비트코인 가격 예측 모델

: AI, 내 코인 오를 수 있는 거지?

20163 10930 신성구

목 차

1. 개요

- 연구 개요
- 데이터 소개
-데이터전처리 및 EDA

연구 개요

비트코인은 주식과는 달리 국제 정세/사회적 이슈에 대해 반응하는 가격변동의 폭이 극단적으로 발생하기 때문에, 보다 안정적인 투자와 극심한 경제 위기의 대비를 위해 비트코인 가격을 예측하는 모델의 연구와 개선이 필요

따라서, 본 연구에서는 2017년 ~ 2021년 동안의 비트코인 가격 데이터를 활용하여, 비트코인의 가격을 예측하고, 가격에 영향을 주는 요인으로는 어떤 것들이 존재하는 지 파악하고자 함.

예측 + 가격에 영향을 주는 변수와 그 영향력을 파악하는 것이 본 연구의 목적이므로 ,

변수의 중요도(영향력)를 파악할 수 있는 Linear Regression 및 Random Forest 모델을 활용할 예정

데이터 소개

출처 : https://kr.investing.com

```
# 데이터셋 로드
bt = pd.read_csv('../data/bt_history.csv', index_col = 'Date', parse_dates = True)
gld = pd.read_csv('../data/GLD_price.csv', index_col = 'Date', parse_dates = True)
nk = pd.read_csv('../data/nikkei.csv', index_col = 'Date', parse_dates = True)
sh = pd.read_csv('../data/shanghai.csv', index_col = 'Date', parse_dates = True)
kospi = pd.read_csv('../data/kospi.csv', index_col = 'Date', parse_dates = True)
nasdaq = pd.read_csv('../data/nasdaq.csv', index_col = 'Date', parse_dates = True)
```

- bt_history : 비트코인 가격 데이터 - [Close, Open, High, Low, Volume]
- GLD_price : 국제 금 시세 + 달러~원 환율 데이터
- nikkei : 닛케이 지수 데이터
- shanghai 상하이종합지수 데이터
- kospi : 코스피지수 데이터
- nasdaq : 나스닥지수 데이터

조사 기간은 <2017-01-01 ~ 2021-05-07>로 동일함

bt.head(3)

Date	Close	Open	High	Low	Volue_K	Volume
2017-01-01	995.4	963.4	1001.6	956.1	41.15	41150.0
2017-01-02	1017.0	995.4	1031.7	990.2	64.95	64950.0
2017-01-03	1033.3	1017.0	1035.5	1006.5	54.79	54790.0

nk.head(3)

Date	Nikkei
2017-01-04	19594.16
2017-01-05	19520.69
2017-01-06	19454.33

sh.head(3)

Date	Shanghai
2017-01-03	3135.92
2017-01-04	3158.79
2017-01-05	3165.41

kospi.head(3)

Date	KOSPI
2017-01-02	2026.16
2017-01-03	2043.97
2017-01-04	2045.64

nasdaq.head(3)

Date	Nasdaq
2017-01-03	5429.08
2017-01-04	5477.01
2017-01-05	5487.94

gld.head(3)

Date	GLD_Price	USD_KRW
2017-01-02	1151.67	1209.0
2017-01-03	1151.35	1206.0
2017-01-04	1164.23	1203.0

Date	Close	Open	High	Low	Volue_K	Volume	GLD_Price	USD_KRW	Nikkei	Shanghai	KOSPI	Nasdaq
2017-01-01	995.4	963.4	1001.6	956.1	41.15	41150.0	NaN	NaN	NaN	NaN	NaN	NaN
2017-01-02	1017.0	995.4	1031.7	990.2	64.95	64950.0	1151.67	1209.0	NaN	NaN	2026.16	NaN
2017-01-03	1033.3	1017.0	1035.5	1006.5	54.79	54790.0	1151.35	1206.0	NaN	3135.92	2043.97	5429.08
2017-01-04	1135.4	1033.3	1148.5	1022.3	156.27	156270.0	1164.23	1203.0	19594.16	3158.79	2045.64	5477.01
2017-01-05	989.3	1135.4	1150.6	874.5	240.01	240010.0	1172.59	1188.0	19520.69	3165.41	2041.95	5487.94

Pd.concat(axis=1, join='outer')를 사용해
하나의 데이터프레임으로 병합

데이터 소개


```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1588 entries, 2017-01-01 to 2021-05-07
Freq: D
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Close       1588 non-null   float64
 1   Open        1588 non-null   float64
 2   High        1588 non-null   float64
 3   Low         1588 non-null   float64
 4   Volue_K     1588 non-null   float64
 5   Volume      1588 non-null   float64
 6   GLD_Price   1071 non-null   float64
 7   USD_KRW     1071 non-null   float64
 8   Nikkei      1088 non-null   float64
 9   Shanghai    1055 non-null   float64
 10  KOSPI       1067 non-null   float64
 11  Nasdaq      1094 non-null   float64
dtypes: float64(12)
memory usage: 161.3 KB
```

- 전체 1588개 행(datetime index)으로 이루어짐
- 12개 컬럼으로 구성됨
- 일별 데이터들의 집합
- 각 항목마다 관측주기가 동일하지 않기 때문에 결측치가 다수 존재
- 변수들은 모두 연속형(float)으로 저장됨 (범주형 변수 x)

데이터 전처리

1. 결측치 처리(보간법 사용)

```
def fmissing(df):
    for col in df.columns:
        df[col] = df[col].interpolate()
    print(f'No. of Missing Values after interpolation:₩n{df.isnull().sum()}')
```

```
fmissing(df)
```

```
No. of Missing Values after interpolation:
Close        0
Open         0
High         0
Low          0
Volue_K      0
Volume       0
GLD_Price    1
USD_KRW      1
Nikkei       3
Shanghai     2
KOSPI        1
Nasdaq       2
dtype: int64
```

2. 나머지 결측치 처리(dropna 사용)

```
df.head()
```

Date	Close	Open	High	Low	Volue_K	Volume	GLD_Price	USD_KRW	Nikkei	Shanghai	KOSPI	Nasdaq
2017-01-01	995.4	963.4	1001.6	956.1	41.15	41150.0	NaN	NaN	NaN	NaN	NaN	NaN
2017-01-02	1017.0	995.4	1031.7	990.2	64.95	64950.0	1151.67	1209.0	NaN	NaN	2026.16	NaN
2017-01-03	1033.3	1017.0	1035.5	1006.5	54.79	54790.0	1151.35	1206.0	NaN	3135.92	2043.97	5429.08
2017-01-04	1135.4	1033.3	1148.5	1022.3	156.27	156270.0	1164.23	1203.0	19594.16	3158.79	2045.64	5477.01
2017-01-05	989.3	1135.4	1150.6	874.5	240.01	240010.0	1172.59	1188.0	19520.69	3165.41	2041.95	5487.94

보간법을 사용했으므로 맨 앞 3번째까지는 여전히 결측치로 존재

```
df.dropna(inplace=True)
```

```
df.isnull().sum()
```

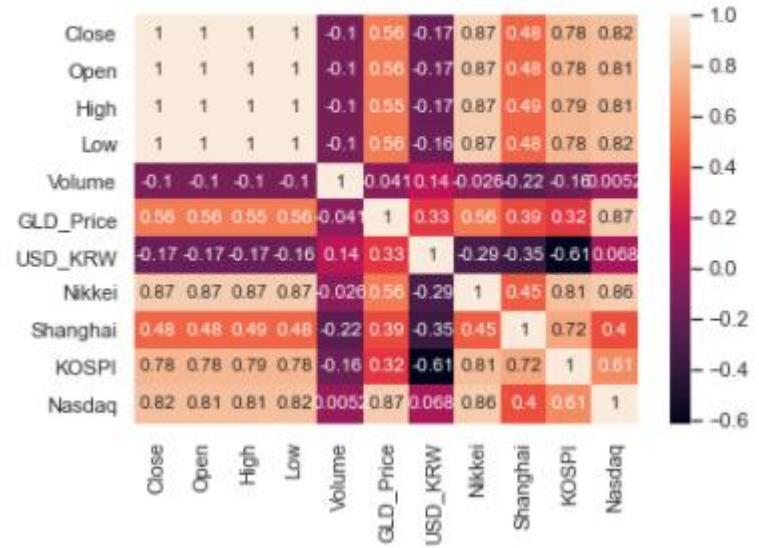
```
Close        0
Volume       0
GLD_Price    0
USD_KRW      0
Nikkei       0
Shanghai     0
KOSPI        0
Nasdaq       0
dtype: int64
```

데이터 전처리

3. 불필요 변수 제거

```
sns.heatmap(df.corr(), annot=True)
```

```
<AxesSubplot:>
```



- 타겟 : Close(종가)
- Close = Open = High = Low는 각각 상관계수가 1
- 각각에 영향을 미치는 것이 아니라 함께 움직인다고 판단, 삭제
- 그 외에 입력변수들 끼리 어느정도 상관성이 존재하는 것을 확인(다중공선성 존재)

```
df.drop(['Open','High','Low'], axis=1, inplace=True)
```

```
df.columns
```

```
Index(['Close', 'Volume', 'GLD_Price', 'USD_KRW', 'Nikkei', 'Shanghai',
       'KOSPI', 'Nasdaq'],
      dtype='object')
```

데이터 전처리

4. 분석에 사용될 "df" 데이터프레임 생성

```
df.head()
```

Date	Close	Volume	GLD_Price	USD_KRW	Nikkei	Shanghai	KOSPI	Nasdaq
2017-01-04	1135.4	156270.0	1164.23	1203.000000	19594.1600	3158.79	2045.640000	5477.010000
2017-01-05	989.3	240010.0	1172.59	1188.000000	19520.6900	3165.41	2041.950000	5487.940000
2017-01-06	886.2	194290.0	1176.25	1193.500000	19454.3300	3154.32	2049.120000	5521.060000
2017-01-07	888.9	130660.0	1176.45	1197.666667	19416.1075	3159.96	2049.006667	5524.646667
2017-01-08	900.9	76910.0	1176.65	1201.833333	19377.8850	3165.60	2048.893333	5528.233333

5. "df" 데이터프레임을 모델의 학습/검정용 데이터셋으로 분할

분할 비율은 일반적인 비율인 7:3으로 설정

```
x = df.iloc[:, 1:]
y = df['Close']
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state = 33)
```

데이터 전처리

입력변수

1. 비트코인 거래량
2. 국제 금 시세
3. 원-달러 환율
4. 닛케이 종합지수
5. 상해종합지수
6. 코스피종합지수
7. 나스닥 종합지수

선형회귀

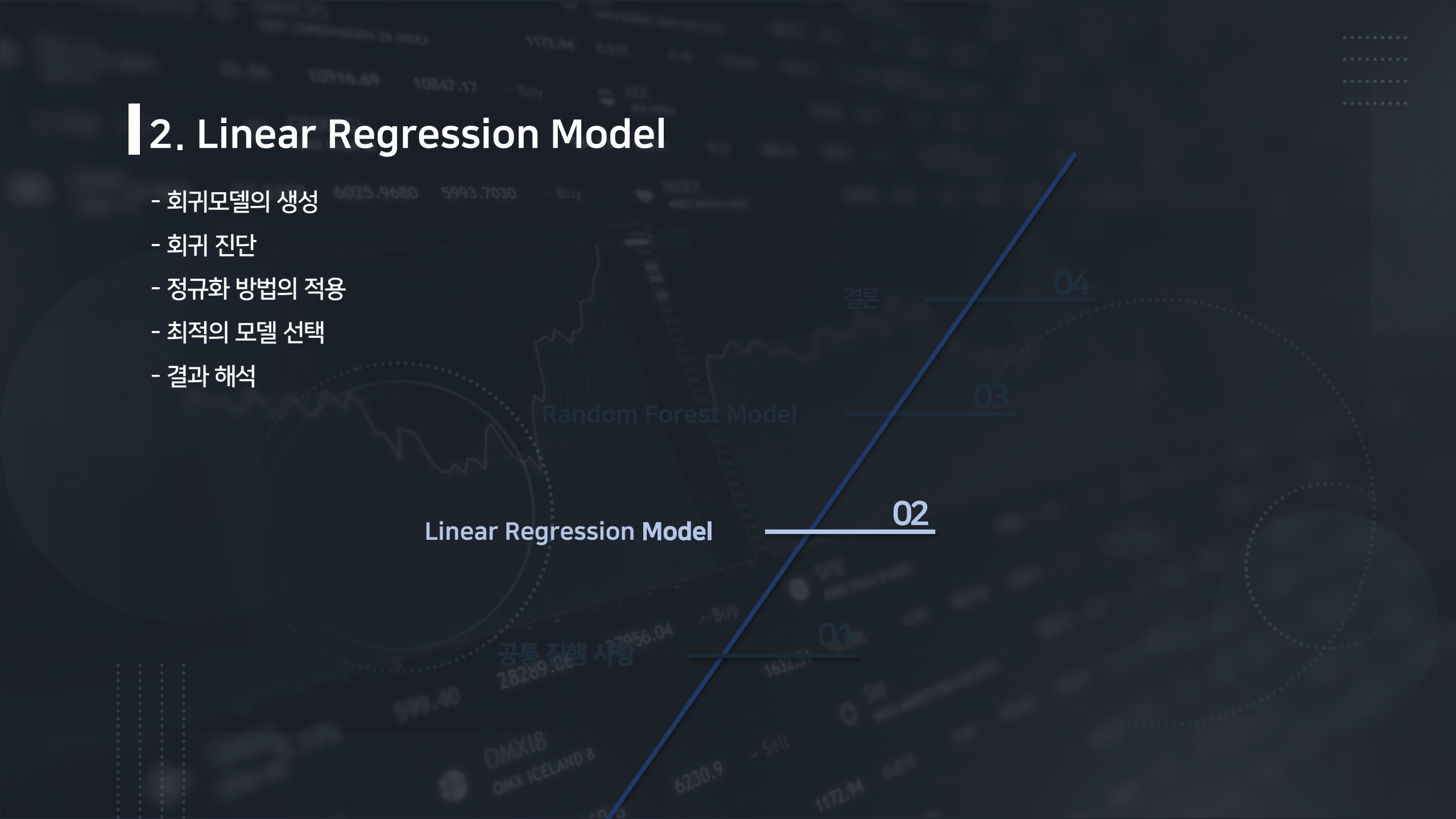
랜덤포레스트

종속변수

Close (종가)

2. Linear Regression Model

- 회귀모델의 생성
- 회귀 진단
- 정규화 방법의 적용
- 최적의 모델 선택
- 결과 해석



회귀 모델의 생성

1. 선형회귀로 적합하기 위해 상수항으로 이루어진 컬럼을 추가

```
x_train_add = sm.add_constant(x_train)
```

```
x_test_add = sm.add_constant(x_test)
```

2. 최소제곱법(OLS)을 적용한 선형 회귀 모델의 적합

OLS Regression Results

Dep. Variable:	Close	R-squared:	0.847
Model:	OLS	Adj. R-squared:	0.846
Method:	Least Squares	F-statistic:	872.2
Date:	Sat, 05 Jun 2021	Prob (F-statistic):	0.00
Time:	13:06:58	Log-Likelihood:	-10965.
No. Observations:	1109	AIC:	2.195e+04
Df Residuals:	1101	BIC:	2.199e+04
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.676e+05	9559.791	-17.530	0.000	-1.86e+05	-1.49e+05
Volume	-0.0006	9.95e-05	-6.010	0.000	-0.001	-0.000
GLD_Price	-16.6264	2.375	-7.001	0.000	-21.286	-11.967
USD_KRW	94.7603	6.384	14.843	0.000	82.233	107.287
Nikkei	0.6613	0.216	3.060	0.002	0.237	1.085
Shanghai	-2.2458	1.062	-2.114	0.035	-4.330	-0.162
KOSPI	25.5925	1.729	14.801	0.000	22.200	28.985
Nasdaq	3.3290	0.419	7.939	0.000	2.506	4.152

Omnibus:	66.534	Durbin-Watson:	2.024
Prob(Omnibus):	0.000	Jarque-Bera (JB):	77.311
Skew:	0.641	Prob(JB):	1.63e-17
Kurtosis:	3.175	Cond. No.	1.11e+08

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.11e+08. This might indicate that there are strong multicollinearity or other numerical problems.

학습데이터에 대해 84.6%의 설명력을 가짐

다른 변수가 고정되었을 때, 해당 변수의 1단위 증가가 종속변수에 미치는 변화량
- 현재 단위: 달러
- 따라서 계수: 1단위 증가당 변화하는 달러의 양을 의미

정규화 방법의 적용


```
alphas = [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1]
```

```
for a in alphas:
    model = ElasticNet(alpha=a).fit(x_train_add,y_train)
    score = model.score(x_train_add, y_train)
    pred_y = model.predict(x_test_add)
    mse = mean_squared_error(y_test, pred_y)
    print("Alpha:{0:.4f}, R2:{1:.2f}, MSE:{2:.2f}, RMSE:{3:.2f}".format(a, score, mse, np.sqrt(mse)))
```

```
Alpha:0.0001, R2:0.85, MSE:21550635.55, RMSE:4642.27
Alpha:0.0005, R2:0.85, MSE:21550635.28, RMSE:4642.27
Alpha:0.0010, R2:0.85, MSE:21550634.95, RMSE:4642.27
Alpha:0.0050, R2:0.85, MSE:21550632.27, RMSE:4642.27
Alpha:0.0100, R2:0.85, MSE:21550628.93, RMSE:4642.27
Alpha:0.0500, R2:0.85, MSE:21550602.20, RMSE:4642.26
Alpha:0.1000, R2:0.85, MSE:21550568.81, RMSE:4642.26
Alpha:0.5000, R2:0.85, MSE:21550302.80, RMSE:4642.23
Alpha:1.0000, R2:0.85, MSE:21549973.06, RMSE:4642.19
```

```
elastic_cv=ElasticNetCV(alphas=alphas, cv=10)
model = elastic_cv.fit(x_train_add, y_train)
print(model.alpha_)
```

```
1.0
```

```
elastic=ElasticNet(alpha=1.0).fit(x_train_add, y_train)
ypred_elastic = elastic.predict(x_test_add)
score_elastic = elastic.score(x_test_add, y_test)
mse_elastic = mean_squared_error(y_test, ypred_elastic)
print("Final Result: ElasticNet R2:{0:.3f}, MSE:{1:.2f}, RMSE:{2:.2f}"
.format(score_elastic, mse_elastic, np.sqrt(mse_elastic)))
```

```
Final Result: ElasticNet R2:0.857, MSE:21549973.06, RMSE:4642.19
```

- Ridge: 각 계수의 영향력을 줄이는 방법(0으로 수렴)
- Lasso: 각 계수의 영향력을 줄이면서 변수를 제거하는 효과
- ElasticNet: Ridge + Lasso

<- 위 과정을 Ridge/Lasso/ElasticNet에 대해 수행

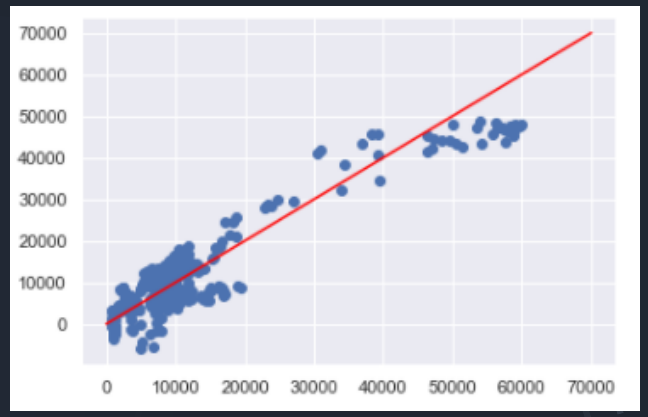
```
print("<<<<Model Comparison>>>>")
print("₩n---- Base LR ----")
print("[Regression] R2:0.856, MSE:",round(mse2, 3), "RMSE:",round(np.sqrt(mse2), 2))
print("₩n---- Normalize Methods ----")
print("[Ridge] R2:{0:.3f}, MSE:{1:.2f}, RMSE:{2:.2f}".format(score_ridge, mse_ridge, np.sqrt(mse_ridge)))
print("[Lasso] R2:{0:.3f}, MSE:{1:.2f}, RMSE:{2:.2f}".format(score_lasso, mse_lasso, np.sqrt(mse_lasso)))
print("[ElasticNet] R2:{0:.3f}, MSE:{1:.2f}, RMSE:{2:.2f}".format(score_elastic, mse_elastic, np.sqrt(mse_elastic)))
```

```
<<<<Model Comparison>>>>

---- Base LR ----
[Regression] R2:0.856, MSE: 21550635.616 RMSE: 4642.27

---- Normalize Methods ----
[Ridge] R2:0.857, MSE:21550634.43, RMSE:4642.27
[Lasso] R2:0.857, MSE:21550635.61, RMSE:4642.27
[ElasticNet] R2:0.857, MSE:21549973.06, RMSE:4642.19
```

-> 최종 모델: **ElasticNet**

최적의 모델 선택

1. 실제 Close 데이터 vs 예측값의 비교



	Actual	Predicted
Actual	1.000000	0.926554
Predicted	0.926554	1.000000

- 실제 – 예측 데이터의 상관계수가 0.9265에 달함
- 예측이 실제와 비슷하게 이루어진다는 결론 도출

2. 최종 모델의 설명력 및 계수 확인

```
[ElasticNet] R2:0.857, MSE:21549973.06, RMSE:4642.19
```

```
elastic.coef_
```

```
array([ 0.00000000e+00, -5.98047335e-04, -1.66276938e+01,  9.46618940e+01,
        6.60884352e-01, -2.24187182e+00,  2.55759251e+01,  3.33091515e+00])
```

- 최종 ElasticNet Model의 R-squared 값은 0.857
- 베이스 선형회귀모델보다 성능이 개선
- 하지만 여전히 높은 MSE

선형회귀모델 분석결과

비트코인 가격 방정식

```
elastic.coef_
```

```
array([ 0.00000000e+00, -5.98047335e-04, -1.66276938e+01,  9.46618940e+01,
        6.60884352e-01, -2.24187182e+00,  2.55759251e+01,  3.33091515e+00])
```

비트코인 가격 =

0 - 0.00059*거래량 - 16.62769*금시세 + 94.66189*원달러환율 + 0.66088*닛케이지수 - 2.24187*상해종합지수 + 25.57592*코스피지수 + 3.33092*나스닥지수

주요 변수 분석

1) 원달러환율이 상승할수록,
2) 한-일-미 주식의 종합지수가 상승할 수록,

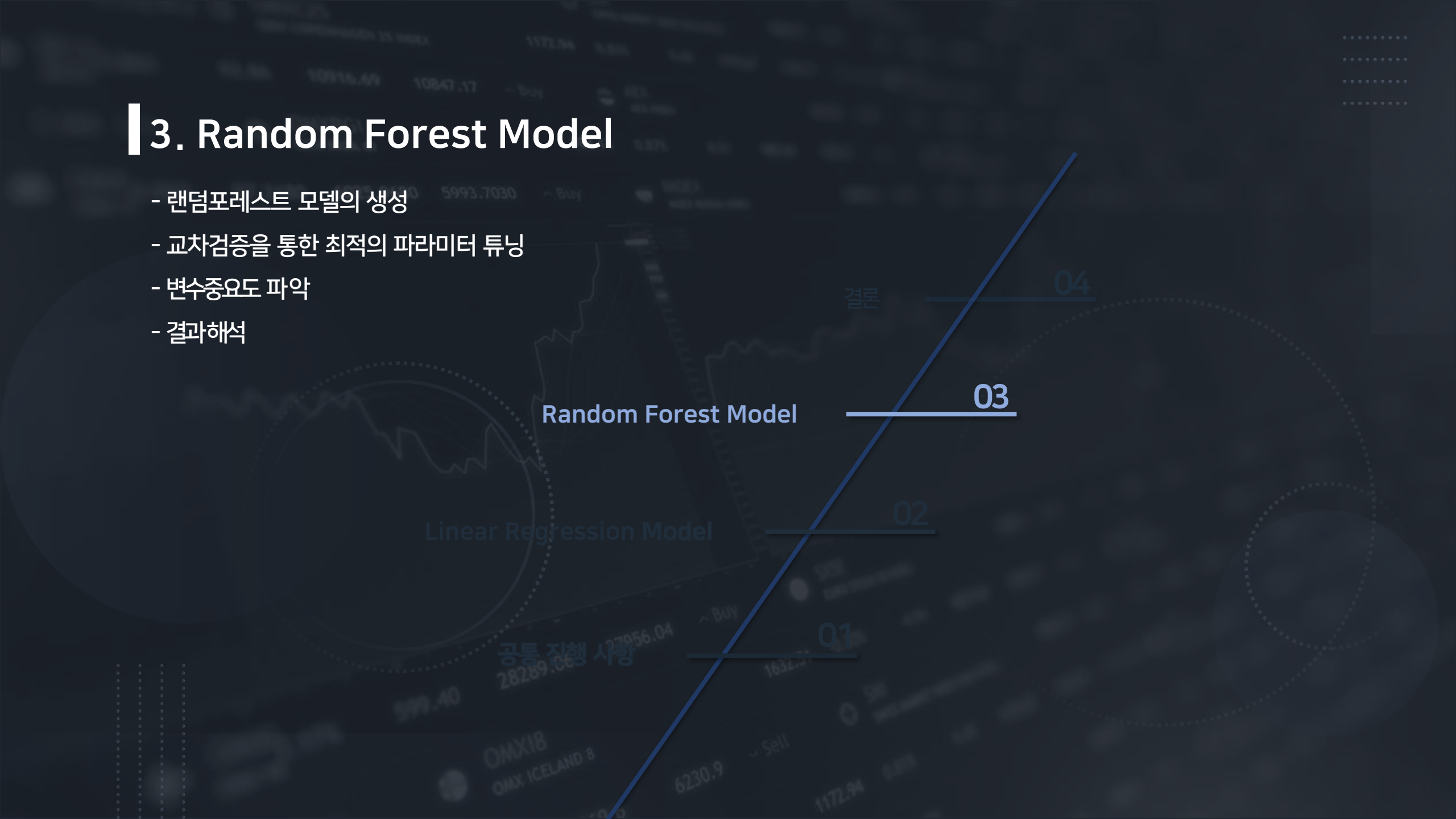
비트코인 가격이 상승하였고,

1) 금 시세가 상승할 수록
2) 중국 주식 지수가 상승할 수록

비트코인 가격이 하락한다는 결론을 도출

3. Random Forest Model

- 랜덤포레스트 모델의 생성
- 교차검증을 통한 최적의 파라미터 튜닝
- 변수중요도 파악
- 결과해석



랜덤포레스트 모델의 생성


```
nTreeList = range(10, 1000, 10)
mseOos = []

for iTrees in nTreeList:
    depth = None
    maxFeat = 2
    rf = ensemble.RandomForestRegressor(n_estimators=iTrees,
                    max_depth=depth, max_features=maxFeat,
                    oob_score=False, random_state=531)
    rf.fit(x_train, y_train)

    prediction = rf.predict(x_test)
    mseOos.append(mean_squared_error(y_test, prediction))
    print(iTrees,' : ',mean_squared_error(y_test, prediction))
print('<< LeastMse RF >>₩n')
print(min(mseOos))
```

```
<< LeastMse RF >>

1248347.6702583968
```

```
nTreeList = range(10, 1000, 10)
mseOos1 = []

for iTrees in nTreeList:
    depth = None
    maxFeat = 3
    rf = ensemble.RandomForestRegressor(n_estimators=iTrees,
                    max_depth=depth, max_features=maxFeat,
                    oob_score=False, random_state=531)
    rf.fit(x_train, y_train)

    prediction = rf.predict(x_test)
    mseOos1.append(mean_squared_error(y_test, prediction))
    print(iTrees,' : ',mean_squared_error(y_test, prediction))
print('<< LeastMse RF >>₩n')
print(min(mseOos1))
```

```
<< LeastMse RF >>

1066831.738765832
```

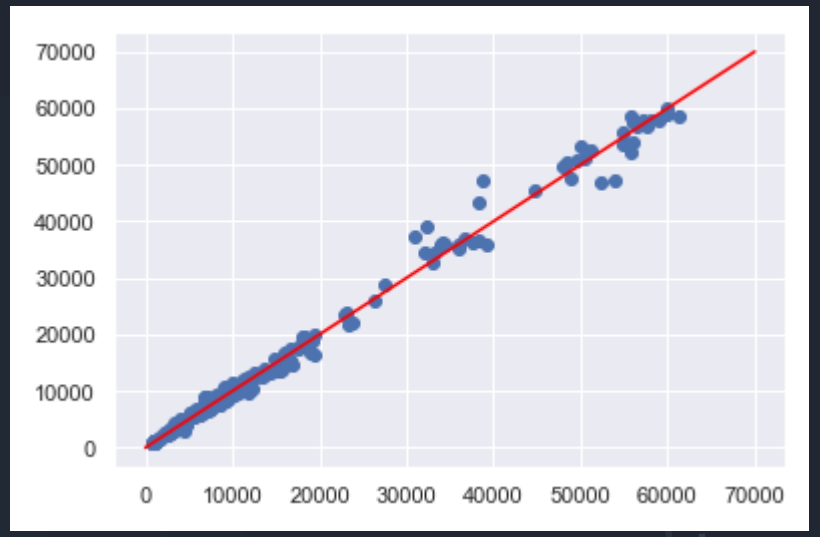
iTrees = 150

- Max_features: 결합할 의사결정나무의 개수, 일반적으로 회귀의 경우 "변수 / 3"으로 지정

  -> 2/3으로 나누어 진행

- Max_features = 3 / n_estimators = 150일 때 MSE값이 가장 낮았음

결과 해석




```
df_rf.corr()
```

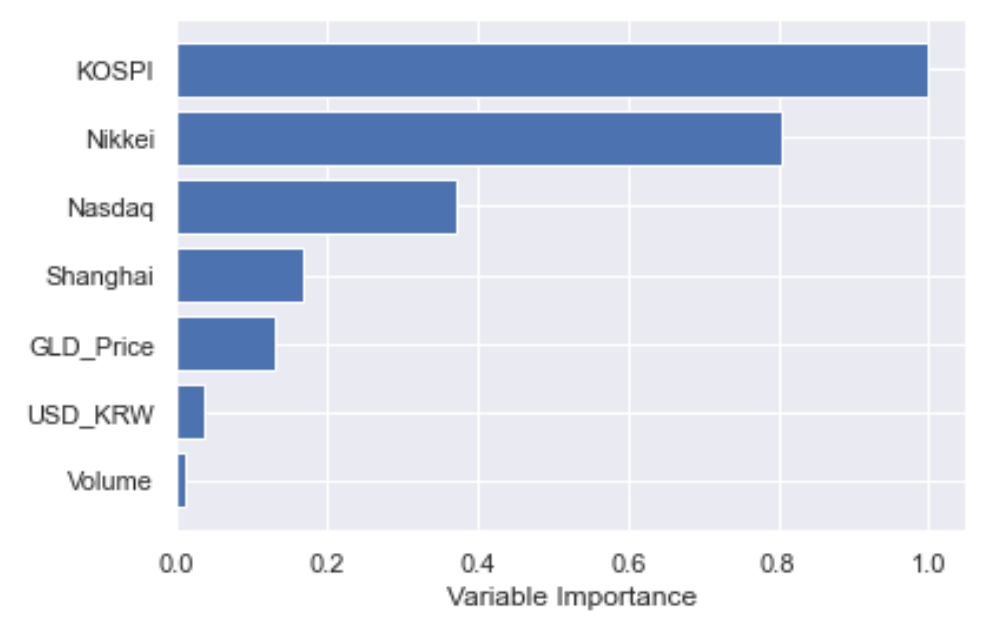
	Actual	Predicted
Actual	1.000000	0.996473
Predicted	0.996473	1.000000

```
print(mean_squared_error(y_test, pred_rf))
```

```
1066831.738765832
```

- 실제 값과 예측 값이 거의 동일하게 분포(상관계수: 0.99647)
- MSE값도 선형회귀_1973만 -> 랜덤포레스트_106만으로 대폭 감소
- $\sqrt{MSE} = 1032$ (달러)로, 수만 달러 단위에 해당하는 비트코인 가격에 대해 낮은 오차를 보임.

변수 중요도



- 선형회귀와 동일하게 코스피지수가 비트코인 가격에 큰 영향을 주는 것으로 파악
- 전반적으로 주식 지수들의 영향력이 높았는데, 이는 다중공선성과 관련이 있을 것으로 예상
- 주식 지수를 제외하고 영향력이 높았던 변수는 "금시세"
- 원-달러 환율은 선형회귀만큼의 비중을 차지하지는 않음
- 거래량은 여전히 미미한 영향

4. 결론

- 시사점
- 최근 데이터 적용
- 한계점 및 보완점

시사점

선형회귀 모델:

- 원-달러 환율과 강한 비례관계
- 코스피지수와 강한 비례관계
- 금 시세와 강한 반비례관계

랜덤포레스트 모델:

- 코스피지수가 가장 높은 중요도 차지
- 금 시세 역시 중요한 변수로 나타남

시사점

코스피

코스피, 外人 매도에 사흘째 하락…3140선 후퇴

- 최근 비트코인을 비롯한 암호화폐들은 지속적인 하락세를 보이고 있음.
- 코스피 지수 역시 5월 24일 기준 사흘째 하락하며 약세를 보이고 있음.
- 따라서 이전 분석 결과와 같이 “코스피와 비트코인이 비례함”을 알 수 있음.

금시세

- 반면 변동성이 큰 암호화폐를 대신해 안정적인 자산으로의 관심이 증가하는 추세이며, 금은 대표적인 안전자산임.
- 2021년 1월 부터 기록된 위 두 그래프는 “비트코인과 금 가격이 서로 반비례함 " 을 보여줌.

2021-05-24일자 데이터 적용

Prediction: current_data

```
x_curr = np.array([310200, 1877.98, 1127.00, 28364.61, 3497.28, 3144.30, 13588.34])
y_curr = np.array(37824.9)
```

```
pred_curr = RF.predict(x_curr.reshape(1,-1))
print("[Actual]: ",y_curr, "₩n[Prediction]: ",pred_curr, "₩n[Residual]: ", y_curr - pred_curr,
      "₩n[Residual_per]: ", (y_curr - pred_curr)/y_curr*100)
```

```
[Actual]:  37824.9
[Prediction]:  [37451.38]
[Residual]:  [373.52]
[Residual_per]:  [0.98749765]
```

- 랜덤포레스트 모델의 예측값은 약 "37451.38 달러"로, 오차는 373.52달러에 불과함
- 이는 실제 값과 0.98%만 차이나는 것으로, 모델이 과적합되지 않고 새로운 데이터에 예측을 잘 수행함을 의미

한계점 및 보완점

- Sparse한 데이터 – 성능이 나쁘지는 않았으나, 과적합의 위험성에서 자유롭다고 할 수는 없을 듯 함(x – 1588개)
- 시계열 분석방법의 부재 – 본 연구에서 활용한 선형회귀, 랜덤포레스트 기법은 지도학습이므로, Y를 도출하기 위해 X가 주어져야 함, 하지만 이러한 방법으로는 "**내일 가격이 오를지 내릴지**"에 대한 해답을 찾기는 힘듦.
- 데이터의 비선형성 – 비트코인 가격 데이터는 불규칙하며 비선형적인 분포를 따른다는 점에서 선형 회귀에 적합하지는 않을 것임. 또한 입력 변수간의 다중공선성 역시 선형회귀모델의 신뢰성을 다소 저하한 바 있음.
- 따라서 **딥러닝 기반의 시계열 분석 모델**의 적용이 필요할 것임.
- **추가로 주식 지수, 거래량, 환율, 금 시세 등의 데이터뿐만 아니라 매일 발생하는 이슈들에 대한 감성 분석까지 도달한다면 보다 유의미하고 차원 높은 예측모델을 생성할 수 있을 것이라 예상**


감사합니다
20163 10930 신성구