

# UpbitAnalyzer

코인차트지표 분석을 통한 추세 예측 서비스  
FireAnts

강유현, 오세훈, 조영일

# 목차

- 프로젝트 개요
- 프로젝트 팀 구성
- 프로젝트 일정
- 프로젝트 개발환경 및 구조
- 프로젝트 수행 결과
- 자체 평가 의견

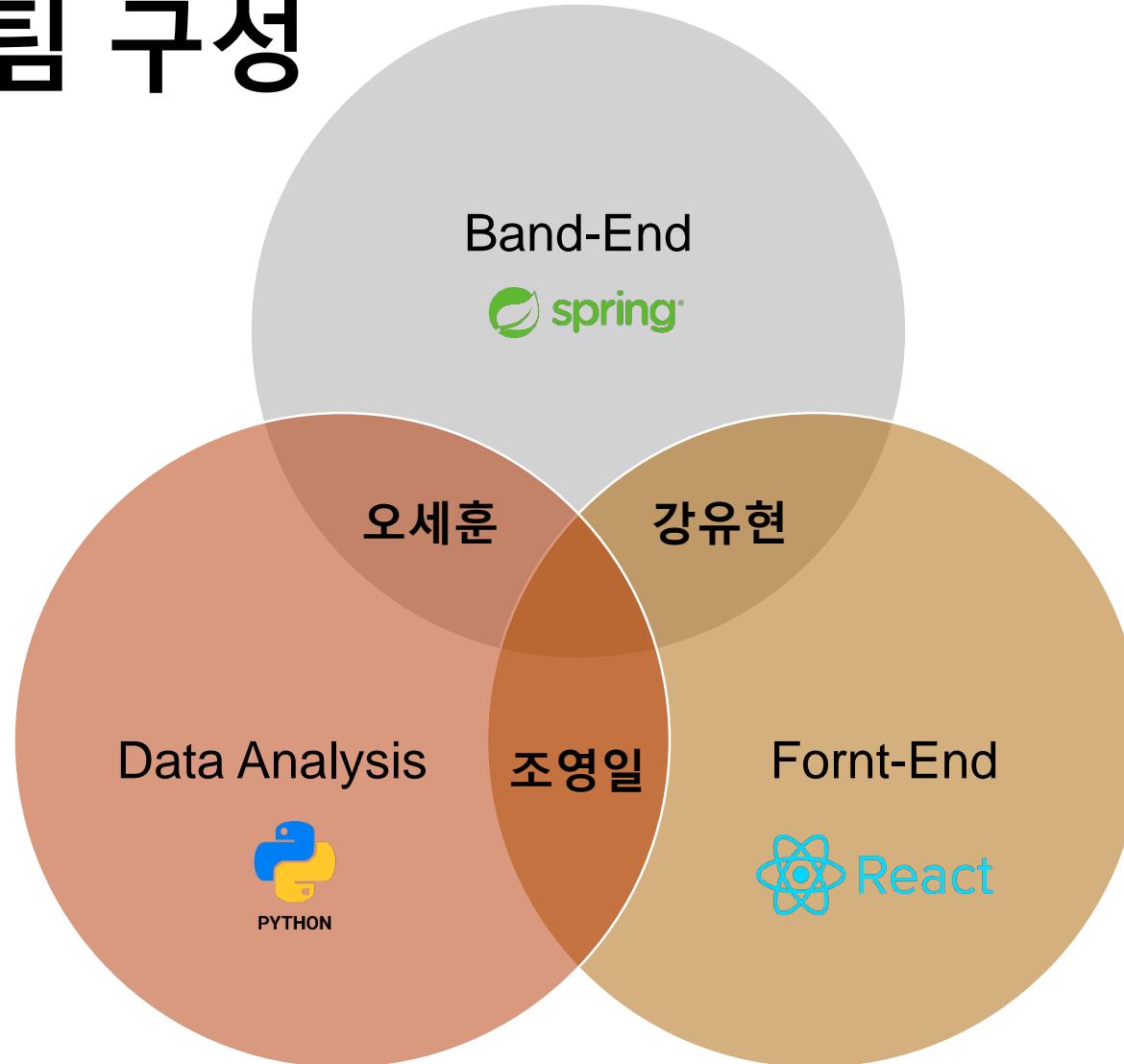
# 프로젝트 개요

- **프로젝트 주제 및 선정 배경**

가상자산 시장에서 코인의 지표 데이터를 분석하여 가격을 예측하고 방향성(상승, 하락)에 대한 확률을 계산하여 퀸트 전략 알고리즘의 기본 토대를 만드는 것을 목표로 합니다.

최근 가상자산 시장의 급격한 성장으로 인해 많은 개인 투자자들이 가상자산에 대한 투자를 시도하고 있지만, 전문적인 지식과 정보 부족으로 인해 많은 투자 실패 사례가 발생하고 있습니다. 이에 본 프로젝트는 가상자산 시장에 대한 접근성을 높이고, 지표 분석과 예측 즉, 딥러닝 솔루션 중 CNN과 LSTM을 활용하여 코인의 지표 데이터를 분석하고 예측 모델을 구축함으로 가상자산에 대한 투자에 대한 전문성과 정보를 제공합니다.

# 프로젝트 팀 구성



# 프로젝트 일정

04-25

개요 작성

계획서 작성

명세서 작성

05-23

Back-end

DB 설계

Rest Api 설계

인증&인가(JWT)

06-20

발표 자료

PPT 작성

결과물 정리 및 확인

05-09

Data Analysis

데이터 수집

모델 설정

분석 및 예측

06-06

Front-end

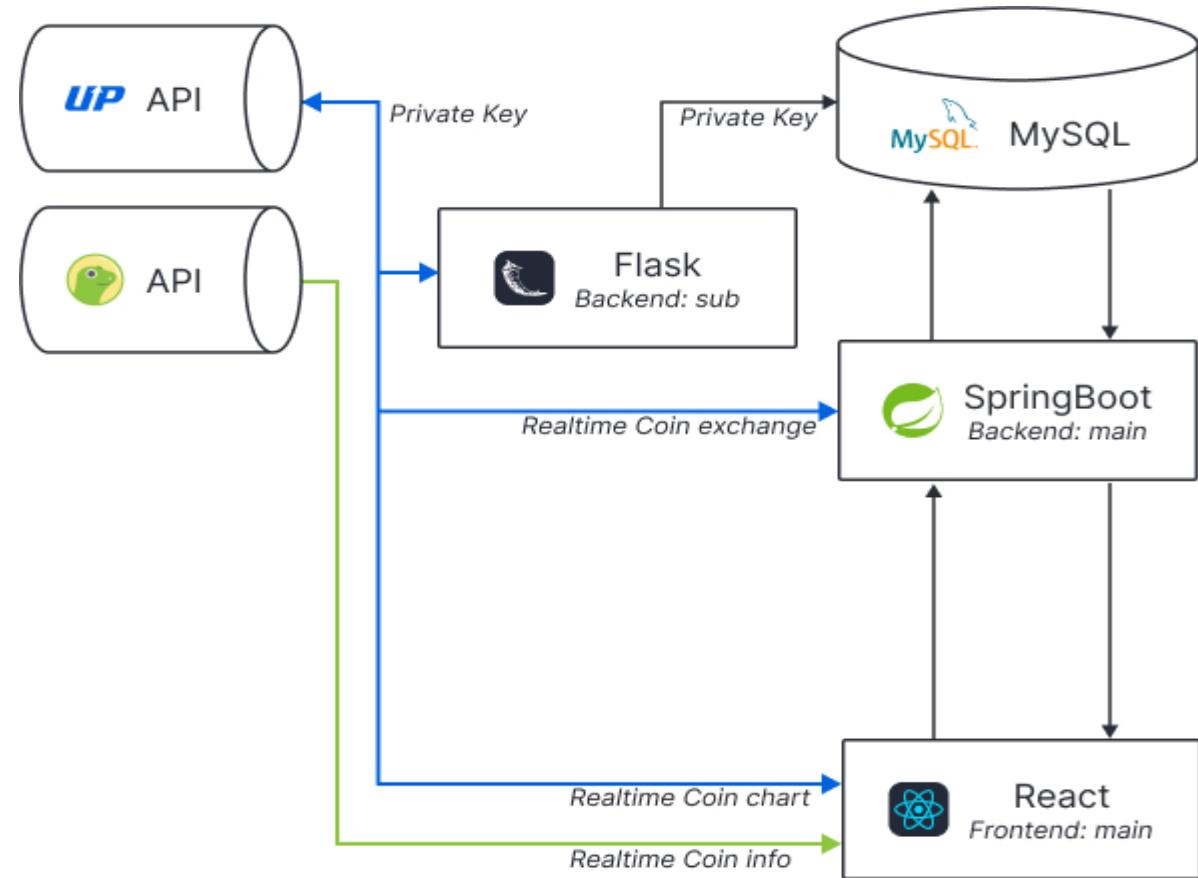
화면 설계

API, Websocket 연동

차트개발

# 프로젝트 개발환경 및 구조

- Python 3.10
- TensorFlow 3.0
- jdk 17
- Spring Boot 3.0.6
- React latest(18.2.0)
- MySQL latest(8.0)

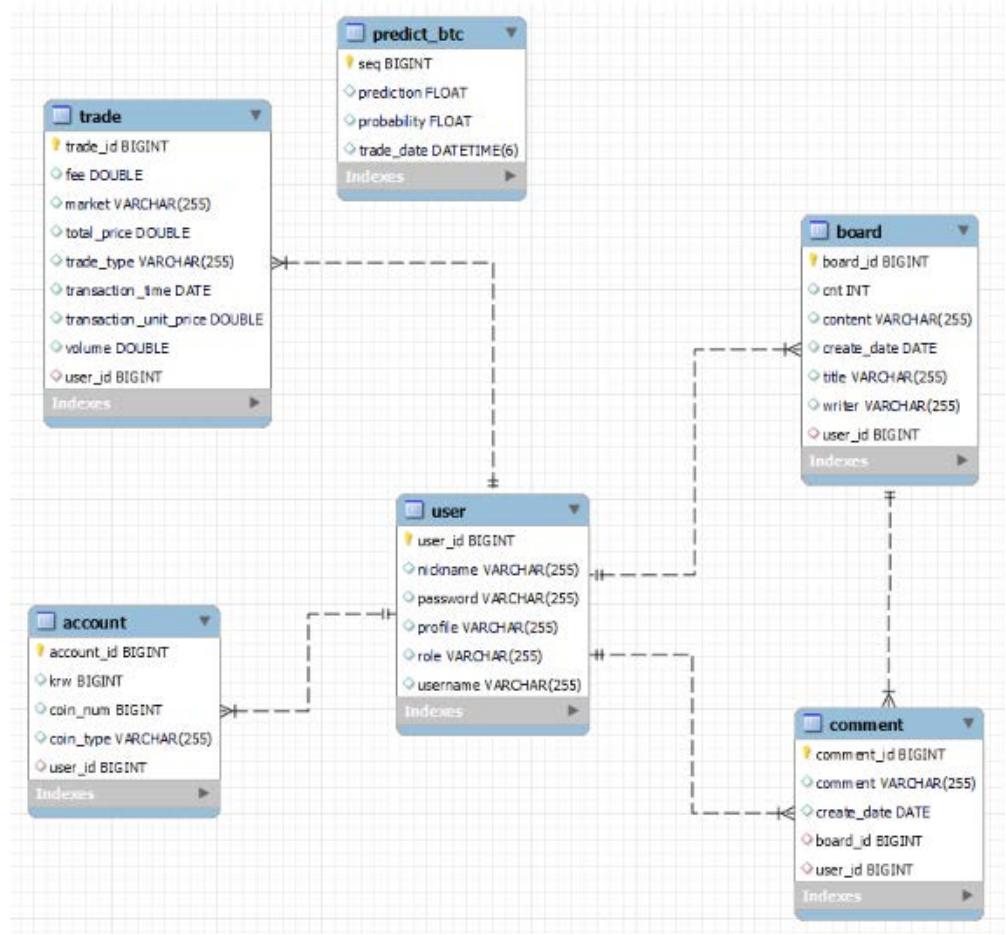


# 프로젝트 요구 명세서

서비스	ID	요구사항명	요구사항 내용	날짜	출처	버전
Data Analysis	REQ-001	가격 방향성 예측	1시간 상승, 하락 확률 표시	23.05.02	수행계획서	0.1
	REQ-002		4시간 상승, 하락 확률 표시	23.05.02	수행계획서	0.1
	REQ-003		24시간 상승, 하락 확률 표시	23.05.02	수행계획서	0.1
시세 그래프	REQ-004	거래 보조지표 표시	거래량, 이동평균선, MACD, RSI, BB	23.05.02	수행계획서	0.1
	REQ-005	비트코인 시세 그래프	업비트 Open API 사용	23.05.02	수행계획서	0.1
회원가입	REQ-006	회원가입	ID, Password, 이름, 닉네임, 전화번호 Form 제공	23.05.02	수행계획서	0.1
	REQ-007	회원가입 인증	[추가] 인증API 사용(카카오, 네이버)	23.05.02	수행계획서	0.1
로그인	REQ-008	로그인	ID, Password Form 제공	23.05.02	수행계획서	0.1
투자내역	REQ-009	계좌내역	종류(자금액, 순이익, 보유코인종목, 보유코인 수, 원화 표시)	23.05.02	수행계획서	0.1
	REQ-010	거래내역	거래내역	23.05.02	수행계획서	0.1
마이페이지	REQ-011	회원정보 수정	닉네임, 비밀번호 수정	23.05.02	수행계획서	0.1
	REQ-012	회원탈퇴	회원 탈퇴 기능	23.05.02	수행계획서	0.1
커뮤니티 게시판	REQ-013	회원 커뮤니티	회원들이 사용할 수 있는 게시판	23.05.18	수행계획서	0.1
	REQ-014	ADMIN 권한	ADMIN 권한으로 게시판 삭제 가능	23.05.18	수행계획서	0.1
	REQ-015	댓글	글마다 댓글 작성 기능	23.05.18	수행계획서	0.1

# Back-End – DB설계

DB ERD



DB Tables

Table	Type	Field	Remarks
User	Long	userId	Primary Key
	String	userName	
	String	password	
	String	nickname	
	String	role	
Account	String	profile	
	Long	accountid	Primary Key
	Long	user_id	Foreign Key(User)
	Long	krw	
	String	coinType	
Trade	Long	coinNum	
	Long	tradeld	Primary Key
	Long	user_id	Foreign Key(User)
	String	market	코인종류
	String	tradeType	거래종류
	Date	transactionTime	거래시점
	Double	totalPrice	총 거래액
Predict_btc	Double	transactionUnitPrice	거래단가
	Double	volume	거래량
	Double	fee	수수료
	Long	probability	확률
Board	Long	seq	Primary Key
	Datetime	tradeDate	예측시간
	Float	prediction	예측가격
	Float	probability	확률
	Long	board_id	Primary Key
	Long	user_id	Foreign Key(User)
Comment	Integer	cnt	
	String	content	
	Date	createDate	
	String	title	
	String	writer	
Comment	Long	commentId	Primary Key
	Long	userId	Foreign Key(User)
	Long	boardId	Foreign Key(Board)
	String	comment	
	Date	createDate	

# Back-End – API(predict, trade, login)

/api/v1/predict		
URI	/getpredict	
Method	GET	
Request	Header	Body
Description	예측 값 조회	
Response	Header	Body 예측 값 반환

URI	/login	
Method	POST	
Request	Header	Body
		String username, String password
Description	회원로그인	
Response	Header	Body Authorization : JWT

/api/v1/trade		
URI	/buy	
Method	POST	
Request	Header	Body Authorization: (JWT), String market(코인종류), Double volume(거래량), Double transactionUnitPrice(거래단가), Double totalPrice(총거래액), Double fee(수수료)
Description	매수	
Response	Header	Body 매수 실행 결과
URI	/sell	
Method	POST	
Request	Header	Body Authorization: (JWT), String market(코인종류), Double volume(거래량), Double transactionUnitPrice(거래단가), Double totalPrice(총거래액), Double fee(수수료)
Description	매도	
Response	Header	Body 매도 실행 결과

# Back-End – API(users)

/api/v1/users		
URI	/join	
Method	POST	
Request	Header -	Body String username, String password, String nickname, String profile
Description	회원가입	
Response	Header -	Body 회원가입 결과
URI	/delete	
Method	DELETE	
Request	Header Authorization: (JWT)	Body -
Description	회원탈퇴	
Response	Header	Body 회원탈퇴 결과

URI	/update	
Method	PUT	
Request	Header Authorization: (JWT)	Body String password, String nickname
Description	회원정보수정	
Response	Header -	Body 회원정보수정 결과
URI	/userInfo	
Method	GET	
Request	Header Authorization: (JWT)	Body -
Description	유저 정보 조회	
Response	Header -	Body 유저 개인정보 및 계좌정보(총 투자금액, 순이익, 등)
URI	/tradeDetails?current=(boolean)	
Method	POST	
Request	Header Authorization: (JWT)	Body -
Description	유저 거래내역 조회	
Response	Header -	Body 유저 거래내역

# Back-End – API(board)

## /api/v1/board

URI	/getList	
Method	GET	
Request	Header	Body
Description	게시판 목록 조회	
Response	Header	Body 게시판 목록 반환
URI	/get	
Method	GET	
Request	Header	Body Authorization: (JWT)
Description	게시글 조회	
Response	Header	Body 게시글 반환
URI	/insert	
Method	POST	
Request	Header	Body Authorization: (JWT) String title, String content
Description	게시글 등록	
Response	Header	Body 등록 결과

URI	/update	
Method	PUT	
Request	Header	Body Authorization: (JWT) String title, String content
Description	게시글 수정	
Response	Header	Body 수정 결과
URI	/delete	
Method	DELETE	
Request	Header	Body Authorization: (JWT) Object board: { Long boardId }
Description	게시글 삭제	
Response	Header	Body 삭제 결과
URI	/deleteByAdmin	
Method	DELETE	
Request	Header	Body Authorization: (JWT) Object board: { Long boardId }
Description	게시글 삭제(Admin)	
Response	Header	Body 삭제 결과

# Back-End – API(comment)

URI	/delete	
Method	DELETE	
Request	Header Authorization: (JWT)	Body Object comment: { Long commentId }
Description	댓글 삭제	
Response	Header -	Body 삭제 결과
URI	/update	
Method	PUT	
Request	Header Authorization: (JWT)	Body Object comment: { Long commentId, String comment }
Description	댓글 수정	
Response	Header -	Body 수정 결과

URI	/api/v1/comment	
Method	GET	
Request	Header -	Body -
Description	댓글 목록 조회	
Response	Header -	Body 댓글 목록 반환
URI	/insert	
Method	POST	
Request	Header Authorization: (JWT)	Body String comment
Description	댓글 등록	
Response	Header -	Body 등록 결과

# Back-End – 문제점

1. Security: SpringBoot의 Version Update에 따른 사용법 변경

꾸준한 TRY

2. Front-End와 통신: REST API

학습 및 협의

# CRA없이 React 시작하기



```
const config: webpack.Configuration = {
  name: 'analyzer-front',
  mode: isDevelopment ? 'development' : 'production',
  devtool: !isDevelopment ? 'hidden-source-map' : 'eval',
  resolve: {
    extensions: ['.js', '.jsx', '.ts', '.tsx', '.json'],
    alias: {
      '@hooks': path.resolve(__dirname, 'src/hooks'),
      '@components': path.resolve(__dirname, 'src/components'),
      '@pages': path.resolve(__dirname, 'src/pages'),
      '@stores': path.resolve(__dirname, 'src/stores'),
      '@types': path.resolve(__dirname, 'src/types'),
      '@utils': path.resolve(__dirname, 'src/utils')
    }
  },
  entry: { app: './src/index' },
  module: {
    rules: [
      {
        test: /\.tsx?$/,
        loader: 'babel-loader',
        options: {
          ...
        }
      }
    ]
  }
}
```

CRA(create-react-app)을 활용하면 webpack 또는 babel에 대한 복잡한 설정 없이 빠르게 React 개발 환경을 세팅할 수 있지만, webpack과 babel에 대해 추가적인 설정이 필요할 경우 어려움이 발생할 수 있습니다.

UpbitAnalyzer 프로젝트에는 직접 React 개발환경을 구축해 개발 시 절대경로를 활용하고, bootstrap sass를 커스터마이징할 수 있었습니다.



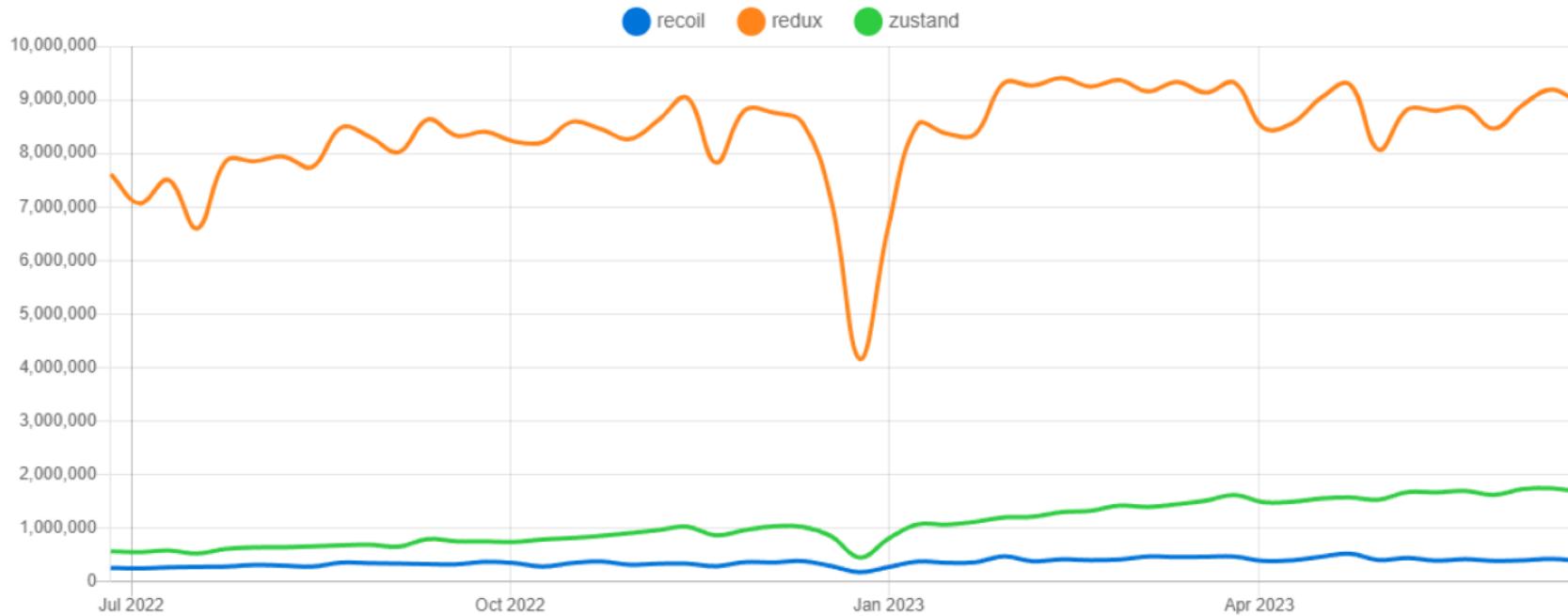
# TS Typescript로 명시적으로 개발하기

UpbitAnalyzer 프로젝트는 다양한 데이터를 표현해야하기 때문에 개발 중 데이터의 타입이 객체의 필드나 함수의 매개변수로 사용될 때의 혼동을 방지할 필요가 있었습니다.

js의 유연성은 빠른 개발경험을 제공해 줄 수 있겠지만, 직관적이고 견고하게 개발하기 위해 Typescript를 도입하였습니다.

```
{  
  "compilerOptions": {  
    "target": "es5",  
    "strict": true,  
    "lib": ["ES2021", "DOM"],  
    "jsx": "react-jsx",  
    "module": "commonjs",  
    "moduleResolution": "node",  
    "allowJs": true,  
    "sourceMap": true,  
    "esModuleInterop": true,  
    "isolatedModules": true,  
    "allowSyntheticDefaultImports": true,  
    "forceConsistentCasingInFileNames": true,  
    "baseUrl": "./src",  
    "paths": {  
      "@components/*": [ "components/*" ],  
      "@pages/*": [ "pages/*" ],  
      "@hooks/*": [ "hooks/*" ],  
      "@stores/*": [ "stores/*" ],  
      "@types/*": [ "types/*" ],  
      "@utils/*": [ "utils/*" ]  
    }  
  }  
}
```

# Redux 대신 zustand



상태관리를 위해 제일 먼저 고려되는 라이브러리는 Redux일텐데, 규모에 따라 Redux는 방대한 기능을 제공하는 리소스로 인해 보일러플레이트를 관리해야 하는 불편함이 생길 수 있습니다. 그 대안으로 Recoil, Jotai등 쉽고 우수한 상태관리 라이브러리를 사용할 수 있습니다.

zustand는 위 라이브러리들에 비해 불필요한 리렌더링을 적게 초래하고, 스토어 내에서 action을 설정하므로써 컴포넌트 외부에서 관리할 수 있습니다.

# immer middleware로 상태에 불변성 관리하기

immer는 draft라는 가상 작업영역에서 변경된 상태에 대해 구조 공유 메커니즘을 활용해 이전 객체와 공유되는 부분을 최대한 활용하고, 변경되지 않는 부분은 이전 객체와 공유하므로써 메모리 사용량을 최소화합니다.

상태의 불변성 관리를 immer middleware에 위임하므로써 얇은 비교로 효율적으로 상태를 업데이트할 뿐만 아니라, 예상치 못한 사이드 이펙트를 방지할 수 있습니다.



## Lothrottle로 밀려오는 Websocket 데이터 관리하기

업비트의 websocket을 활용하면 업비트의 실시간 데이터를 받아올 수 있습니다. 그러나 1초에 150개가 넘는 데이터가 밀려오고 매번 상태를 변경하려고 한다면?

React.memo를 사용해 리렌더링을 최적화하려해도 업비트의 제한사항을 위반하거나, 사이트가 마비될 것입니다.

이를 해결하기 위해 일종의 **buffer**를 사용할 수 있습니다. 시간 단위로 데이터를 쌓아 처리하는 throttle을 사용할 수 있습니다.

(해결 후 확인해보니, throttle 대신 buffer만 활용하는 것이 더 좋을 것으로 판단됩니다. 누락되는 데이터가 있을 수 있기 때문.)



# React-Financial-Charts



react-financial-charts는 d3 기반의 금융 차트 라이브러리입니다. d3에 의존해 데이터를 처리하지만, 비교적 어렵지 않은 난이도로 금융차트를 표시할 수 있습니다.

# Data Analysis – 데이터 출처 : 업비트 데이터

## Request Parameters

필드	설명	타입
market	마켓 코드 (ex. KRW-BTC)	String
to	마지막 캔들 시작 (exclusive). ISO8601 포맷 (yyyy-MM-dd'T'HH:mm:ss'Z' or yyyy-MM-dd HH:mm:ss), 기본적으로 UTC 기준 시간이며 2023-01-01T00:00:00+09:00 과 같이 KST 시간으로 요청 가능. 비워서 요청시 가장 최근 캔들	String
count	캔들 개수(최대 200개까지 요청 가능)	Integer

## Response

필드	설명	타입
market	마켓명	String
candle_date_time_utc	캔들 기준 시간(UTC 기준) 포맷: yyyy-MM-dd'T'HH:mm:ss	String
candle_date_time_kst	캔들 기준 시간(KST 기준) 포맷: yyyy-MM-dd'T'HH:mm:ss	String
opening_price	시가	Double
high_price	고가	Double
low_price	저가	Double
trade_price	종가	Double
timestamp	해당 캔들에서 마지막 틱이 저장된 시각	Long
candle_acc_trade_price	누적 거래 금액	Double
candle_acc_trade_volume	누적 거래량	Double
unit	분 단위(유닛)	Integer

INSTALLATION

```
$ python -m pip install requests
```

REQUEST

```
1 import requests
2
3 url = "https://api.upbit.com/v1/candles/minutes/1?market=KRW"
4
5 headers = {"accept": "application/json"}
6
7 response = requests.get(url, headers=headers)
8
9 print(response.text)
```

RESPONSE

EXAMPLES ▾

Choose an example:

application/json

● 200 - Result ● 400 - Result

	opening_price	high_price	low_price	trade_price
candle_date_time_kst	2023-05-30 19:15:00	37213000.0	37213000.0	37200000.0
	2023-05-30 19:16:00	37203000.0	37215000.0	37200000.0
	2023-05-30 19:17:00	37216000.0	37238000.0	37215000.0
	2023-05-30 19:18:00	37237000.0	37248000.0	37235000.0
	2023-05-30 19:19:00	37247000.0	37253000.0	37247000.0

	candle_acc_trade_price	candle_acc_trade_volume
candle_date_time_kst	2023-05-30 19:15:00	1.004070e+08
	2023-05-30 19:16:00	4.257396e+07
	2023-05-30 19:17:00	1.049734e+08
	2023-05-30 19:18:00	8.637001e+07
	2023-05-30 19:19:00	8.504999e+07

# Data Analysis – 데이터셋

```
# 데이터 전처리
scaler = MinMaxScaler()
scale_cols = ['trade_price']
scaled = scaler.fit_transform(df[scale_cols])
test_scaled = scaler.transform(df[scale_cols])

# 데이터 분할
train_size = int(len(df) * 0.8)
train_data = df[:train_size]
test_data = df[train_size:]

# 시퀀스 길이
sequence_length = 30

# 데이터셋 생성
def create_dataset(data, sequence_length):
    X, y = [], []
    for i in range(len(data) - sequence_length):
        X.append(data[i:(i + sequence_length)])
        y.append(data[i + sequence_length][0])
    return np.array(X), np.array(y)

X_train, y_train = create_dataset(scaled, sequence_length)
X_test, y_test = create_dataset(test_scaled, sequence_length)
```

(1039220, 30, 1)  
(1039220,)  
(1039220, 30, 1)  
(1039220,)

# Data Analysis – 모델 CNN, LSTM

```
# CNN 모델과 LSTM 모델
```

```
model = Sequential()
model.add(Conv1D(filters=32, kernel_size=3, activation='relu', input_shape=(sequence_length, 1)))
model.add(Conv1D(filters=32, kernel_size=3, activation='relu'))
model.add(Dropout(0.1))
model.add(LSTM(units=64, activation='tanh', return_sequences=True))
model.add(LSTM(units=64, activation='tanh'))
model.add(Dense(units=128, activation='relu'))
model.add(Dense(units=1))
model.compile(optimizer=Adam(learning_rate=0.0001), loss='mean_squared_error', metrics=['accuracy'])

early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

from tensorflow.keras.callbacks import EarlyStopping

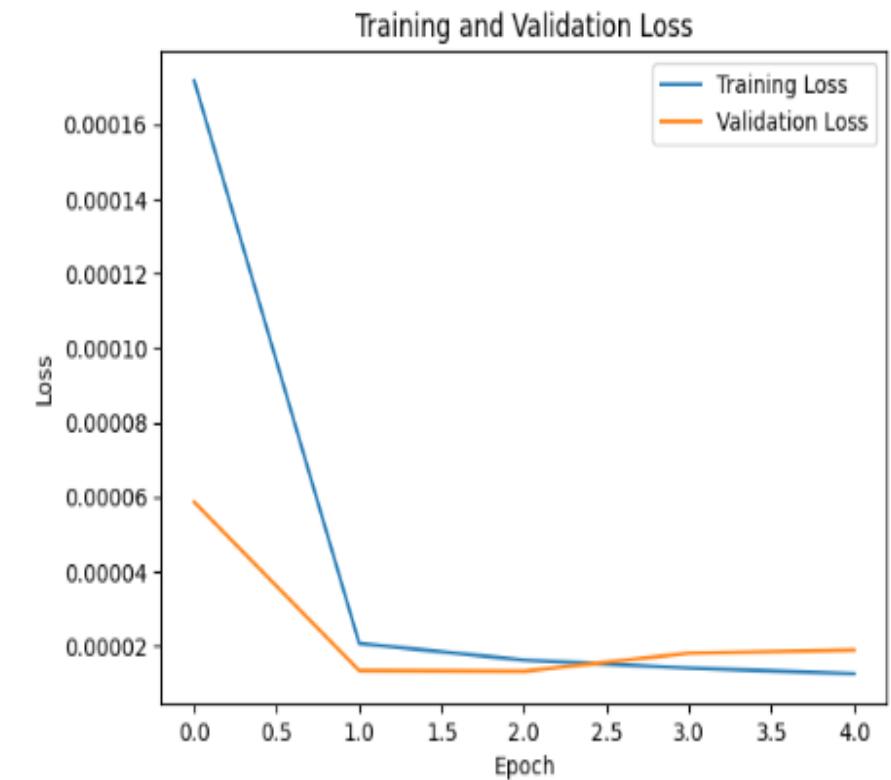
# Define the EarlyStopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Train the model with the EarlyStopping callback
model.fit(X_train, y_train, validation_split=0.2, epochs=5, batch_size=32, callbacks=[early_stopping])
```

7일치 예상 가격표

```
Actual Prices:
[0.34261366 0.34261366 0.34261366 0.34261366 0.34261366 0.34261366
 0.34261366]

Predicted Prices:
[41992232. 41992736. 41992744. 41992744. 41992744. 41992744. 41992744.]
```



# Data Analysis – 감성분석

	Date	news_title	close_price
0	2018-01-01 09:00:00	비트코인 두 달만에 3만 달러 돌파..파월 "화폐의 지위 가진 듯"(종합)	18860000.0
1	2018-01-01 09:00:00	블록체인 띄운 비트코인 금값..한때 2만8천달러 넘기도	18860000.0
2	2018-01-01 09:00:00	비트코인 두 달 만에 3만 달러 돌파	18860000.0
3	2018-01-01 09:00:00	두 달만에 3만달러 돌파한 비트코인	18860000.0
4	2018-01-01 09:00:00	비트코인 금값세...	18860000.0

뉴스 크롤링 과 업비트 데이터를 한곳에 모으기  
데이터 전처리 작업  
감성 분석 을 이용하여 부정, 중립, 긍정  
LSTM 모델 사용 후 결과 확인  
보조지표 사용하여 결과 확인

# Data Analysis – 데이터 전처리

```
# 텍스트 정제
text_data = [text.translate(str.maketrans('', '', '!"#$%^&()_*.;<>?@[\\]^_{}~`')) for text in text_data]
text_data = [re.sub(r'\d+', '', text) for text in text_data]
text_data = [text.strip() for text in text_data]

# 불용어 처리
okt = Okt()
stopwords = ["은", "는", "이", "가", "들", "를", "으로", "로", "에", "에서"]
text_data = [' '.join([word for word in okt.morphs(text) if word not in stopwords]) for text in text_data]

# 토큰화 처리
tokenizer = Tokenizer()
tokenizer.fit_on_texts(text_data)
sequences = tokenizer.texts_to_sequences(text_data)
word_index = tokenizer.word_index

# 시퀀스 패딩
max_sequence_length = max([len(seq) for seq in sequences])
padded_sequences = pad_sequences(sequences, maxlen=max_sequence_length)

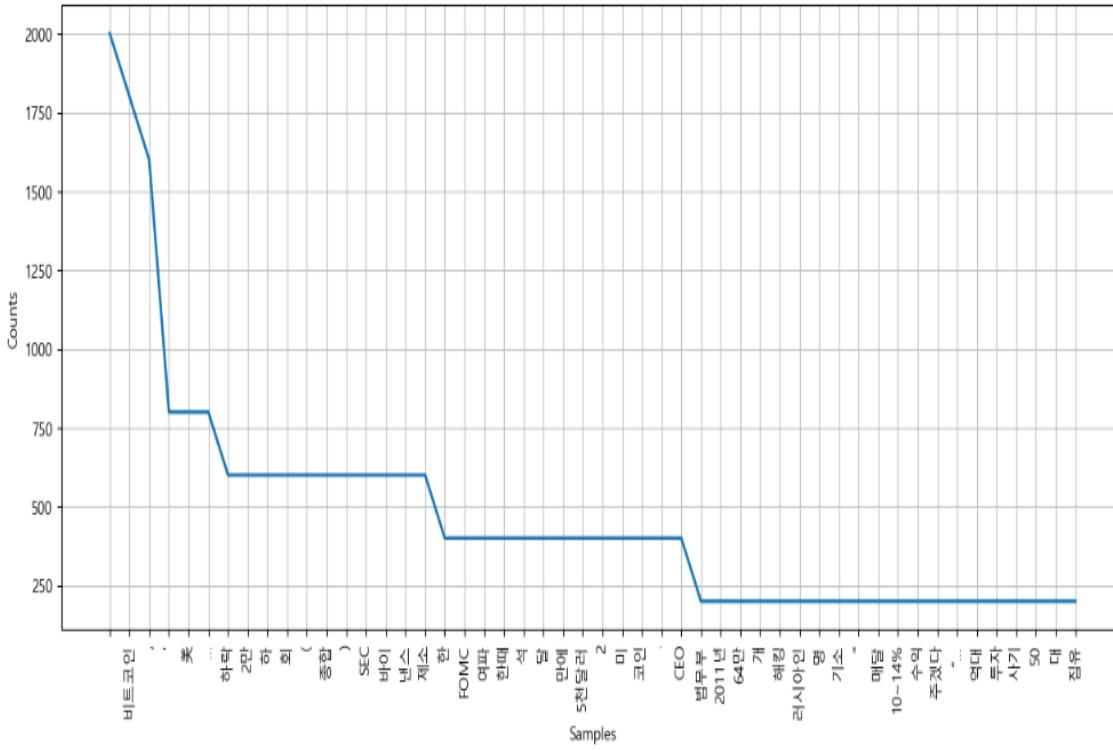
# 워드 임베딩 행렬 생성
embedding_dim = 300
word2vec_model = Word2Vec(sentences=[text.split() for text in text_data], vector_size=embedding_dim, window=5, min_count=1, workers=4)
embedding_matrix = np.zeros((len(word_index) + 1, embedding_dim))
for word, i in word_index.items():
    if word in word2vec_model.wv:
        embedding_matrix[i] = word2vec_model.wv[word]
```

불필요한 문자 및 숫자를 제거하고, 각 문장을 정제  
한국어에서 불용어로 간주되는 단어들을 제거합니다.  
문장을 단어 단위로 토큰화합니다.

Word2Vec 모델을 사용하여 단어의 임베딩 벡터를  
생성하고, 임베딩 행렬을 만듭니다.

# Data Analysis – 단어 추출 시각화

[('\'n', 2000), ('비트코인', 1800), ('.', 1600), ('"', 800), ('英', 800), ('\_ ', 800), ('하락', 600), ('2月', 600), ('한', 600), ('회', 600), ('(', 600), ('증권', 600), (')', 600), ('SEC', 600), ('파이', 600), ('모스', 600), ('포스', 600), ('한화', 400), ('FOMC', 400), ('대부', 400), ('한화', 400), ('설', 400), ('율', 400), ('한화', 400), ('5천원권', 400), ('2', 400), ('%', 400), ('인도', 400), ('.', 400), ('CEO', 400), ('법률부', 200), ('2011년', 200), ('64만', 200), ('개', 200), ('해원', 200), ('리시아인', 200), ('증', 200), ('기소', 200), ('"', 200), ('법률', 200), ('10~14%', 200), ('수익', 200), ('주겠다', 200), ('\_ ', 200), ('역대', 200), ('투자', 200), ('사기', 200), ('\_ ', 200), ('집유', 200), ('증권', 200), ('포트', 200), ('통증', 200), ('사업자', 200), ('까지', 200), ('회계', 200), ('서비스', 200), ('잇단', 200), ('출금', 200), ('출판', 200), ('로열티', 200), ('보', 200), ('이어', 200), ('비이스', 200), ('도', 200), ('온', 200), ('방문', 200), ('회', 200), ('밀원', 200), ('출판', 200), ('출판', 200), ('회', 200), ('기준', 200), ('첫', 200)]



# Data Analysis – 데이터

```
from sklearn.linear_model import LogisticRegression
# 감성 분석 결과 생성
sentiment_labels = ['0', '1', '2'] # 0 : 부정, 1 : 중립, 2 : 긍정
sentiments = np.random.choice(sentiment_labels, size=len(text_data))

# 데이터에 감성 레이블 추가
data['sentiment'] = sentiments

# GPT 데이터 불러오기
X = padded_sequences # 페드드 텍스트 데이터
y = [sentiment_labels.index(sentiment) for sentiment in sentiments] # 감성 레이블 인덱스로 변환

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 로지스틱 회귀 모델 초기화
model = LogisticRegression()
```

2023-06-18 09:00:00	美 SEC, 바이낸스 CEO...	34466000.0	2
2023-06-18 09:00:00	美 SEC, 바이낸스 이어...	34466000.0	1
2023-06-18 09:00:00	미 SEC, 바이낸스 CEO...	34466000.0	0
2023-06-18 09:00:00	챗GPT에 밀린 비트코...	34466000.0	2
2023-06-19 09:00:00	하락한 비트코인	34553000.0	0
2023-06-19 09:00:00	비트코인, '美 FOMC ...	34553000.0	1
2023-06-19 09:00:00	美 법무부, 2011년 비...	34553000.0	2
2023-06-19 09:00:00	"매달 10~14% 수익 ...	34553000.0	0
2023-06-19 09:00:00	비트코인, '미 FOMC ...	34553000.0	0
2023-06-19 09:00:00	금융당국 등록사업자...	34553000.0	1
2023-06-19 09:00:00	美 SEC, 바이낸스 CEO...	34553000.0	2
2023-06-19 09:00:00	美 SEC, 바이낸스 이어...	34553000.0	0
2023-06-19 09:00:00	미 SEC, 바이낸스 CEO...	34553000.0	2
2023-06-19 09:00:00	챗GPT에 밀린 비트코...	34553000.0	2

부정, 중립, 긍정 데이터 까지 컬럼 추가하기

# Data Analysis 종가 감성 모델

```
# 데이터 스케일링
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(df[['close_price', 'sentiment']].values)

# 데이터 분할
X = pd.to_numeric(df.index, errors='coerce').values.reshape(-1, 1, 1)
y = scaled_data

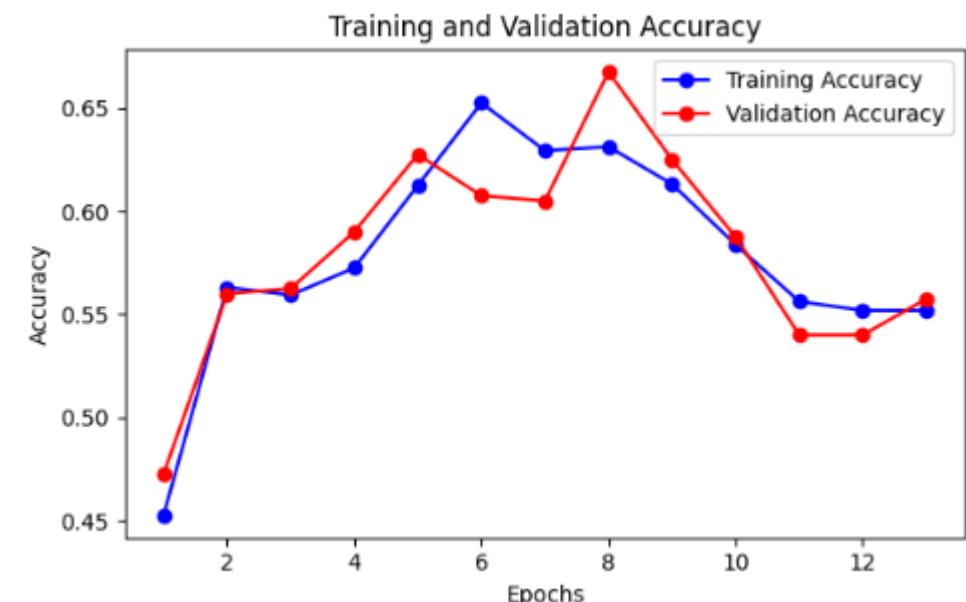
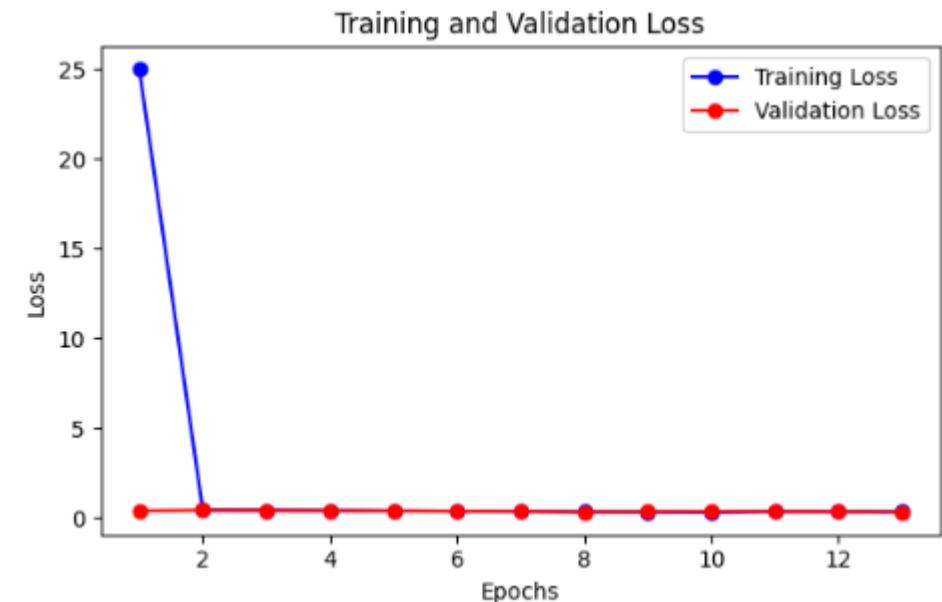
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.2)

# 모델 구조
model = Sequential()
model.add(LSTM(32, activation='relu', input_shape=(1, 1)))
model.add(Dense(2)) # 2개의 출력 노드: close_price & sentiment

model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
early_stop = EarlyStopping(monitor='val_loss', patience=5)

# 모델 학습
history = model.fit(X_train, y_train,
                      epochs=50,
                      batch_size=64,
                      validation_data=(X_valid, y_valid),
                      callbacks=[early_stop])

Epoch 10/50
25/25 [=====] - 0s 2ms/step - loss: 0.2886 - accuracy: 0.!
Epoch 11/50
25/25 [=====] - 0s 2ms/step - loss: 0.3454 - accuracy: 0.!
Epoch 12/50
25/25 [=====] - 0s 2ms/step - loss: 0.3376 - accuracy: 0.!
Epoch 13/50
25/25 [=====] - 0s 2ms/step - loss: 0.3269 - accuracy: 0.!
1/1 [=====] - 0s 128ms/step
예측 결과: 하락 | 8.58%
```



# Data Analysis 보조지표 모델

```
# 이동평균 계산
window_size = 60
data['MA'] = data['close_price'].rolling(window=window_size).mean()

# MACD 계산
indicator_macd = MACD(close=data['close_price'], window_slow=26, window_fast=12, window_sign=9)
data['MACD'] = indicator_macd.macd()

# RSI 계산
indicator_rsi = RSIIndicator(close=data['close_price'], window=14)
data['RSI'] = indicator_rsi.rsi()
```

```
# 데이터 전처리
x = data[['MA', 'MACD', 'RSI', 'sentiment']].values
y = np.where(data['close_price'].shift(-1) > data['close_price'], 1, 0)

# 데이터 스케일링
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(x)

scaled_data = scaled_data.reshape(-1, 1, 4)

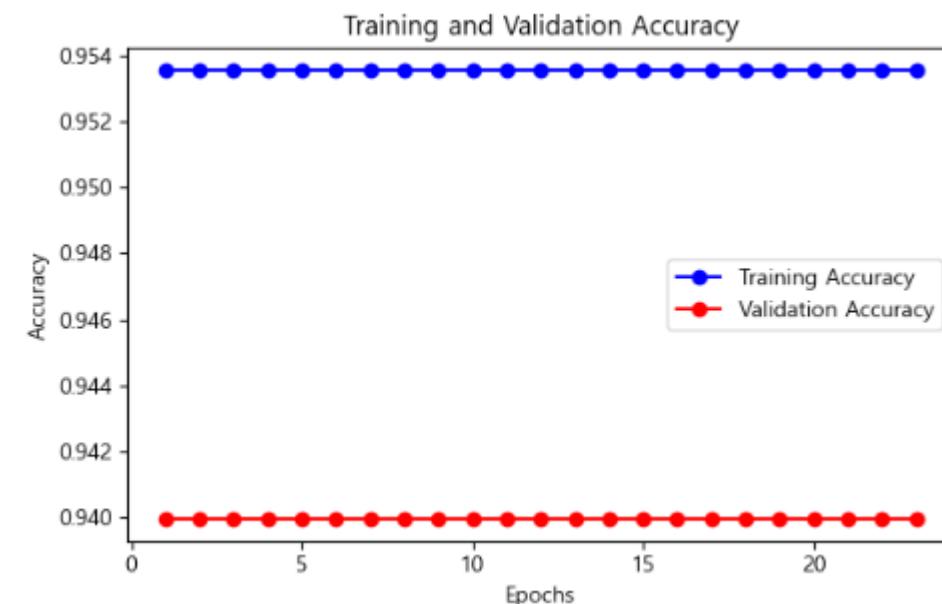
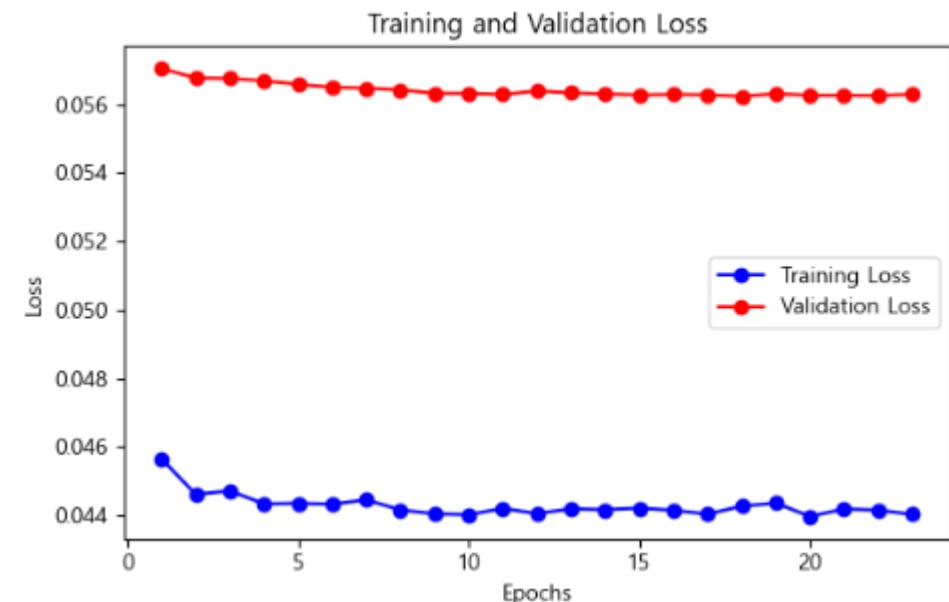
x_train, x_valid, y_train, y_valid = train_test_split(scaled_data, y, test_size=0.2)

# 모델 구성
model = Sequential()
model.add(LSTM(16, activation='relu', input_shape=(1, 4)))
model.add(Dropout(0.2)) # 드롭아웃 추가
model.add(Dense(1)) # 2개의 출력 노드: close_price와 sentiment

# mean_squared_error 예측과 결과 사이의 평균 제곱 오차를 최적화한다
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
early_stop = EarlyStopping(monitor='val_loss', patience=5)

# 모델 학습
history = model.fit(x_train, y_train,
                      epochs=100,
                      batch_size=64,
                      validation_data=(x_valid, y_valid),
                      callbacks=[early_stop])
```

[1912 rows x 6 columns]  
예측 가격: 21745236.5863199



# 자체 평가 의견

결과:

1. 비트코인 가격 분석, 예측을 통해 예상가격 그래프를 시각화
2. 실제 코인 시세 그래프와 예측 그래프를 한 화면에서 비교 가능
3. 예측이 코인투자에서 하나의 지표로 활용되는 것이 기대됨
4. 커뮤니티 구현: 웹 이용 활성화

향후 목표:

1. 이러한 예측 결과로 자동매매시스템을 제공
2. 한 화면에서 차트만 보는 것을 넘어서 실제 업비트 거래서비스를 제공
3. 감성분석을 하여 그 결과를 또 다른 지표로 제공
4. 커뮤니티 기능 심화