반도체 주가 요인 분석 및 예측

팀장 오재호 | 팀원 최수지 김종민 구현주 권동형

ID 투자증권

ID 투자증권

목차

1. 주제 리뷰

주제 제안

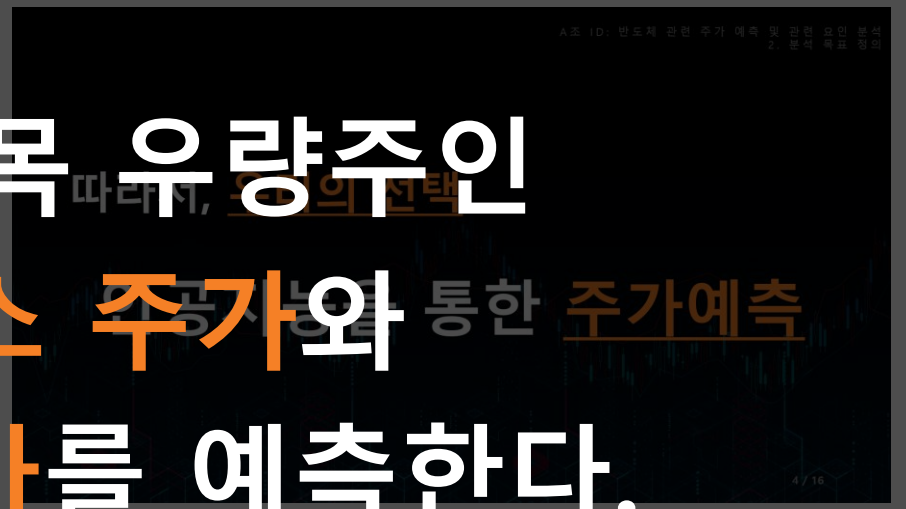
▶ "주식 보유자의 수가 1천만명이 넘어"

▶ 제안발표(21.04.22)

주제 제안

분석 목표

▶ "주식 보유자의 수가 1천만명이 넘어"



▶ 제안발표(21.04.22)

▶ 반도체 종목 유량주인
SK하이닉스 주가와
삼성전자의 주가를 예측한다.

요인 분석

2. Stock prediction model

오래전부터 주가 예측 연구는 끊임없이 진행됐다. 하지만 수많은 변수와 정보는 반영하기 쉽지 않다. 주가 분석 방법에는 크게 기술적 분석(Technical Analysis)과 거시적 경제 분석(Macroeconomics Analysis)으로 나눌 수 있다. 기술적 분석은 간단히 말하자면 과거 주가 움직임과 거래량을 바탕으로 예측을 하는 방법이며, 거시적 경제 분석은 주가에 영향을 미치는 변수 즉 금값, 유가, 환율, 물가상승률 등을 고려하여 예측하는 것이 대표적이다.

Reference : 장은아, 최회련, 이홍철(2020), "BERT를 활용한 뉴스 감성분석과 거시경제지표 조합을 이용한 주가지수 예측"

세 가지 분석 기법

▶ **기술적 분석**
주가 데이터

▶ **거시적 경제 분석**
환율, 유가, 금리 +α

▶ **기업 내부요인 분석**
기업 재무제표

2. 데이터 전처리

데이터셋 수집


SK하이닉스 주가
삼성전자 주가
기술적 분석
국제 유가
원달러환율
경제 성장률
기준 금리
반도체수출 금액지수
필라델피아 반도체지수
정기예금
코스피
코스닥
비트 코인
S&P 500
SK하이닉스 재무제표
삼성전자 재무제표
내부적 요인

데이터셋 수집

SK하이닉스 주가
삼성전자 주가

기술적 분석

국제 유가
원달러 환율
경제 성장률
기준 금리
반도체 수출 금액지수
필라델피아 반도체지수

▶ 총 15개 데이터셋

정기예금
코스피
코스닥
비트코인
S&P 500

SK하이닉스 재무제표
삼성전자 재무제표

내부적 요인

데이터셋 수집

Df = pd.merge(df1, df2, how="left") // index는 d1에 맞춰서 인자 넣어주기

A	B	C	D	E
Date	X1	X2	…	Y
Data1				
Data2				
…				

+

A	B	C
Date	X1	X2
Data1		
Data2		
…		

종가의 날짜(연별, 일별)를 기준으로 15개 데이터셋 Merge 진행

▶ 일별 데이터셋(주가,유가,환율,금리..등)

연별 데이터셋(주가, 재무제표..등) 생성

결측치 대체

A	B	C	D	E
Date	**H종가**	**한국은행 기준금리**	**WTI 현물유가 등락률**	**…**
Data1	64319	0.75	4.98	…
Data2	78568	**NaN**	**NaN**	…
Data3	75829	**NaN**	2.96	…
…	…	…		…

Merge한 데이터셋에서 NaN 발생

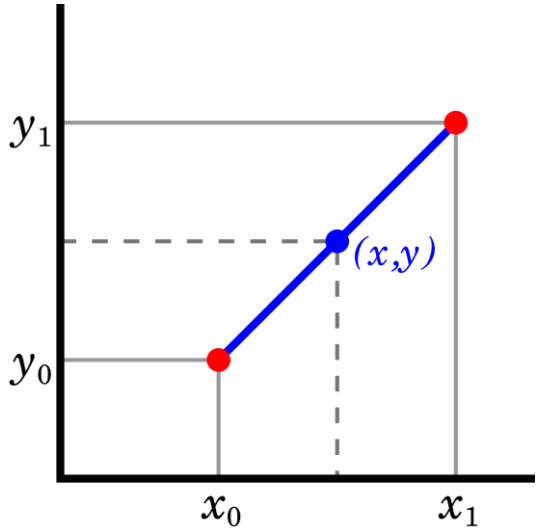
▶ **결측치 처리 진행**

결측치 대체

보간법 : 존재하는 값들 사이에 존재하지 않는 값을 새롭게 추정하는 방법

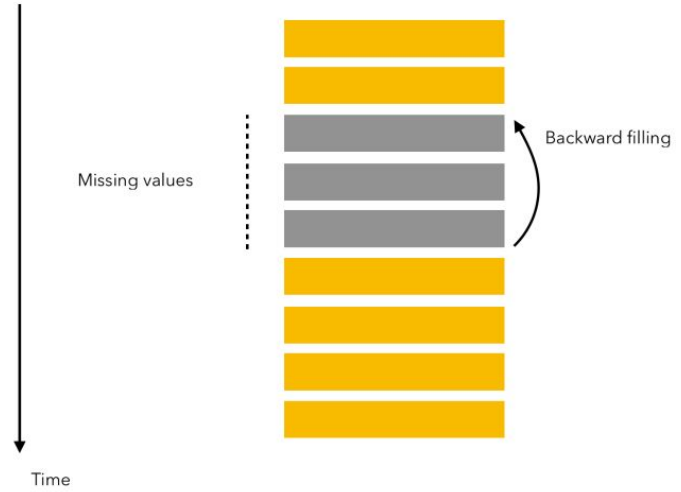
선형보간법

추정 값을 직선 거리에 따라
선형적으로 계산하는 보간법

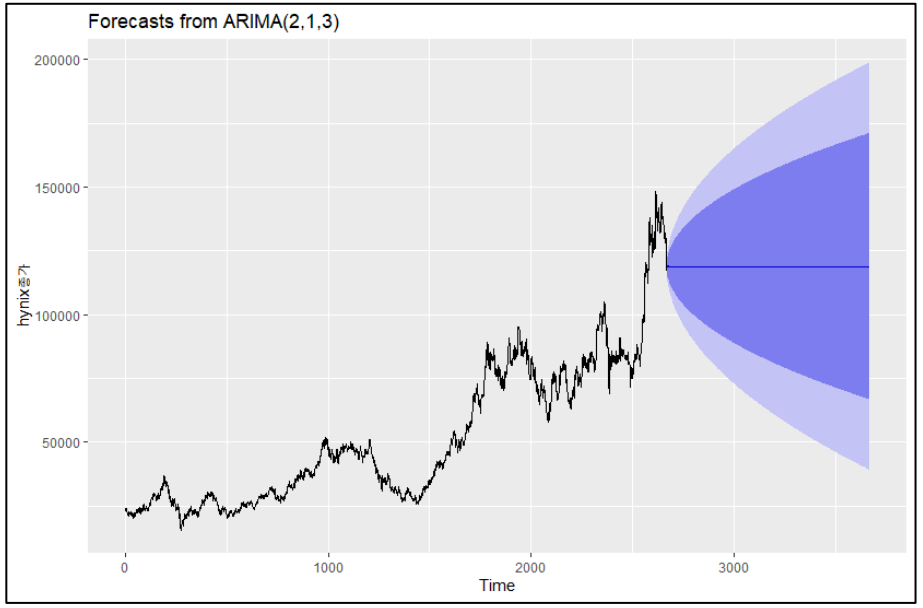


후행보간법

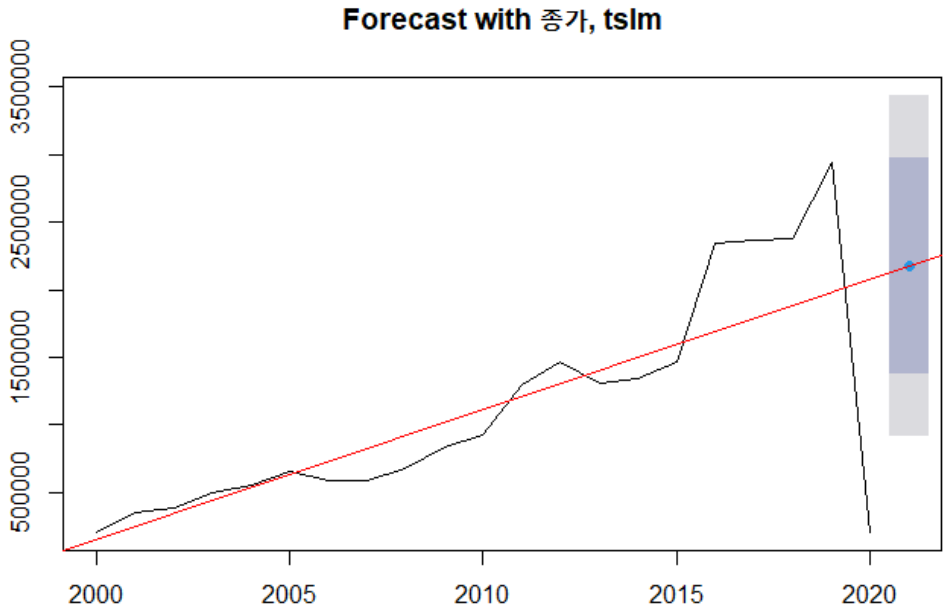
결측치가 존재할 경우 직후 마지막
값을 끌어와서 메꾸는 보간법



데이터셋 변환

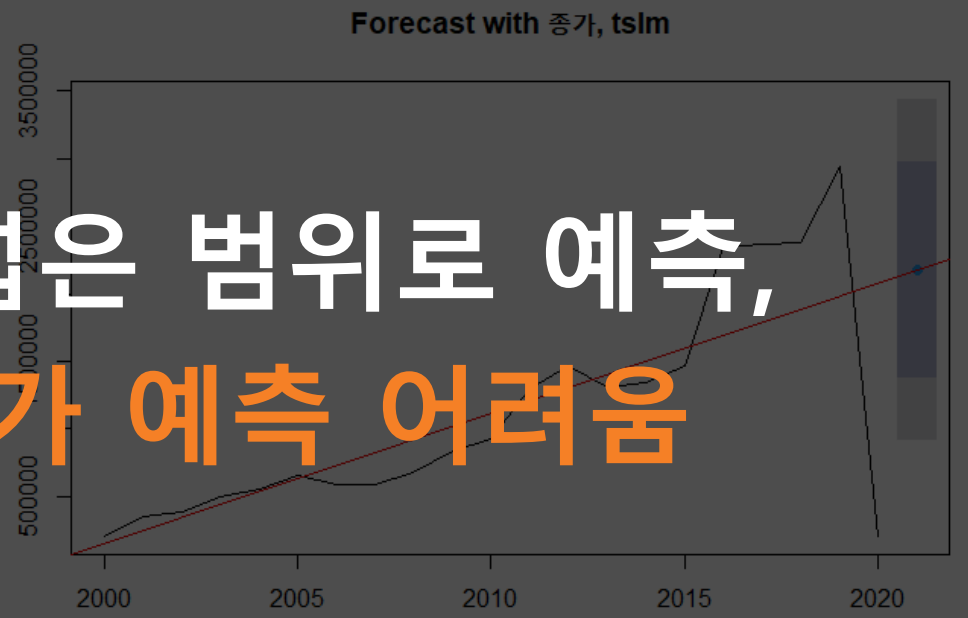


▶ ARIMA 모델의 예측 결과



▶ Tslm 모델의 예측 결과

데이터셋 변환

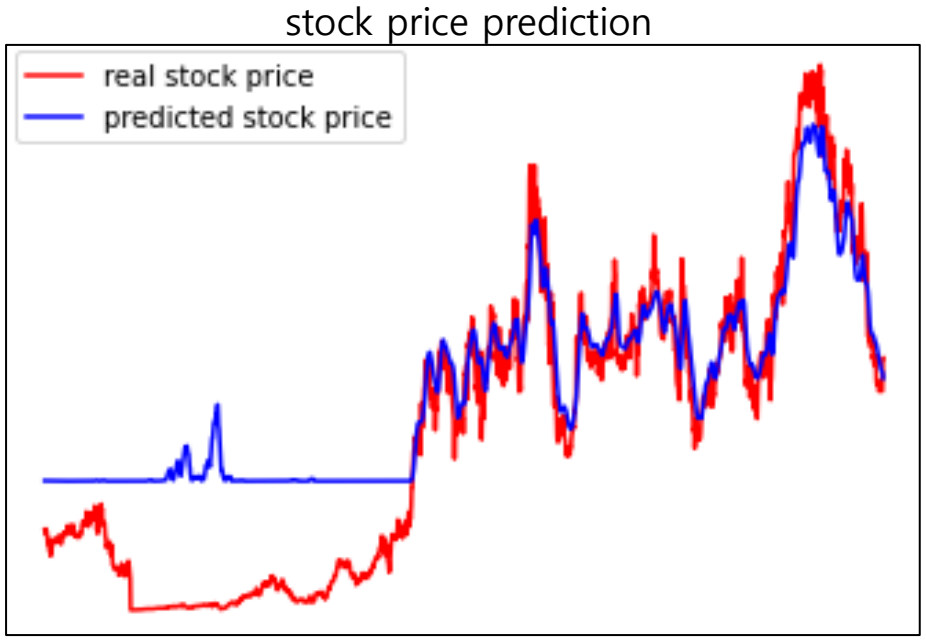


▶ 시계열 모델은 넓은 범위로 예측,
따라서 **명확한 주가 예측 어려움**

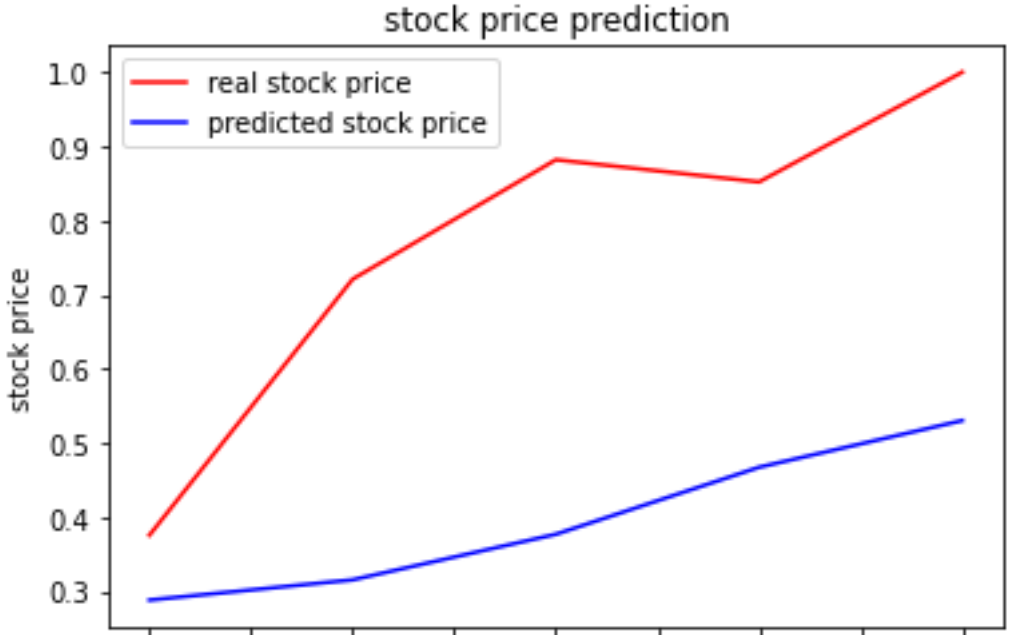
▶ ARIMA 모델의 예측 결과

▶ Tslm 모델의 예측 결과

데이터셋 변환



▶LSTM이 오버피팅일 때



▶LSTM이 오버피팅이 아닐 때

데이터셋 변환

▶LSTM이 오버피팅일 때

▶LSTM을 오버피팅이지 않을 때

▶ LSTM 모델 주가 예측은

후행 문제 발생

데이터셋 변환

기존 종가

Y
H종가
123000
119500
117500
…

Y
H등락
1
0
0
…

새 종속변수 '등락'

$f(\text{종가 변동률} > \mathbf{0}) = \mathbf{1}$

Y
H금리대비
1
0
0
…

새 종속변수 '금리대비'

$f(\text{변동률} > \frac{\text{금리}}{\mathbf{365}}) = \mathbf{1}$

변환된 데이터셋

일별 데이터셋

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2670 entries, 2010-07-19 to 2021-05-14
Data columns (total 45 columns):
 #   Column      Non-Null Count  Dtype
---
 0   S금리대비       2670 non-null   int64
 1   H금리대비       2670 non-null   int64
 2   S등락         2670 non-null   int64
 3   H등락         2670 non-null   int64
 4   H등락분류       2670 non-null   int64
 5   S종가         2670 non-null   float64
 6   S등락률        2670 non-null   float64
 7   H종가         2670 non-null   float64
 8   H대비         2670 non-null   int64
 9   H등락률        2670 non-null   float64
 10  H시가         2670 non-null   float64
 11  H고가         2670 non-null   int64
 12  H저가         2670 non-null   int64
 13  H거래량        2670 non-null   int64
 14  H거래대금       2670 non-null   float64
 15  H시가총액       2670 non-null   float64
 16  H상장주식수      2670 non-null   int64
 17  원달러환율       2670 non-null   float64
 18  유가종가        2670 non-null   float64
 19  유가오픈        2670 non-null   float64
 20  유가고가        2670 non-null   float64
 21  유가저가        2670 non-null   float64
 22  유가변동률       2670 non-null   float64
 23  한국기준금리      2670 non-null   float64
 24  미국기준금리      2670 non-null   float64
 25  한미기준금리차이    2670 non-null   float64
 26  정기예금환산      2670 non-null   float64
 27  SOX종가       2670 non-null   float64
 28  SOX변동률      2670 non-null   float64
 29  BTC종가       2670 non-null   float64
 30  BTC변동률      2670 non-null   float64
 31  코스피종가       2670 non-null   float64
 32  코스피고가       2670 non-null   float64
 33  코스피저가       2670 non-null   float64
 34  코스피거래량      2670 non-null   float64
 35  코스피변동       2670 non-null   float64
 36  코스닥종가       2670 non-null   float64
 37  코스닥시가       2670 non-null   float64
 38  코스닥고가       2670 non-null   float64
 39  코스닥저가       2670 non-null   float64
 40  코스닥거래량      2670 non-null   float64
 41  코스닥변동       2670 non-null   float64
 42  SP종가        2670 non-null   float64
 43  SP변동률       2670 non-null   float64
 44  반도체수출금액지수   2670 non-null   float64
dtypes: float64(35), int64(10)
memory usage: 959.5 KB
```

▶ **일별 데이터셋의 독립변수 개수 44개**

데이터셋

연도별 데이터셋

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20 entries, 2001 to 2020
Data columns (total 26 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   S종가           20 non-null     int64
 1   S/ROE           20 non-null     float64
 2   S/ROA           20 non-null     float64
 3   S/영업이익      20 non-null     float64
 4   S/당기순이익    20 non-null     float64
 5   H종가           20 non-null     int64
 6   H등락률         20 non-null     float64
 7   H시가           20 non-null     int64
 8   H고가           20 non-null     int64
 9   H저가           20 non-null     int64
 10  H거래량         20 non-null     int64
 11  H거래대금       20 non-null     float64
 12  H시가총액       20 non-null     float64
 13  H상장주식수     20 non-null     int64
 14  H/ROA           20 non-null     float64
 15  H/ROE           20 non-null     float64
 16  H주당배당금     20 non-null     int64
 17  H배당수익률     20 non-null     float64
 18  H배당금총액     20 non-null     float64
 19  H/당기순이익    20 non-null     float64
 20  H매출액증가율   20 non-null     float64
 21  H/영업이익      20 non-null     float64
 22  반도체 수출금액지수  20 non-null  float64
 23  금리            20 non-null     float64
 24  WTI 현물유가등락률  20 non-null  float64
 25  원달러환율(원)  20 non-null     float64
dtypes: float64(18), int64(8)
memory usage: 4.2 KB
```

▶ **연도별 데이터셋의 독립변수 개수 25개**

탐색적 데이터 분석

※ EDA (Exploratory Data Analysis)

중요한 독립변수 위주

연도별 데이터셋

Ldata.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20 entries, 2001 to 2020
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   S종가       20 non-null     int64
 1   H종가       20 non-null     int64
 2   S/ROE     20 non-null     float64
 3   S/영업이익    20 non-null     float64
 4   S/ROA     20 non-null     float64
 5   S/당기순이익   20 non-null     float64
 6   H/ROE     20 non-null     float64
 7   H/영업이익    20 non-null     float64
 8   H/ROA     20 non-null     float64
 9   H/당기순이익   20 non-null     float64
dtypes: float64(8), int64(2)
memory usage: 1.7 KB
```

▶ 연도별 데이터셋의 데이터 개수는 20개, 기간은 2001년부터 2020년까지 존재한다.

탐색적 데이터 분석

※ EDA (Exploratory Data Analysis)

일별 데이터셋

중요한 독립변수 위주

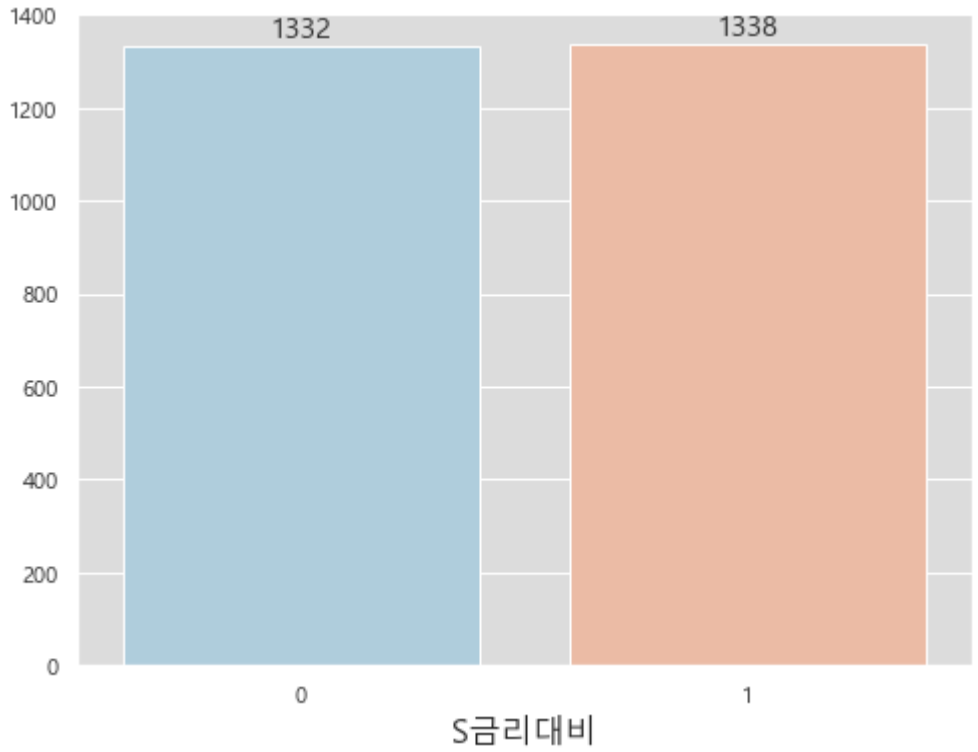
Sdata.info()

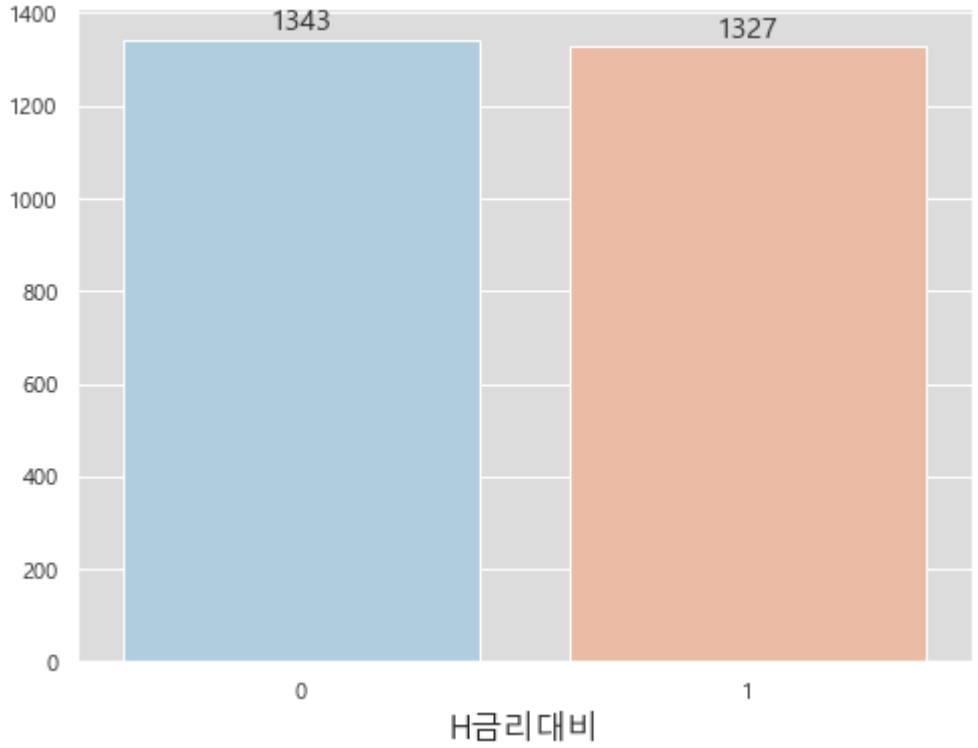
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2670 entries, 2010-07-19 to 2021-05-14
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   S금리대비       2670 non-null   int64
 1   H금리대비       2670 non-null   int64
 2   코스피변동        2670 non-null   float64
 3   코스닥변동        2670 non-null   float64
 4   S종가         2670 non-null   float64
 5   H종가         2670 non-null   float64
 6   유가종가        2670 non-null   float64
 7   한국기준금리      2670 non-null   float64
 8   SOX변동률      2670 non-null   float64
 9   반도체수출금액지수  2670 non-null   float64
dtypes: float64(8), int64(2)
memory usage: 229.5 KB
```

▶ 일별 데이터셋의 데이터 개수는 2670개, 기간은 2010년 7월 19일부터 2021년 5월 14일까지 이다.

탐색적 데이터 분석

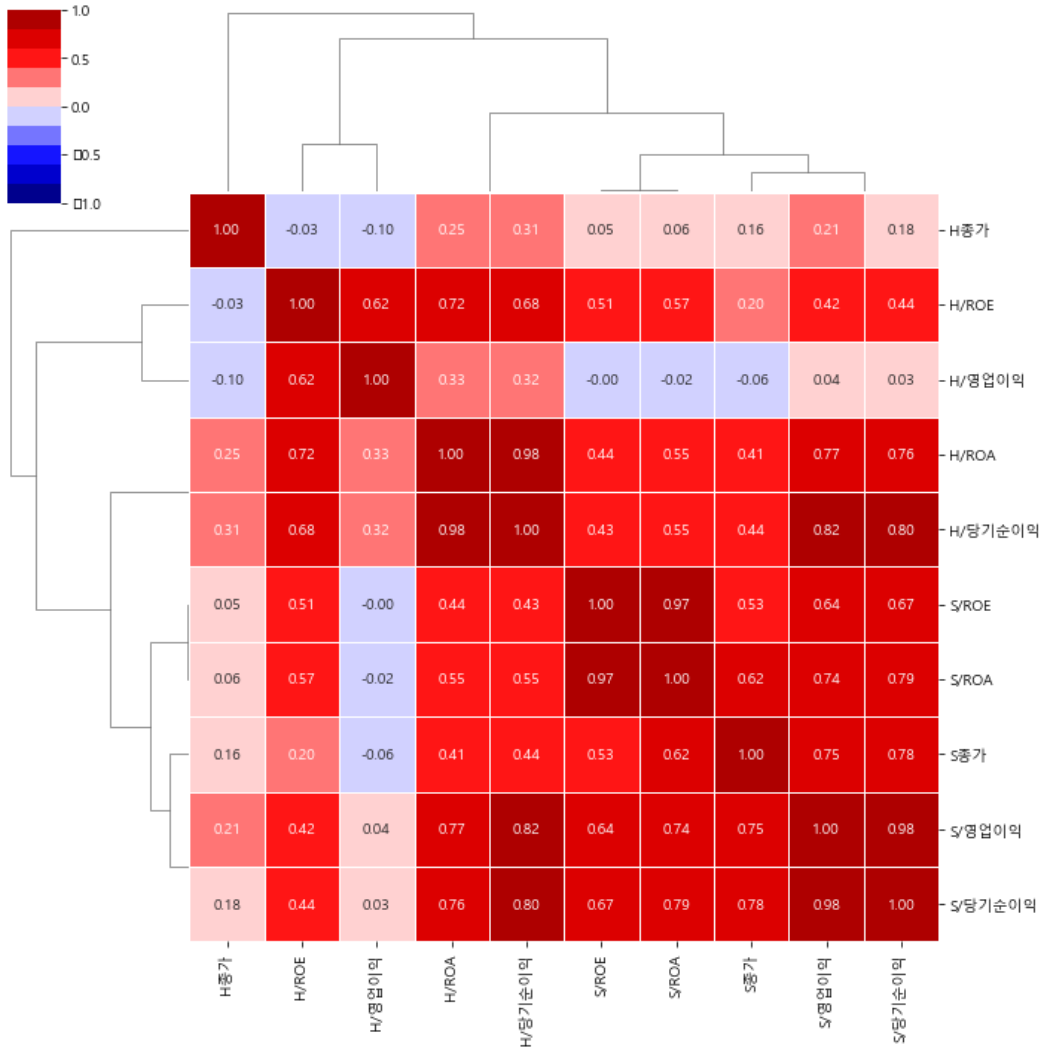
※ EDA (Exploratory Data Analysis)





▶ SK하이닉스 금리대비 및 삼성전자 금리대비 모두 각각의 등락 비율이 유사함을 알 수 있다.

상관관계 분석



연도별 데이터셋

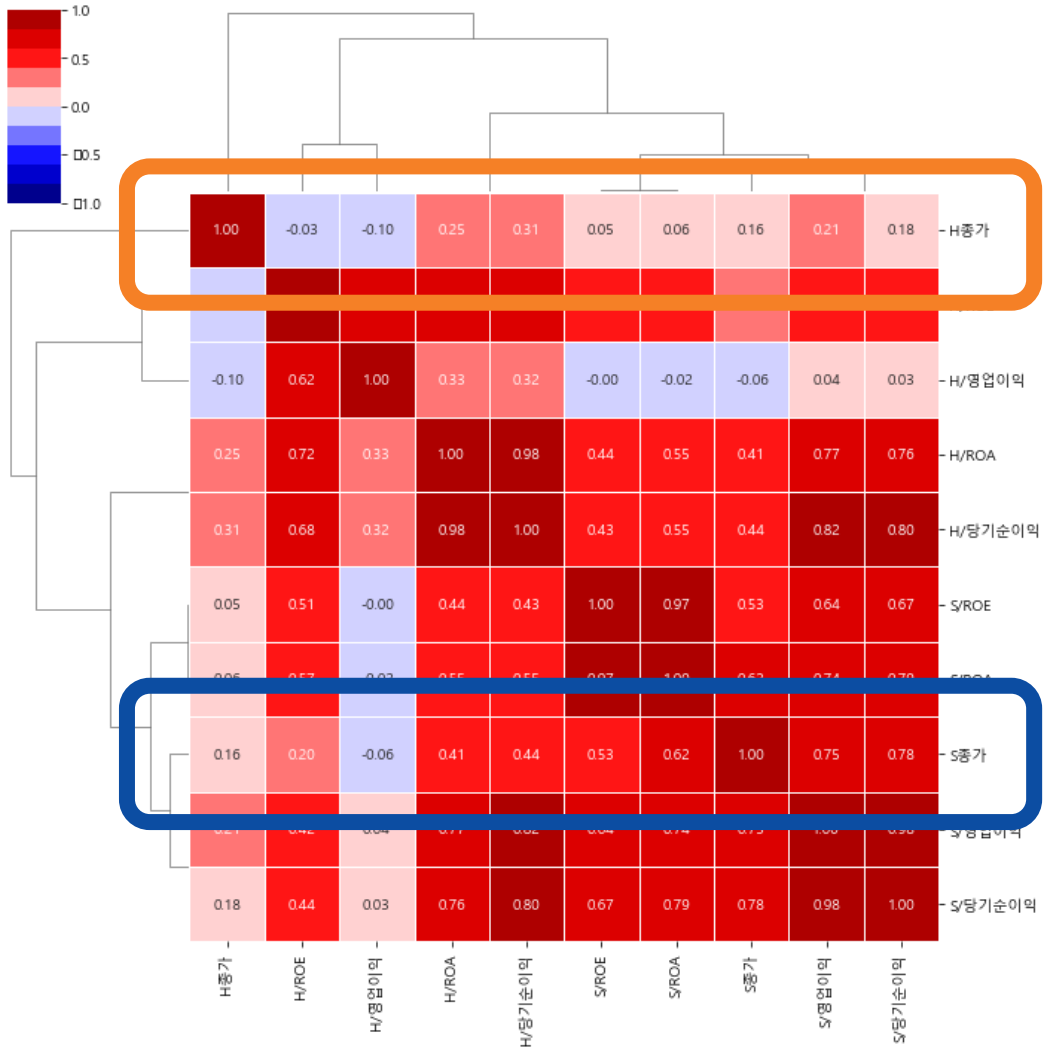
- 보간법을 적용하지 않음
- 시계열 데이터의 특징인 추세 존재

▶ **차분 진행 후 피어슨 상관 계수를 확인**

차분?

: 시계열 데이터 `처리 시 추세를 제거하여 기존 값을 유동성 값으로 변경함

상관관계 분석



연도별 데이터셋

SK하이닉스의 종가, ROE, ROA, 영업이익, 당기순이익 및

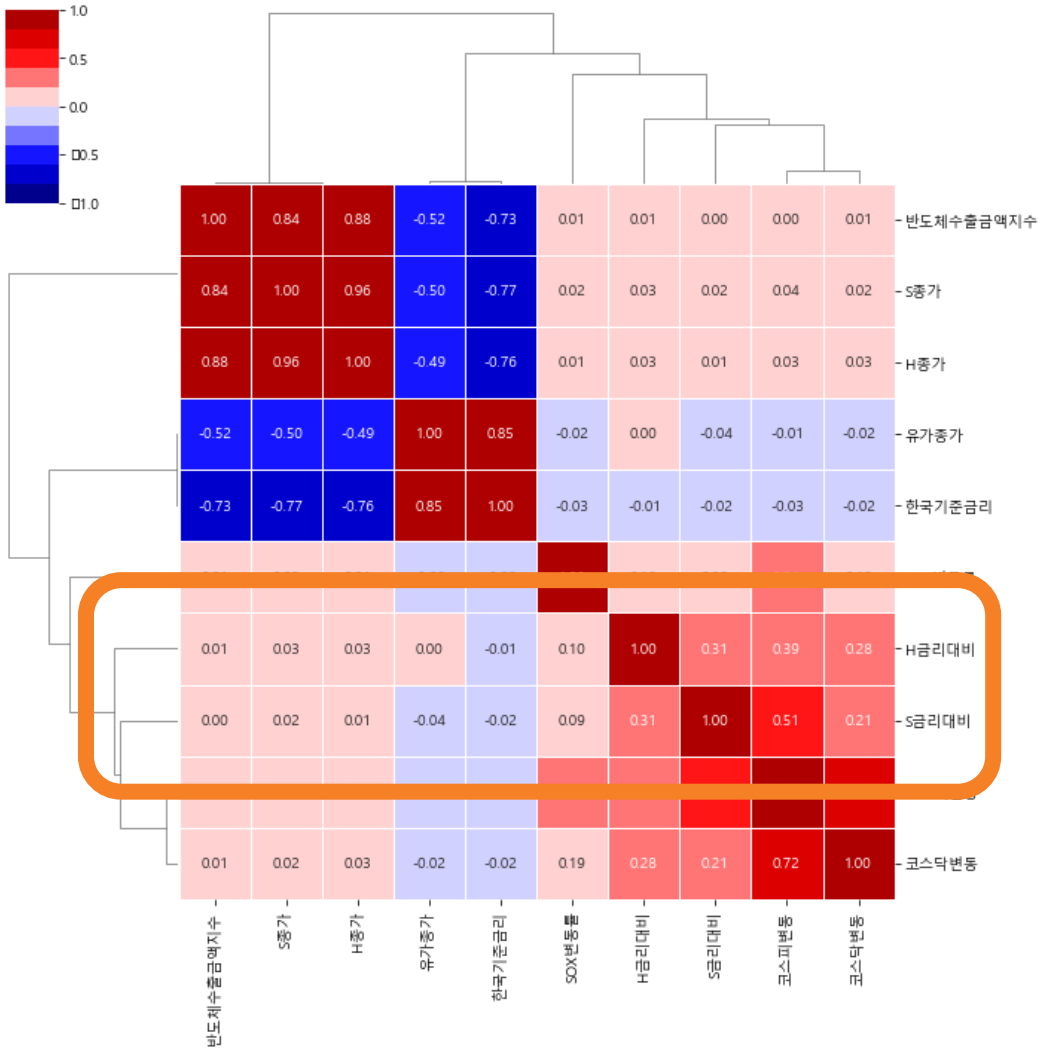
삼성전자의 종가, ROE, ROA, 영업이익, 당기순이익의 상관계수 확인

▶ **SK하이닉스 종가는 SK하이닉스의 당기순이익과 0.31의 상관성**

▶ **삼성전자는 삼성전자의 당기순이익과 0.78의 상관성**

▶ **서로의 당기순이익은 0.80의 높은 상관성**

상관관계 분석

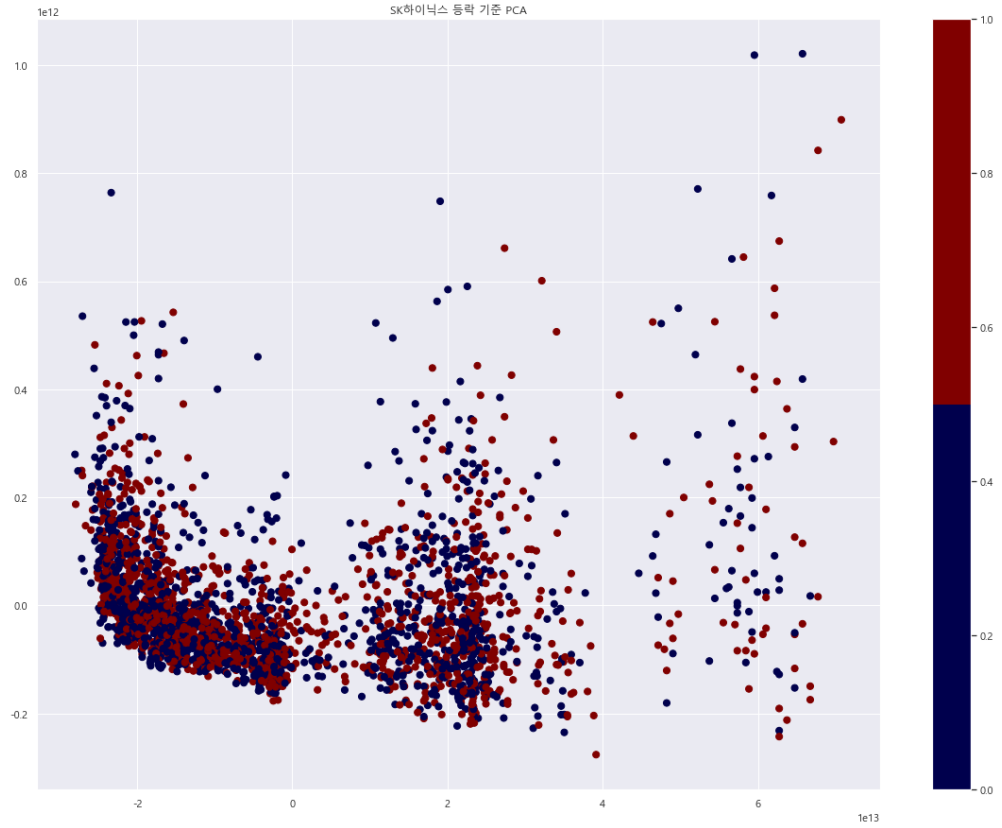


일별 데이터셋

SK하이닉스의 금리대비 등락 여부/
삼성전자의 금리대비 등락 여부 둘과
거시적 경제 지표들의 상관계수 확인

▶ **SK하이닉스와 삼성전자의 금리대비 등락여부 값은 코스피변동에 각각 0.39, 0.51의 상관성을 보인다.**

차원축소



PCA

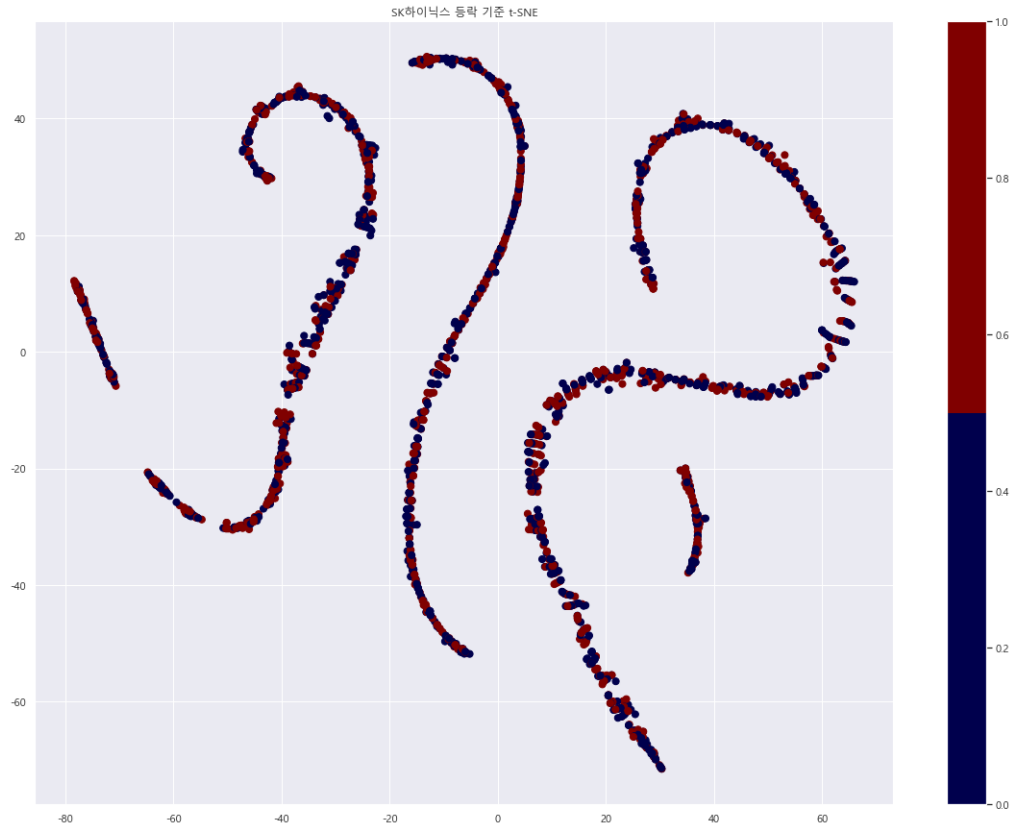
※ Principal Component Analysis

Dimension : 2

Plot : SK하이닉스 등락 기준 PCA

▶ **PCA 방식으로는 등락 여부가 잘 구분되질 않음을 알 수 있다.**

차원축소



t-SNE

※ t-Stochastic Neighbor Embedding

Dimension : 2

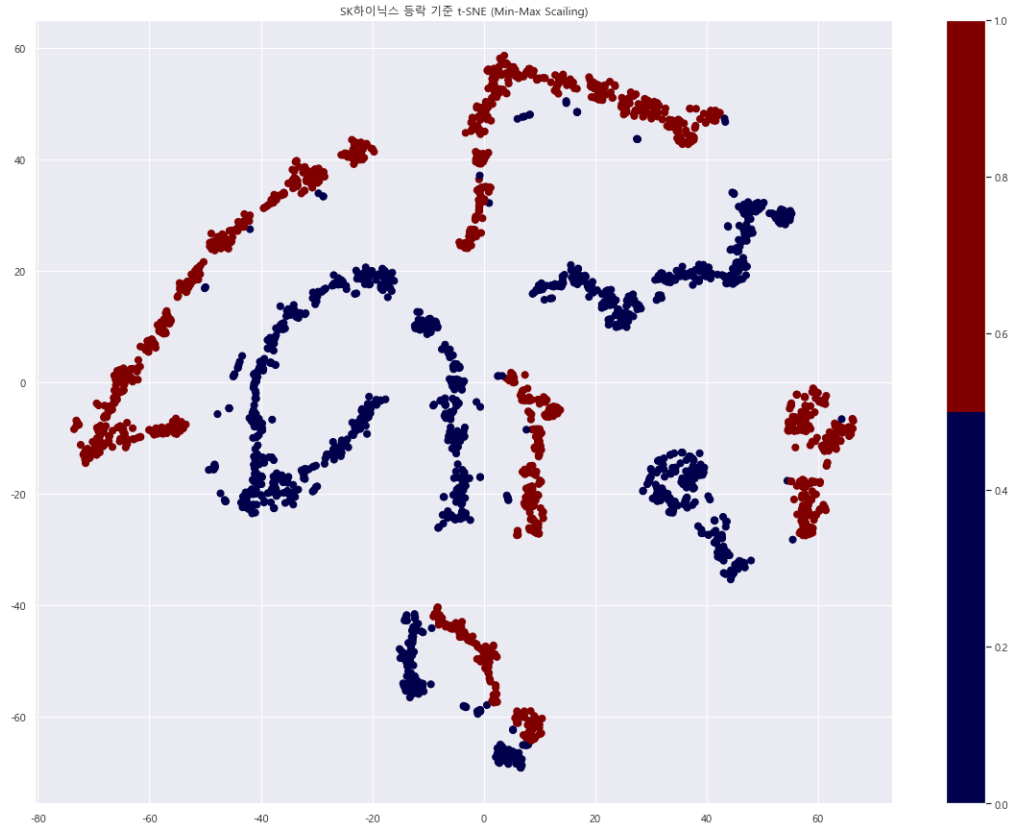
Plot : SK하이닉스 등락 기준 t-SNE

▶ **PCA방식보다 안정적인 임베딩 결과를 보인다.**

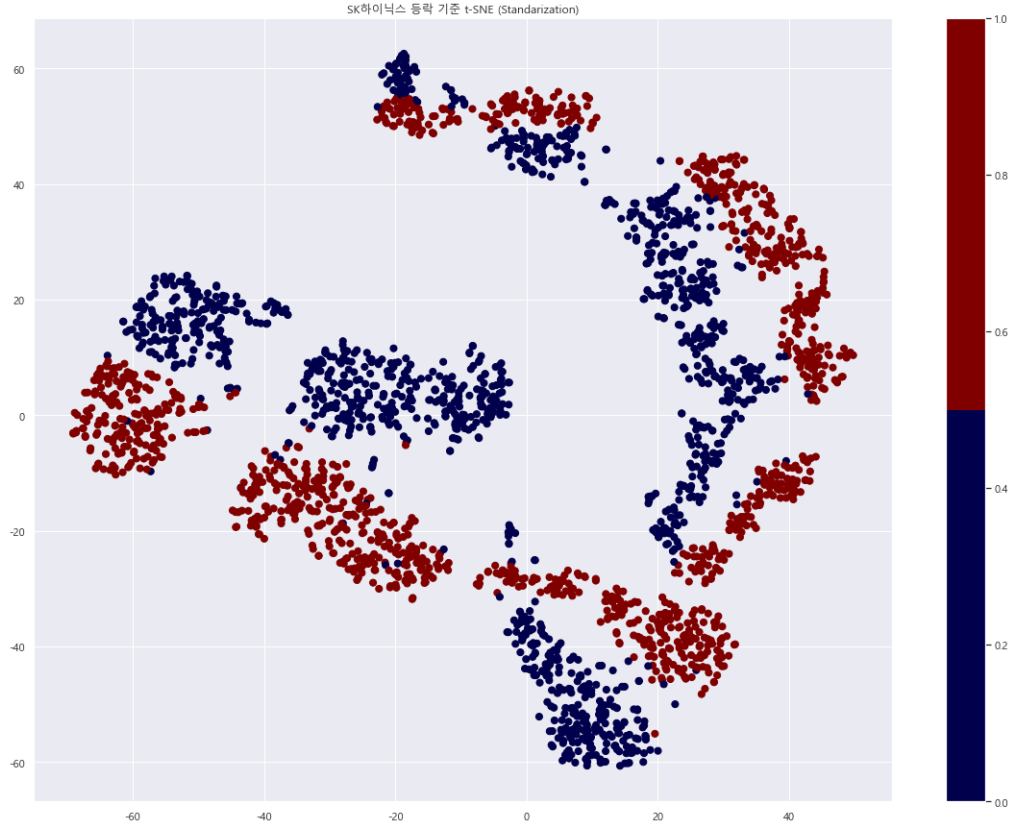
▶ **그럼에도 등락 여부가 잘 구분되질 않음을 알 수 있다.**

차원축소

```
tsne_result = tsne.fit_transform(df.drop(['H등락', 'H금리대비','S금리대비','S등락'], axis=1))
points = ax.scatter(tsne_result[:,0], tsne_result[:,1], c=df['H등락'], s=50, cmap=plt.cm.get_cmap('seismic', 2))
```



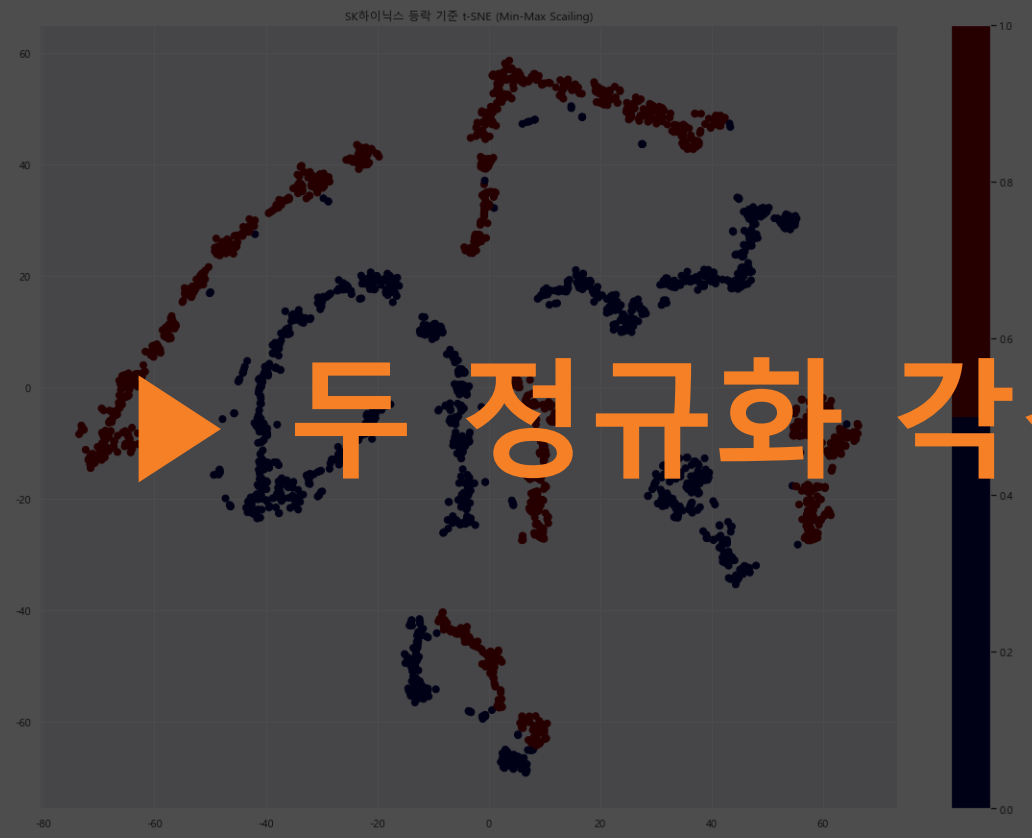
▶ Min-Max Scaling



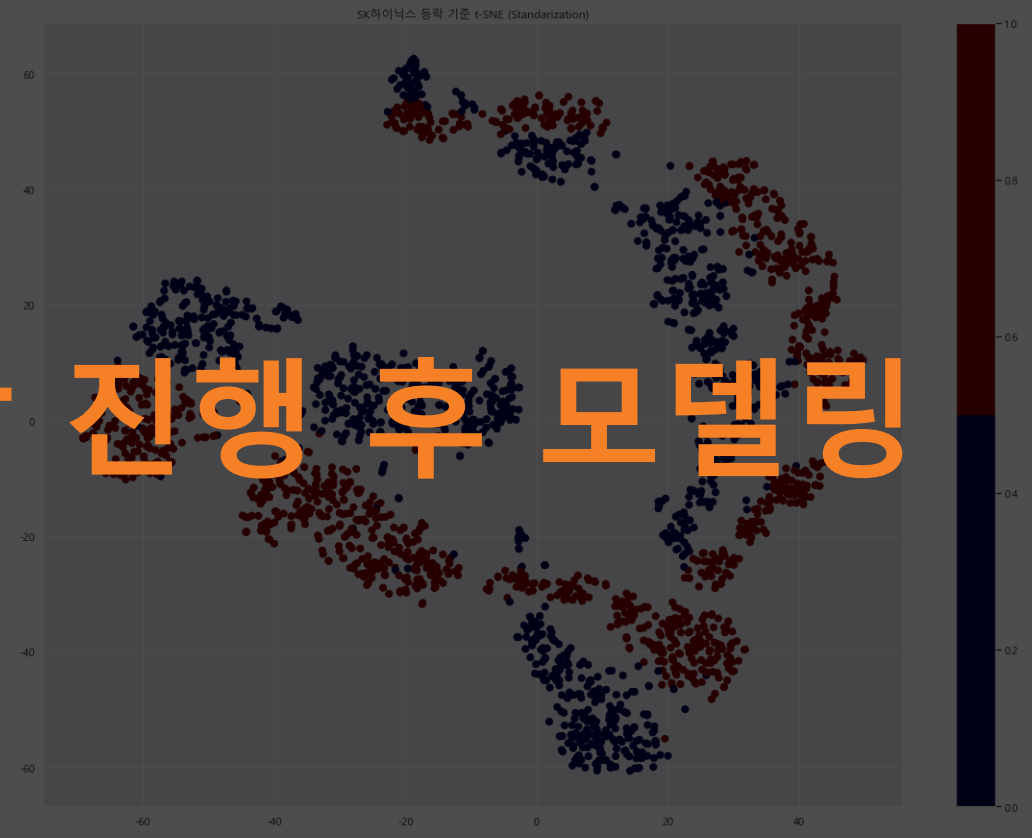
▶ Standardization

차원축소

```
tsne_result = tsne.fit_transform(df.drop(['H등락', 'H금리대비','S금리대비','S등락'], axis=1))
points = ax.scatter(tsne_result[:,0], tsne_result[:,1], c=df['H등락'], s=50, cmap=plt.cm.get_cmap('seismic', 2))
```



▶ Min-Max Scailing



▶ Standarization

▶ 두 정규화 각각 진행 후 모델링

3. 모델 예측

3.1 데이터셋 나누기

3.2 모델 하한선 구축

3.3 모델 예측

3.4 평가지표 비교

데이터셋 나누기

```
random_seed = np.random.seed(2021)
x_train, x_test, y_train, y_test = train_test_split(features, target, test_size=0.1, random_state=random_seed)
```

Date	S종가	H종가	유가
2010-07-19			
…			
2021-05-14			

X

H금리대비
1
0
0

Y1

S금리대비
0
1
0

Y2

Random Sampling

Train set : Test set = 0.9 : 0.1

모델 하한선(Baseline) 구축

Key point1: 모델 성능 최소 하한선 제공
Key point2: tight한 하한선으로 평가지표 생성

ML Baseline,

Logistic Regression

[Vanilia] SK하이닉스 정확도: 46.81%, 삼성전자 정확도: 49.81%

[Min-Max Scailing] SK하이닉스 정확도: 61.42%, 삼성전자 정확도: 75.65%

[Standardization] SK하이닉스 정확도: 64.04%, 삼성전자 정확도: 78.2%

모델 하한선(Baseline) 구축

Key point1: 모델 성능 최소 하한선 제공
Key point2: tight한 하한선으로 평가지표 생성

▶ **Standardization** 적용

ML Baseline,

Logic Regression을 하한선

(정확도 64.04%, 78.2%)

[Vanilla] SK하이닉스 정확도: 46.81%, 삼성전자 정확도: 49.81%

[Min-Max Scailing] SK하이닉스 정확도: 61.42%, 삼성전자 정확도: 75.65%

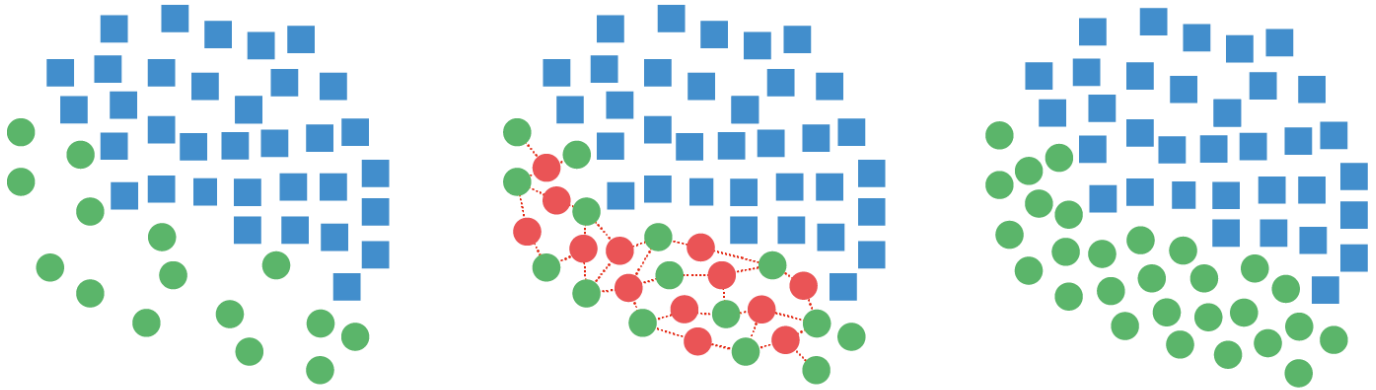
[Standardization] SK하이닉스 정확도: 64.04%, 삼성전자 정확도: 78.2%

AutoML

※ 여러 학습 모델 자동화 프로세스

Google Colab Library Pycaret

SMOTE ※Synthetic Minority Oversampling technique



AutoML을 통한 모델 학습 시 SMOTE 자동 적용

▶ 데이터 개수가 적은 종속변수 값의 표본을 가져와 임의의 값 추가 후 새로운 데이터를 만들어 기존 데이터에 추가하는 Over Sampling 방식

AutoML

※ 여러 학습 모델 자동화 프로세스

삼성전자
금리대비 예측

SMOTE + Standardization AutoML결과

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	0.7548	0.8446	0.7633	0.7510	0.7562	0.5096	0.5111	0.315
ridge	Ridge Classifier	0.7479	0.0000	0.7601	0.7422	0.7504	0.4957	0.4968	0.018
lda	Linear Discriminant Analysis	0.7473	0.8404	0.7601	0.7414	0.7501	0.4947	0.4957	0.022
gbc	Gradient Boosting Classifier	0.7323	0.8246	0.7516	0.7236	0.7365	0.4647	0.4663	0.676
rf	Random Forest Classifier	0.7297	0.8118	0.7312	0.7304	0.7292	0.4593	0.4613	0.766
ada	Ada Boost Classifier	0.7291	0.8149	0.7548	0.7185	0.7354	0.4582	0.4601	0.225
lightgbm	Light Gradient Boosting Machine	0.7190	0.8152	0.7259	0.7169	0.7207	0.4379	0.4389	0.258
svm	SVM - Linear Kernel	0.7018	0.0000	0.6896	0.7159	0.6976	0.4036	0.4089	0.022
et	Extra Trees Classifier	0.7018	0.7736	0.7045	0.7018	0.7020	0.4036	0.4049	0.567
knn	K Neighbors Classifier	0.6697	0.7192	0.6725	0.6698	0.6699	0.3395	0.3408	0.120
dt	Decision Tree Classifier	0.6617	0.6617	0.6605	0.6630	0.6605	0.3233	0.3245	0.037
qda	Quadratic Discriminant Analysis	0.6510	0.7392	0.8128	0.6174	0.6990	0.3022	0.3237	0.021
nb	Naive Bayes	0.6370	0.7268	0.8171	0.6053	0.6908	0.2744	0.2997	0.017

AutoML

※ 여러 학습 모델 자동화 프로세스

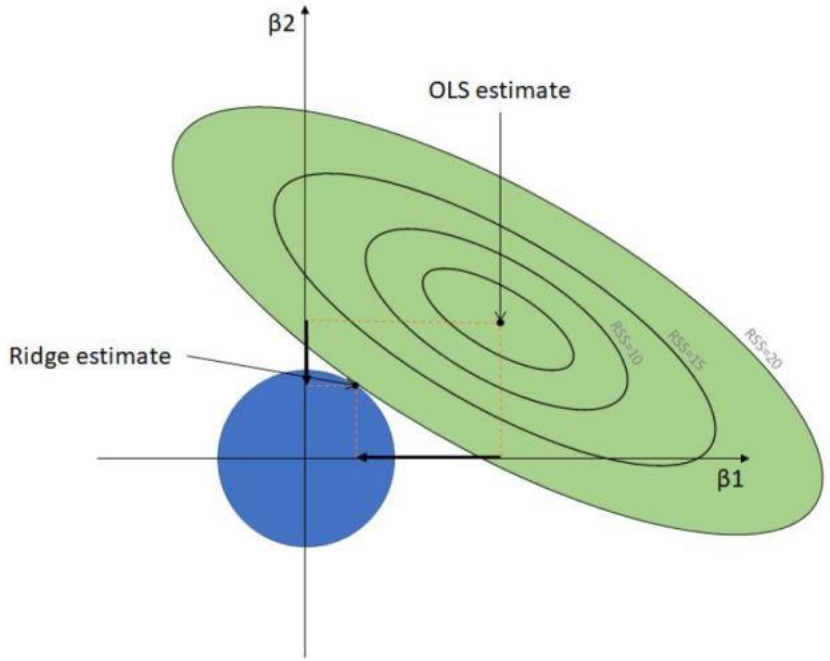
SMOTE + Standardization AutoML결과

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	0.7548	0.8446	0.7633	0.7510	0.7562	0.5096	0.5111	0.315
ridge	Ridge Classifier	0.7479	0.0000	0.7601	0.7422	0.7504	0.4957	0.4968	0.018
lda	Linear Discriminant Analysis	0.7473	0.8404	0.7601	0.7414	0.7501	0.4947	0.4957	0.022
gbc	Gradient Boosting Classifier	0.7373	0.8246	0.7579	0.7236	0.7365	0.4647	0.4663	0.676
rf	Random Forest Classifier	0.7297	0.8118	0.7312	0.7304	0.7292	0.4593	0.4613	0.766
ada	Ada Boost Classifier	0.7291	0.8149	0.7548	0.7185	0.7354	0.4582	0.4601	0.225
lightgbm	Light Gradient Boosting Machine	0.7190	0.8152	0.7259	0.7169	0.7207	0.4379	0.4389	0.258
svm	SVM - Linear Kernel	0.7018	0.0000	0.6896	0.7159	0.6976	0.4036	0.4089	0.022
et	Extra Trees Classifier	0.7018	0.7736	0.7045	0.7018	0.7020	0.4036	0.4049	0.567
knn	K Neighbors Classifier	0.6697	0.7192	0.6725	0.6698	0.6699	0.3395	0.3408	0.120
dt	Decision Tree Classifier	0.6617	0.6617	0.6605	0.6630	0.6605	0.3233	0.3245	0.037
qda	Quadratic Discriminant Analysis	0.6510	0.7392	0.8128	0.6174	0.6990	0.3022	0.3237	0.021
nb	Naive Bayes	0.6370	0.7268	0.8171	0.6053	0.6908	0.2744	0.2997	0.017

▶ **정확도 상위 4개 모델 선택**

삼성전자
금리대비 예측

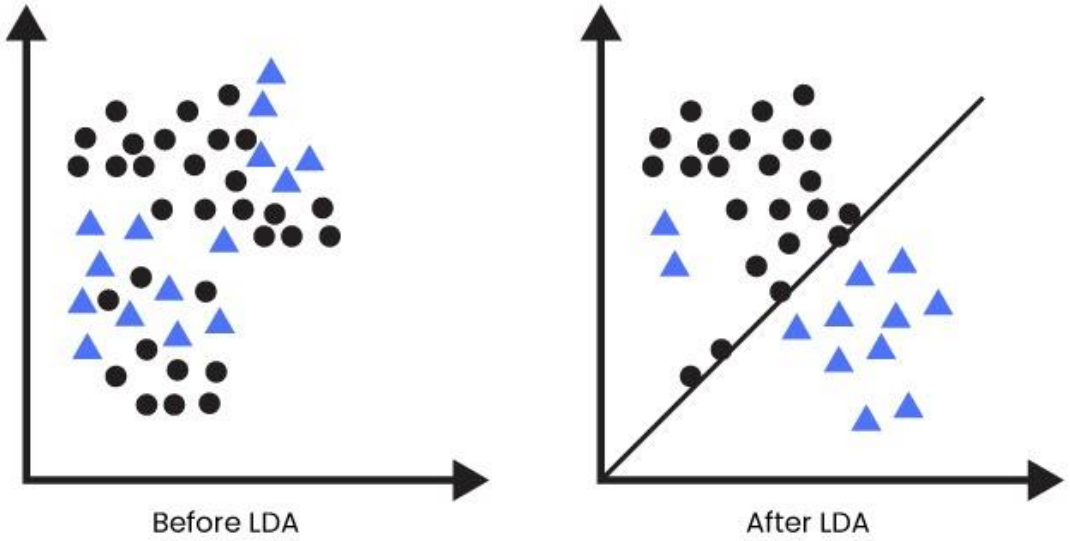
Ridge Classifier



Ridge Classifier은 기존 Ridge Regression을 Classification 모델로 변경

▶ **[Standardization] SK하이닉스 정확도: 64.41%, 삼성전자 정확도: 78.65%**

LDA ※Linear Discriminant Analysis



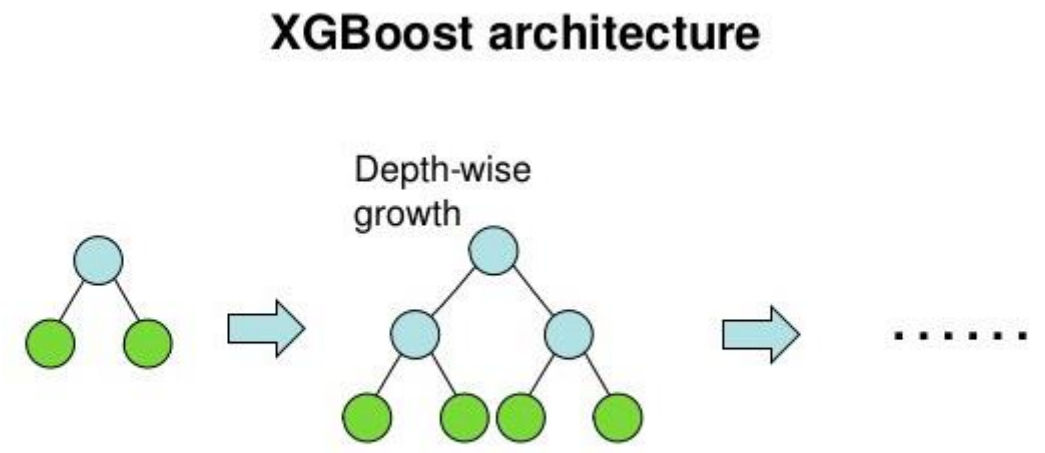
LDA는 지도학습의 분류에 사용되는 차원 축소 방법이며,

종속변수의 0, 1를 최대한 분리할 수 있는 축을 구축하는 방법

▶ **[Standardization] SK하이닉스 정확도: 64.04%, 삼성전자 정확도: 78.27%**

XGBoost + Optimization

XGBoost: Kaggle에서 Boosting 모델 중 평가지표 GBC보다 상위권
Hyperparameter Optimization: 모델 하이퍼파라미터 최적화로 성능 향상



XGBoost는 CART(Classification and Regression Tree)기반 앙상블 모델

모델 내부에 과적합 제약식이 존재, Depth-Wise(=Level-wise) 알고리즘 채택

XGBoost + Optimization

XGBoost: Kaggle에서 Boosting 모델 중 평가지표 GBC보다 상위권
Hyperparameter Optimization: 모델 하이퍼파라미터 최적화로 성능 향상

1. GridSearch 방식

▶ 모든 hyperparameter의 경우의 수에 대하여 cross-validation 결과가 가장 좋은 수를 선택

장점: 전역적 탐색 가능

단점: Hyperparameter의 개수에 따라 탐색 시간 기하급수적 증가

2. Bayesian TPE 방식

※Tree-structured Parzen Estimator

▶ 어느 입력값(x)를 받는 미지의 목적 함수(f(x))를 상정하여,
그 함숫값(f(x))을 최대로 만드는 최적해를 찾는 방법

장점 : 적은 시간으로 최적해 찾기 가능

단점 : 사용하는 함수에 따라 성능이 크게 변화함

XGBoost + Optimization

XGBoost: Kaggle에서 Boosting 모델 중 평가지표 GBC보다 상위권
Hyperparameter Optimization: 모델 하이퍼파라미터 최적화로 성능 향상

1. GridSearch 방식 평가지표

▶ [Standardization] SK하이닉스 정확도: 72.78%, 삼성전자 정확도: 82.01%

2. Bayesian TPE 방식 평가지표

※Tree-structured Parzen Estimator

▶ [Standardization] SK하이닉스 정확도: 78%, 삼성전자 정확도: 87%

CNN

※ Convolutional Neural Network

```
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam

model = Sequential()
```

Layer 3개,
활성화함수 Relu
Dropout 설정 X

```
model.add(Dense(64,input_shape=(29,),activation='relu'))
model.add(Dense(64,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
```

```
model.compile(loss='binary_crossentropy',
              optimizer='Adam',
              metrics=['accuracy'])
```

손실함수 Binary_crossEntropy
옵티마이저 Adam
평가지표 정확도
Epochs = 100

▶ **SK하이닉스 정확도: 58.05%, 삼성전자 정확도: 77.53%**

MTL ※Multi-Task Learning

MTL는 서로 연관성이 있는 Task를 동시에 학습하여

Task 수행 성능을 전반적으로 향상시키는 학습 패러다임

Layer 3개,
활성화함수 Relu
Dropout 설정 X

```
Model = Sequential()
model.add(Dense(128, input_shape(29,), activation = 'relu', kernel_initializer='random_uniform'))
model.add(Dense(128, input_shape(29,), activation = 'relu', kernel_initializer='random_uniform'))
model.add(Dense(128, activation='relu', kernel_initializer='normal'))
Model.add(Dense(data_y.shape[1], activation='sigmoid', kernel_initializer='random_uniform'))

Model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

Model.fit(data_x, data_y, epochs=i, batch_size=j, validation_split=0.1, verbose=2)
```

손실함수 Binary_crossEntropy
옵티마이저 Adam
평가지표 정확도

epochs = [32, 64, 128]
batch_size = [8, 16, 32]

MTL ※Multi-Task Learning

```
Epoch 117/128
76/76 - 0s - loss: 0.6932 - accuracy: 0.8294 - val_loss: 0.6932 - val_accuracy: 0.8427
Epoch 118/128
76/76 - 0s - loss: 0.6932 - accuracy: 0.8294 - val_loss: 0.6932 - val_accuracy: 0.8427
Epoch 119/128
76/76 - 0s - loss: 0.6932 - accuracy: 0.8294 - val_loss: 0.6932 - val_accuracy: 0.8427
Epoch 120/128
76/76 - 0s - loss: 0.6932 - accuracy: 0.8294 - val_loss: 0.6932 - val_accuracy: 0.8427
Epoch 121/128
76/76 - 0s - loss: 0.6932 - accuracy: 0.8294 - val_loss: 0.6932 - val_accuracy: 0.8427
Epoch 122/128
76/76 - 0s - loss: 0.6932 - accuracy: 0.8294 - val_loss: 0.6932 - val_accuracy: 0.8427
Epoch 123/128
76/76 - 0s - loss: 0.6932 - accuracy: 0.8294 - val_loss: 0.6932 - val_accuracy: 0.8427
Epoch 124/128
76/76 - 0s - loss: 0.6932 - accuracy: 0.8294 - val_loss: 0.6932 - val_accuracy: 0.8427
Epoch 125/128
76/76 - 0s - loss: 0.6932 - accuracy: 0.8294 - val_loss: 0.6932 - val_accuracy: 0.8427
Epoch 126/128
76/76 - 0s - loss: 0.6932 - accuracy: 0.8294 - val_loss: 0.6932 - val_accuracy: 0.8427
Epoch 127/128
76/76 - 0s - loss: 0.6932 - accuracy: 0.8294 - val_loss: 0.6932 - val_accuracy: 0.8427
Epoch 128/128
76/76 - 0s - loss: 0.6932 - accuracy: 0.8294 - val_loss: 0.6932 - val_accuracy: 0.8427
INFO:tensorflow:Assets written to: model_128_32/assets
```

▶ **SK하이닉스 정확도: 82.94%, 삼성전자 정확도: 84.27%**

모델 예측 평가지표

베이스라인 모델	SK하이닉스 정확도	삼성전자 정확도
Standardization + Logistic Regression	64.04%	78.2%

	SK하이닉스 정확도	삼성전자 정확도
Ridge Classifier	**64.41%**	**78.65%**
Linear Discriminant Analysis	**64.04%**	**78.27%**
XGBoost	**78%**	**87%**
Convolutional Neural Network	58.05%	77.53%
MTL	**82.94%**	**84.27%**

모델 예측 평가지표

베이스라인 모델	SK하이닉스 정확도	삼성전자 정확도
Standardization + Logistic Regression	64.04%	78.2%

▶ 대시보드에는 최종적으로 XGBoost 모델 적용

	SK하이닉스 정확도	삼성전자 정확도
Ridge Classifier	**64.41%**	**78.65%**
Linear Discriminant Analysis	**64.04%**	**78.27%**
XGBoost	**78%**	**87%**
Convolutional Neural Network	58.05%	77.53%
MTL	**82.94%**	**84.27%**

4. 대시보드

4.1 대시보드 기획 개요

4.2 대시보드 제작

4.3 대시보드 최종

대시보드 기획 개요

모델링을 통한 전략

+ 차트를 통한 분석 대시보드

기획 배경 및 타겟

반도체 우량주에 '투자'하려는 **20대 청년들**에게

분석 결과를 통하여 **전략적 투자**를 도와줄 수 있는 대시보드

주요 기능

모델링으로 주가 예측 + Drill Down으로 주식 투자 안정성 확인 + 영향력 있는 지수 확인

기대 효과

우량주라는 이유로 사실상 투기에 가까운 투자가 아닌, **전략적 투자 가능**

대시보드 페르소나

24살 군대 적금을 막 깬 대학생(남)

22살 아르바이트를 하고 있는 대학생(여)

공통점

주식이 두려움 / 주식 입문자, 또는 주식 경험 없음(관심 有)
/ 시드머니 100만원 이상 / 수익률은 금리보다 높길 바람

▶ 페르소나 특징을 반영하여 **대시보드 기능 정의**

대시보드 스타일

대표 캐릭터

남성 페르소나

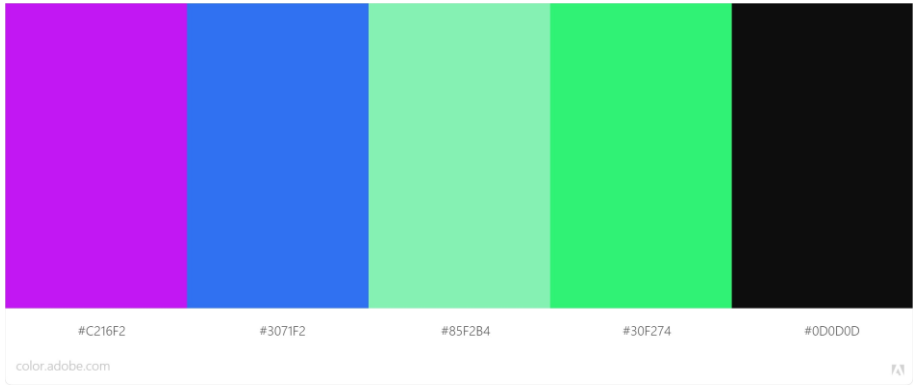
여성 페르소나

상업용 폰트

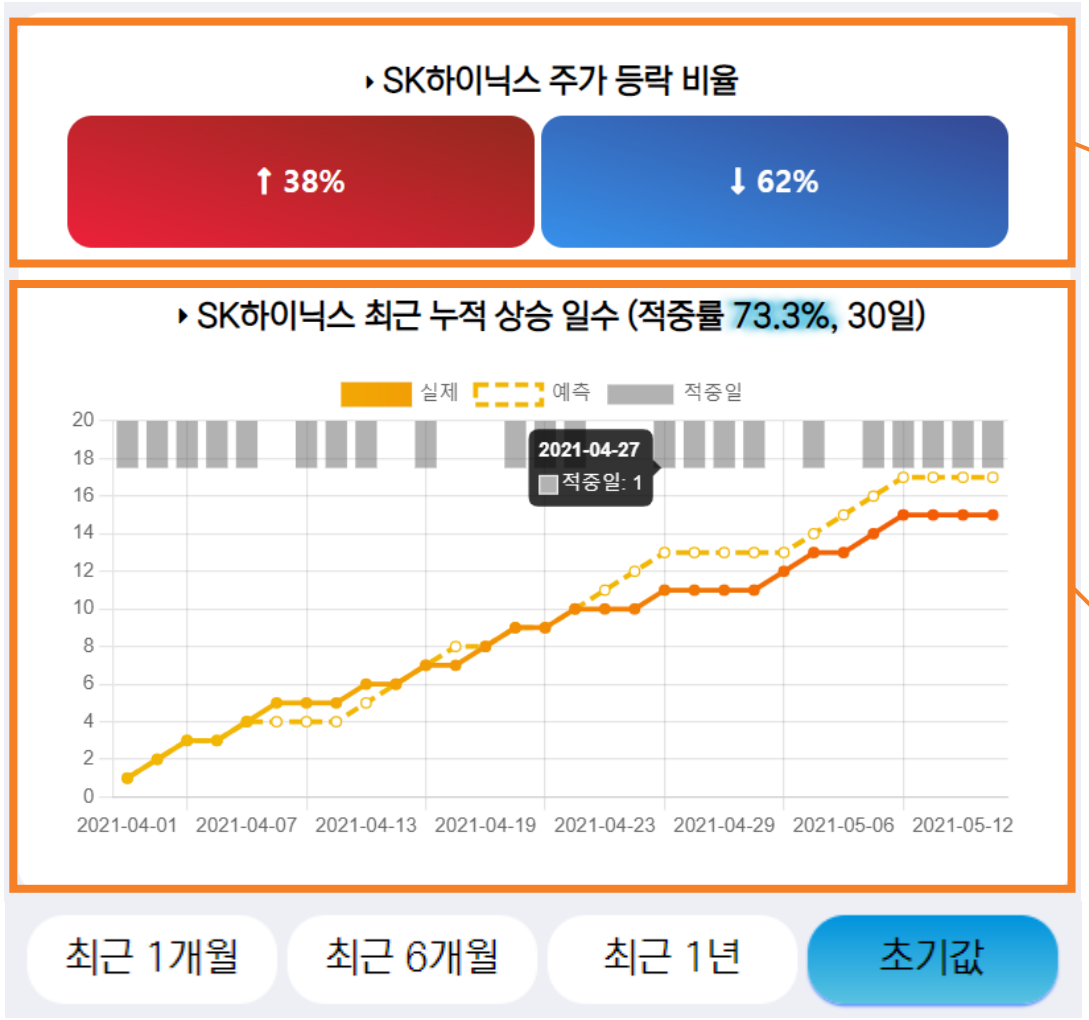
주제: 카페24 아네모네

내용: 넥슨 Lv.2 고딕 Bold

색상 차트



대시보드 제작



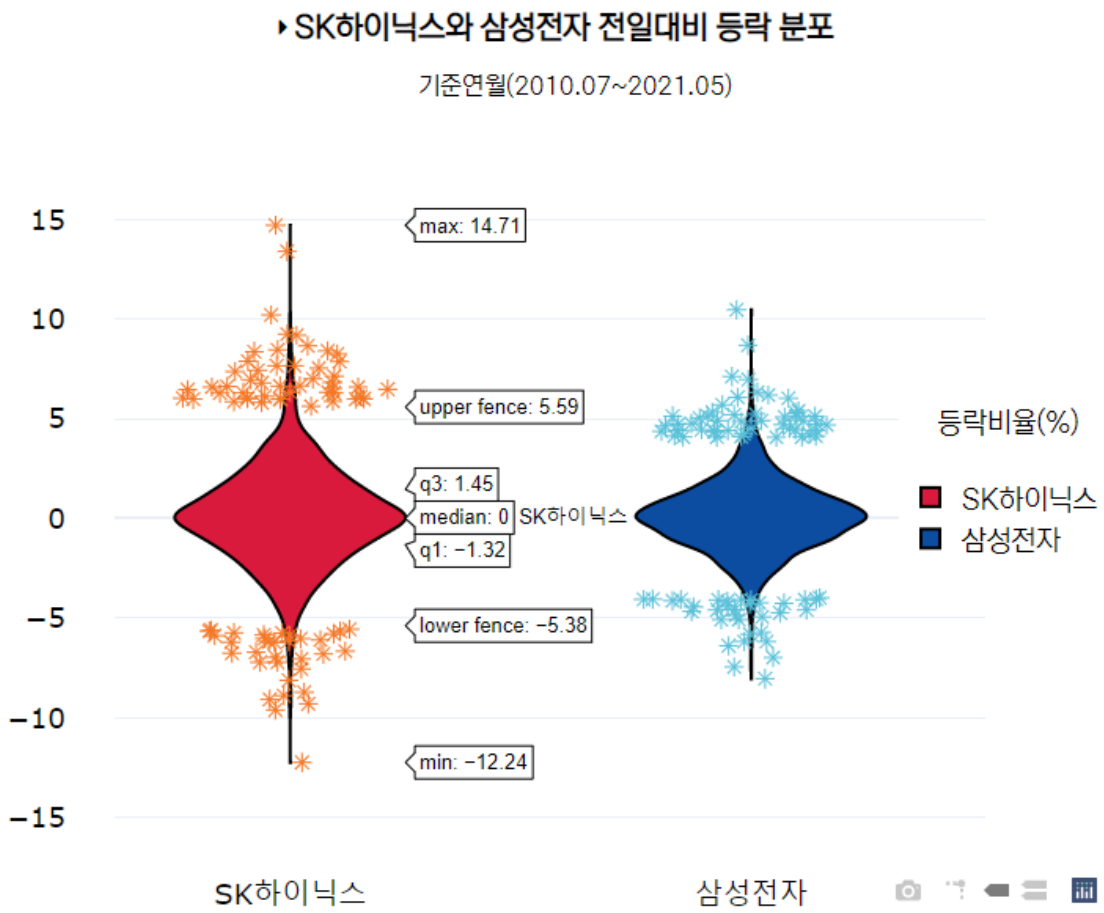
▶ 안정성 지표

하단 버튼을 통하여 각 종목의 일정 기간 동안 상승(빨강), 하락(파랑)의 비율 확인 가능

▶ 머신러닝을 통한 주가 등락예측

XGBoost 모델을 통하여
등락 예측 결과를 라인 차트로 표현
상단의 회색 바는
모델이 적중했는지에 대한
여부를 알 수 있음

대시보드 제작

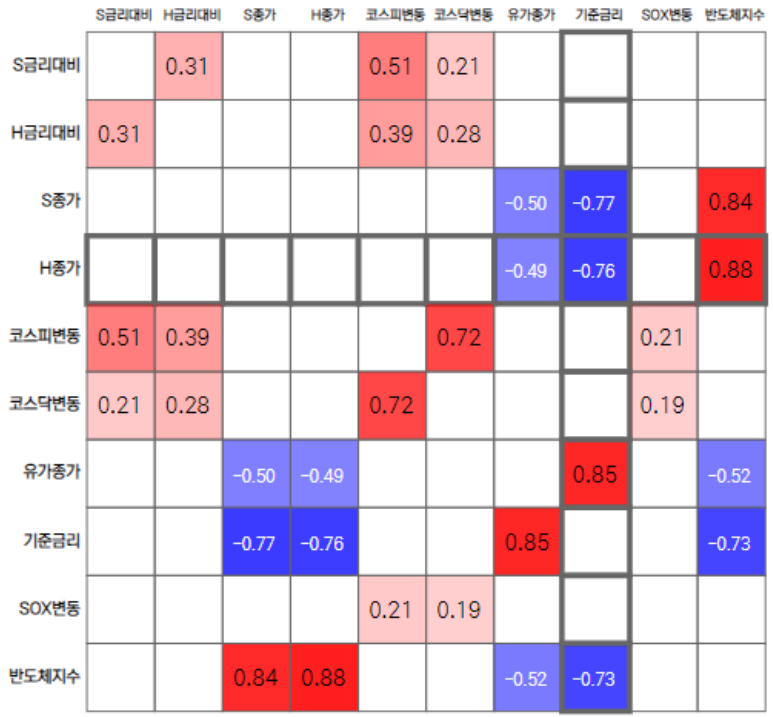


▶ 전체 등락 분포

SK하이닉스(빨강), 삼성전자(파랑)의
전체 기간 동안 등락 %로
얼마나 오르내렸는지 확인 가능

대시보드 제작

▸주가관련지수 외부요인 상관성 분석



	S금리대비	H금리대비	S종가	H종가	코스피변동	코스닥변동	유가종가	기준금리	SOX변동	반도체지수
S금리대비		0.31			0.51	0.21				
H금리대비	0.31				0.39	0.28				
S종가							-0.50	-0.77		0.84
H종가							-0.49	-0.76		0.88
코스피변동	0.51	0.39				0.72			0.21	
코스닥변동	0.21	0.28			0.72				0.19	
유가종가			-0.50	-0.49				0.85		-0.52
기준금리			-0.77	-0.76			0.85			-0.73
SOX변동					0.21	0.19				
반도체지수			0.84	0.88			-0.52	-0.73		

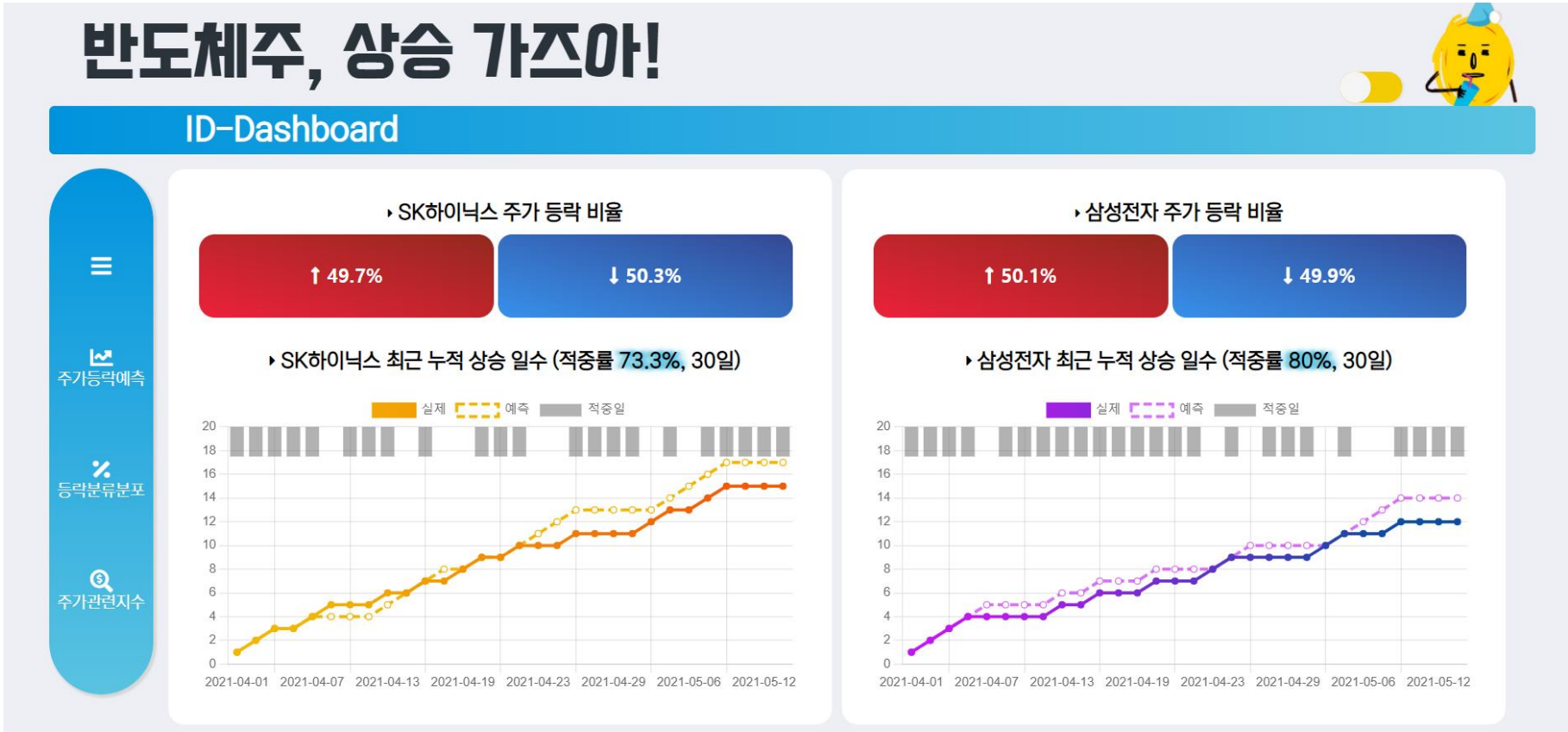
▶ 주가 상관성 지수 차트

두 데이터셋의 지표를 통하여
주가와 상관성을 보이는 지수 확인

양의 상관성(빨강), 음의 상관성(파랑)

대시보드 최종

https://iddashboard.github.io/



결론

프로젝트 결론

데이터를 다방면으로 수집하고,
전처리를 진행하며
예측 모델을 구축하고
최적화를 통하여 모델의 성능을 향상시켰습니다.

또한, 모델 구축에서 그치지 않고
실제 대시보드를 만들어
모델을 적용한 뒤
호스팅하는 과정까지 겪었습니다.

아쉬운 점도 분명히 존재하지만,
이러한 경험을 쌓을 수 있도록 기회를 주신 교수님들과 학우 분들
모두에게 감사 인사를 드립니다.

이상 발표를 마칩니다.

Q&A

Reference

Slide 6_논문 : 장은아, 최회련, 이홍철(2020), "BERT를 활용한 뉴스 감성분석과 거시경제지표 조합을 이용한 주가지수 예측"

Slide 12_선형보간법_그림 : https://ko.wikipedia.org/wiki/%EC%84%A0%ED%98%95_%EB%B3%B4%EA%B0%84%EB%B2%95
Slide 12_후행보간법_그림 : https://maelfabien.github.io/statistics/TimeSeries5/
Slide 34_SMOTE_그림 : https://john-analyst.medium.com/smote%EB%A1%9C-%EB%8D%B0%EC%9D%B4%ED%84%B0-%EB%B6%88%EA%B7%A0%ED%98%95-%ED%95%B4%EA%B2%B0%ED%95%98%EA%B8%B0-5ab674ef0b32
Slide 37_Ridge Classifier_그림 : https://deepai.org/machine-learning-glossary-and-terms/ridge-regression
Slide 38_LDA_그림 : https://www.analyticssteps.com/blogs/introduction-linear-discriminant-analysis-supervised-learning
Slide 39_XGBoost_그림 : https://medium.com/analytics-vidhya/introduction-to-xgboost-algorithm-d2e7fad76b04
Slide 49_캐릭터 : https://www.sktinsight.com/109145

대시보드 아이콘 : https://fontawesome.com/
대시보드 폰트 : https://noonnu.cc/
전체 테마 그림 : https://stock.adobe.com/kr/

데이터셋 출처 : 딥서치, KRX, e-나라지표, K-stat, SK hynix 홈페이지, Ecos 한국은행 경제통계시스템, fred 사이트
데이터마이닝 프로젝트 code(github) : https://github.com/Data-mining-Information-design/TEAM-ID
정보디자인 프로젝트 code(github) : https://github.com/IDdashboard/IDdashboard.github.io