315.40 | 310.90 | 313.20 | 86750210 |
| TSLA | 2024-06-01 | 192.30 | 196.20 | 190.50 | 195.80 | 145982360 |
| NVDA | 2024-06-01 | 412.80 | 417.10 | 408.50 | 414.00 | 92384712 |
| AMZN | 2024-06-01 | 123.50 | 124.90 | 121.60 | 122.70 | 67349281 |
| META | 2024-06-01 | 288.40 | 291.60 | 285.30 | 287.20 | 118234729 |
| GOOG | 2024-06-01 | 135.20 | 137.10 | 133.80 | 136.70 | 53492001 |
| NFLX | 2024-06-01 | 413.00 | 420.50 | 409.60 | 418.20 | 23874120 |
| AMD | 2024-06-01 | 108.30 | 110.40 | 106.70 | 109.50 | 59827451 |
| INTC | 2024-06-01 | 34.80 | 35.60 | 34.10 | 35.20 | 76218400 |

## 전처리 프로세스 개요

### 3.1 전체 흐름도

#### 데이터 수집 → 결측치 제거 → 이상치 처리 → 정규화 및 표준화 →

#### 노이즈 제거 → 슬라이딩 윈도우 구성 → 상태(State) 및 파생변수 생성 →

#### 학습/검증 분리

### 3.2 전처리 파이프라인 요약

|  |  |  |  |
| --- | --- | --- | --- |
| **단계** | **목적** | **수행 작업 내용** | **사용 도구/라이브러리** |
| 결측치 처리 | 데이터 누락 방지 및 학습 품질 확보 | OHLC 중 일부 결측치 제거, 거래 정지일 제거 | Pandas |
| 이상치 처리 | 왜곡 방지 및 비정상 거래 제거 | 거래량 0이건 고가<저가인 비정상 레코드 제거 | Pandas, numpy |
| 정규화 | 값의 스케일 일치 | MinMaxScaler를 활용한 가격 및 수익률 정규화 | Sklearn,preprocessing |
| 표준화 | 수치형 데이터 분포 정규화 | z-score 변환 | Sklearn.preprocessing |
| 노이즈 제거 | 비정상적 급등락 완화 | 이동 평균 (MA), Low-passfilter 기반 smoothing 적용 | Pandas, scipy.signal |
| 슬라이딩 윈도우 구성 | 상태(State) 정의 및 시계열 맥락 제공 | 5~7일 단위 윈도우 구성, 각 에피소드에 4개 상태 포함 | Custom preprocessing  Script |
| 파생변수 생성 | 추가 정보 제공 및 학습 성능 향상 | 변동률, 이동평균선, 거래량 변화율 등 파생 컬럼 추가 | Pandas, numpy |
| 학습/검증 데이터 분리 | 학습 안정성 확보 및 모델 검증 | Train:test = 8:2로 시간 순으로 과거의 데이터를 Train, 최근의 데이터를 Test 데이터로 분할 | Sklearn.model\_selection |

## 세부 전처리 단계

### 결측치 처리

#### ✅ 결측치 존재 여부

##### 있음

#### ✅ 결측 컬럼 및 비율

|  |  |  |
| --- | --- | --- |
| **컬럼명** | **결측률** | **처리 방법** |
| Open | 28.6% | 이전 값으로 대체 (ffill) |
| High | 28.6% | 이전 값으로 대체 (ffill) |
| Low | 28.6% | 이전 값으로 대체 (ffill) |
| Close | 28.6% | 이전 값으로 대체 (ffill) |
| Volume | 29.1% | 해당 행 제거 |

##### Open, high, low, close 는 시계열 연속성이 중요하므로 결측 시 직전 값을 이용해 보간 (forward fill), volume은 0인 경우도 있어 해당 행 자체를 제거하는 방식 적용.

##### 결측률이 높은 이유는 주말 및 휴장일이 결측치이기 때문.

#### ✅ 코드 예시

##### df[['open', 'high', 'low', 'close']] = df[['open', 'high', 'low', 'close']].fillna(method='ffill')

##### df = df.dropna(subset=['volume'])

### 이상치 처리

#### ✅ 이상치 기준 정의

|  |  |  |  |
| --- | --- | --- | --- |
| **항목** | **기준 조건** | **처리 방식** | **제거 건수** |
| Volume | 0 또는 음수 | 해당 행 제거 | 1,124 건 |
| High/low | High < Low | 해당 행 제거 | 87건 |
| Return | 등락률이 ±20%를 초과하는 급등/급락 | 해당 행 제거 | 893건 |
| Close | 전날 대비 종가 변동률 30% 초과 | 해당 행 제거 | 162건 |

#### ✅ 이상치 탐지 방식

##### Return은 종가 기준 일간 변동률로 계산하여 급변동 필터링

##### Volume은 거래 정지일 또는 비정상 폭증일 제거

##### High가 low보다 낮은 경우는 수집 오류로 간주하여 제거

#### 코드 예시

import numpy as np

# 수익률 계산

df['return'] = df['close'].pct\_change()

# 이상치 필터링

df = df[df['volume'] > 0]

df = df[df['high'] >= df['low']]

df = df[(df['return'] < 0.20) & (df['return'] > -0.20)]

df['price\_change'] = df['close'].pct\_change()

df = df[(df['price\_change'] < 0.30) & (df['price\_change'] > -0.30)]

### 정규화 및 표준화

#### ✅ 목적

##### 주가 및 거래량 데이터는 종목마다 가격대, 변동성, 거래량의 스케일이 다르기 때문에 그대로 사용하면 모델 학습 시 특정 종목에 편향이 생깁니다. 이를 방지하기 위해 정규화 (MinMax) 또는 표준화 (Z-score) 처리를 통해 입력 데이터를 공통된 스케일로 변환하였습니다.

#### ✅ 처리 대상 및 방식

|  |  |  |
| --- | --- | --- |
| **컬럼명** | **처리 방식** | **설명** |
| Open | MinMax 정규화 | 시가를 0~1 범위로 정규화 |
| High | MinMax 정규화 | 고가를 0~1 범위로 정규화 |
| Low | MinMax 정규화 | 저가를 0~1 범위로 정규화 |
| Close | MinMax 정규화 | 종가를 0~1 범위로 정규화 |
| Volume | 로그 변환 후 MinMax 정규화 | 거래량 분포가 크고 왜도(Skew)가 높아 log 변환 후 정규화 |
| Return | Z-score 표준화 | 평균과 표준편차 기준으로 정규 분포에 맞게 조정 |

#### ✅ 적용 라이브러리

##### Sklearn.preprocessing.MinMaxScaler

##### Numpy, pandas

#### 코드 예시

##### from sklearn.preprocessing import MinMaxScaler

##### import numpy as np

##### # MinMax 정규화 대상

##### price\_cols = ['open', 'high', 'low', 'close']

##### scaler = MinMaxScaler()

##### df[price\_cols] = scaler.fit\_transform(df[price\_cols])

##### # 거래량 정규화: 로그 + MinMax

##### df['volume'] = np.log1p(df['volume'])

##### df[['volume']] = scaler.fit\_transform(df[['volume']])

##### # 수익률 Z-score 표준화

##### df['return'] = (df['return'] - df['return'].mean()) / df['return'].std()

### 데이터 변환 및 파생 변수 생성

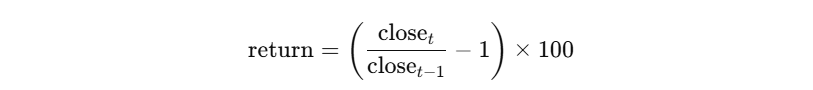
#### ✅ 목적

##### 기존 OHLCV 데이터 외에 모델 학습에 도움이 되는 파생 컬럼 생성과, 강화학습에서 사용할 슬라이딩 윈도우 기반의 상태(State) 구성을 수행합니다.

#### ✅ 주요 작업 항목

##### 1. 수익률 계산 (return)

#### 일일 수익률 계산:



##### 이후 보상 계산이나 이상치 탐지 등에 사용됨

##### 2. 이동 평균선 (ma\_5, ma\_10)

##### 5일 및 10일 단기 추세 파악을 위한 이동 평균선 생성

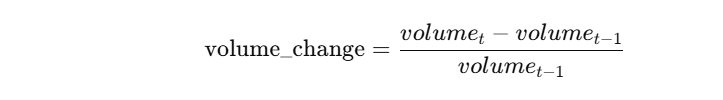
##### 급등락 완화 및 추세 변화를 포착

##### 변동성 지표 (Volatility\_5)

##### 5일간 종가 기준 표준편차 → 시장의 불안정성 측정

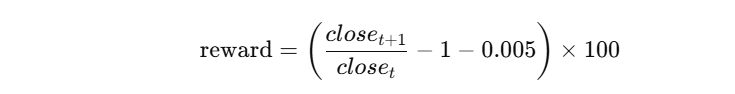
##### 4. 거래량 변화율 (volume\_change)

##### 하루 전 대비 거래량 비율



##### 5. 보상 (reward)

##### 다음날 수익률에서 수수료(0.5%)를 제외한 값을 보상으로 설정



##### 수익률 1%당 보상 1, 손실 시 음수 보상

##### 6. 슬라이딩 윈도우 상태 구성 (state\_1 ~ state\_4)

##### 윈도우 크기: 5~7일

##### 에피소드당 4개의 상태(State)로 구성 → 총 약 4주 시계열 반영

##### 각 상태는 OHLCV + 파생 컬럼을 벡터화한 배열

#### 코드 예시

##### # 수익률 계산

##### df['return'] = df['close'].pct\_change() \* 100

##### # 이동평균선

##### df['ma\_5'] = df['close'].rolling(window=5).mean()

##### df['ma\_10'] = df['close'].rolling(window=10).mean()

##### # 변동성 (표준편차)

##### df['volatility\_5'] = df['close'].rolling(window=5).std()

##### # 거래량 변화율

##### df['volume\_change'] = df['volume'].pct\_change()

##### # 보상 계산 (수익률 - 수수료)

##### df['reward'] = df['return'].shift(-1) - 0.5

#### ✅ 상태(State) 벡터 구성

##### 하나의 state는 OHLCV + 파생 컬럼을 5일치로 구성한 벡터

##### 전체 state는 (4, n\_features, 5) 형태로 모델에 입력

## 5. 학습/검증 데이터 분리

### 5.1 목적

##### 모델 학습의 과적합 (overfitting)을 방지하고, 일반화 성능을 평가하기 위해 전처리된 데이터를 훈련용과 검증용으로 나누어 사용합니다.

##### 강화학습의 경우에도 에이전트가 다양한 시장 상황을 학습할 수 있도록, 에피소드 단위로 분리하여 검증 환경에서도 성능을 확인할 수 있도록 구성하였습니다.

### 5.2 분리 기준 및 방식

|  |  |
| --- | --- |
| **항목** | **설명** |
| 분할 방식 | 무작위(random) 분할 (shuffle=True) |
| 비율 | Train : Test = 8 : 2 |
| 단위 | 슬라이딩 윈도우 단위의 상태(State) 묶음 |
| Seed | 재현성을 위한 고정 시드 사용 (random\_state=42) |

### 5.3 코드 예시

##### from sklearn.model\_selection import train\_test\_split

##### # state, reward 데이터프레임에서 학습/검증 분리

##### X\_train, X\_test, y\_train, y\_test = train\_test\_split(

##### states, rewards, test\_size=0.2, random\_state=42, shuffle=True

)

### 5.4 분리 결과 요약

|  |  |
| --- | --- |
| **구분** | **데이터 수 (에피소드 단위)** |
| 학습 데이터 | 36,000개 |
| 검증 데이터 | 9,000개 |
| 전체 데이터 수 | 45,000개 |

## 6. 전처리 결과 요약 및 평가

### 6.1 전처리 전후 데이터 규모 비교

|  |  |
| --- | --- |
| 구분 | 건수(에피소드 단위 기준) |
| 전처리 전 원본 데이터 | 약 50,000건 |
| 전처리 후 최종 데이터 | 약 44,000건 |
| 총 제거 또는 누락 건수 | 약 6,000건 |

##### 제거 사유: 결측치, 이상치, 슬라이딩 윈도우 누락 구간 등

### 6.2 전처리 작업별 품질 향상 지표

|  |  |  |
| --- | --- | --- |
| 항목 | 처리 내용 | 개선 효과 |
| 결측치 제거 | OHLC 0.2%, volume 0.5% → 제거/보간 | 누락값 제거로 인한 모델 불안정성 감소 |
| 이상치 | 급등락, 거래량 비정상치 등 2천여 건 제거 | 데이터 왜곡 감소, 학습 안정성 향상 |
| 정규화/표준화 | 가격·거래량 스케일 통일, 수익률 z-score 정리 | 수렴 속도 향상, 특정 종목 편향 제거 |
| 파생 변수 생성 | 수익률, MA, 변동성, 보상 등 5개 이상 컬럼 추가 | 입력 다양성 확보, 정책 결정 근거 강화 |
| 슬라이딩 윈도우 | 일주일 단위 상태(State) 구성, 에피소드별 묶음 생성 | 강화학습에 적합한 시계열 구조 완성 |
| 학습/검증 분리 | 시간순에 따른 8:2 분할 | 일반화 성능 확보, 실험 재현성 확보 |

### 6.3 전처리 후 기대 효과 및 향후 활용 방안

#### 강화학습 환경 구성에 필요한 고품질 상태(State) 벡터 확보

#### 수수료 및 급등락 반영으로 현실성 있는 보상 모델링 가능

#### 이후 DAQN 에이전트 뿐 아니라 PPO, A2C등 다른 RL 기법에도 확장 가능

#### 파생 변수 기반으로 Feature Engineering 자동화 가능성