Gold Price Forecasting

시계열 분석 기말 팀프로젝트

허준봉
22512091

김경아
24510097

김정현
23530006

한상훈
24510115

SEOULTECH

Index

1. Data Information

원자재

GC

<class 'pandas.core.frame.DataFrame'>

Datetime Index	3618 entries	2010-01-04 to 2024-05-15

Data columns (total 6 columns)

Column	Non-Null Count	Dtype
Open	3614 non-null	float64
High	3614 non-null	float64
Low	3614 non-null	float64
Close	3614 non-null	float64
Adj Close	3614 non-null	float64
Volume	3614 non-null	float64
dtypes	float64(6)	

CL

<class 'pandas.core.frame.DataFrame'>

Datetime Index	3618 entries	2010-01-04 to 2024-05-15

Data columns (total 6 columns)

Column	Non-Null Count	Dtype
Open	3615 non-null	float64
High	3615 non-null	float64
Low	3615 non-null	float64
Close	3615 non-null	float64
Adj Close	3615 non-null	float64
Volume	3615 non-null	float64
dtypes	float64(6)	

BZ

<class 'pandas.core.frame.DataFrame'>

Datetime Index	3618 entries	2010-01-04 to 2024-05-15

Data columns (total 6 columns)

Column	Non-Null Count	Dtype
Open	3578 non-null	float64
High	3578 non-null	float64
Low	3578 non-null	float64
Close	3578 non-null	float64
Adj Close	3578 non-null	float64
Volume	3578 non-null	float64
dtypes	float64(6)	

NG

<class 'pandas.core.frame.DataFrame'>

Datetime Index	3618 entries	2010-01-04 to 2024-05-15

Data columns (total 6 columns)

Column	Non-Null Count	Dtype
Open	3615 non-null	float64
High	3615 non-null	float64
Low	3615 non-null	float64
Close	3615 non-null	float64
Adj Close	3615 non-null	float64
Volume	3615 non-null	float64
dtypes	float64(6)	

SI

<class 'pandas.core.frame.DataFrame'>

Datetime Index	3618 entries	2010-01-04 to 2024-05-15

Data columns (total 6 columns)

Column	Non-Null Count	Dtype
Open	3613 non-null	float64
High	3613 non-null	float64
Low	3613 non-null	float64
Close	3613 non-null	float64
Adj Close	3613 non-null	float64
Volume	3613 non-null	float64
dtypes	float64(6)	

HG

<class 'pandas.core.frame.DataFrame'>

Datetime Index	3618 entries	2010-01-04 to 2024-05-15

Data columns (total 6 columns)

Column	Non-Null Count	Dtype
Open	3614 non-null	float64
High	3614 non-null	float64
Low	3614 non-null	float64
Close	3614 non-null	float64
Adj Close	3614 non-null	float64
Volume	3614 non-null	float64
dtypes	float64(6)	

1. Data Information

환율

USDKRW

<class 'pandas.core.frame.DataFrame'>

Datetime Index	3749 entries	2010-01-01 to 2024-05-15
Data columns (total 6 columns)		
Column	Non-Null Count	Dtype
Open	3743 non-null	float64
High	3743 non-null	float64
Low	3743 non-null	float64
Close	3743 non-null	float64
Adj Close	3743 non-null	float64
Volume	3743 non-null	float64
dtypes	float64(6)	

USDEUR

<class 'pandas.core.frame.DataFrame'>

Datetime Index	3749 entries	2010-01-01 to 2024-05-15
Data columns (total 6 columns)		
Column	Non-Null Count	Dtype
Open	3744 non-null	float64
High	3744 non-null	float64
Low	3744 non-null	float64
Close	3744 non-null	float64
Adj Close	3744 non-null	float64
Volume	3744 non-null	float64
dtypes	float64(6)	

USDCNY

<class 'pandas.core.frame.DataFrame'>

Datetime Index	3749 entries	2010-01-01 to 2024-05-15
Data columns (total 6 columns)		
Column	Non-Null Count	Dtype
Open	3742 non-null	float64
High	3742 non-null	float64
Low	3742 non-null	float64
Close	3742 non-null	float64
Adj Close	3742 non-null	float64
Volume	3742 non-null	float64
dtypes	float64(6)	

암호화폐

BTCUSD

<class 'pandas.core.frame.DataFrame'>

Datetime Index	3529 entries	2014-09-17 to 2024-05-15
Data columns (total 6 columns)		
Column	Non-Null Count	Dtype
Open	3529 non-null	float64
High	3529 non-null	float64
Low	3529 non-null	float64
Close	3529 non-null	float64
Adj Close	3529 non-null	float64
Volume	3529 non-null	int64
dtypes	float64(5), int64(1)	

ETHUSD

<class 'pandas.core.frame.DataFrame'>

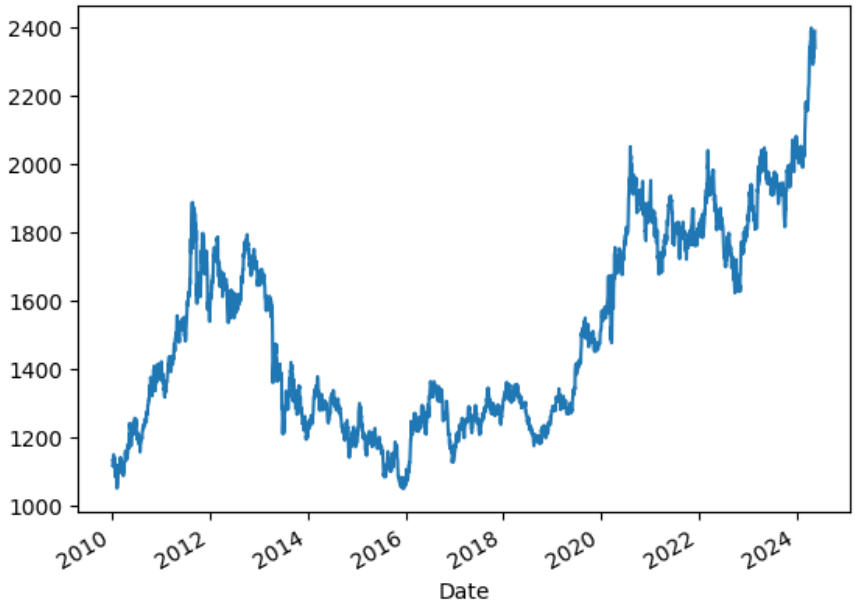
Datetime Index	2380 entries	2017-11-09 to 2024-05-15
Data columns (total 6 columns)		
Column	Non-Null Count	Dtype
Open	2380 non-null	float64
High	2380 non-null	float64
Low	2380 non-null	float64
Close	2380 non-null	float64
Adj Close	2380 non-null	float64
Volume	2380 non-null	int64
dtypes	float64(5), int64(1)	

2. ARIMA

데이터 전처리

- **결측치 처리:** Quadratic 보간법을 사용하여 결측치 보완.
- **컬럼명 변경:** 데이터 구분을 위해 각 데이터셋의 컬럼명에 접두어 추가.
- **공통 인덱스 설정:** 모든 데이터셋의 인덱스를 금 선물(GC)의 인덱스로 통일.

GC['GC_Close'].plot()

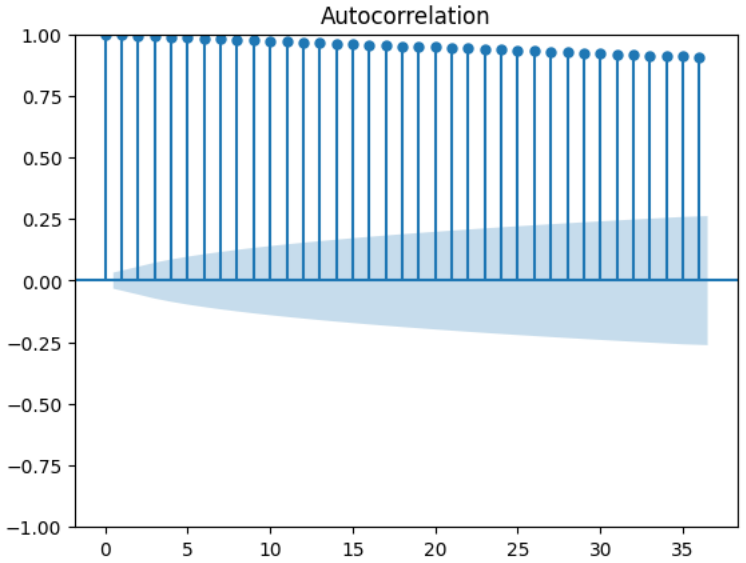


2. ARIMA

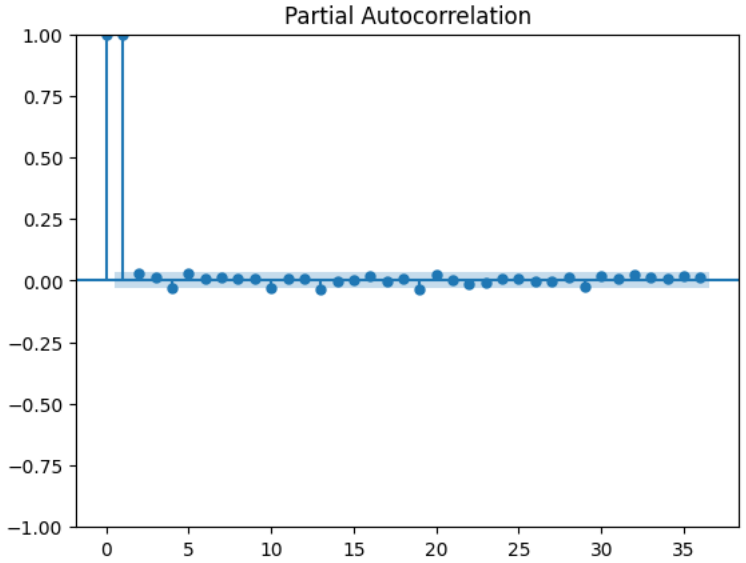
데이터 전처리

- **결측치 처리:** Quadratic 보간법을 사용하여 결측치 보완.
- **컬럼명 변경:** 데이터 구분을 위해 각 데이터셋의 컬럼명에 접두어 추가.
- **공통 인덱스 설정:** 모든 데이터셋의 인덱스를 금 선물(GC)의 인덱스로 통일.

```
acf_original = plot_acf(GC['GC_Close'])
```




```
pacf_original = plot_pacf(GC['GC_Close'])
```



2. ARIMA

시계열 안정성 검정

•**ADF 검정:** 시계열 데이터가 안정적인지 확인.

- **결과:** ADF 통계량, p-value, 임계값.

•**KPSS 검정:** 시계열 데이터의 안정성 검증.

- **결과:** KPSS 통계량, p-value, 임계값.

ADF Statistic: -0.3354138832333168
p-value: 0.9203367547319173
Critical Value (1%): -3.432
Critical Value (5%): -2.862
Critical Value (10%): -2.567

KPSS Statistic: 4.141669968097387
p-value: 0.01
Critical Value (10%): 0.347
Critical Value (5%): 0.463
Critical Value (2.5%): 0.574
Critical Value (1%): 0.739

2. ARIMA

모델 학습 및 최적화

•**차분(d) 최적화:** KPSS 및 ADF 검정을 통해 최적의 차분 차수(d) 결정.

- **결과:** 최적의 d 값.

•**Box-Cox 변환:** 데이터의 정규성을 맞추기 위해 Box-Cox 변환 적용.

- **결과:** Lambda 값.

•**모델 학습:** ARIMA 모델을 사용하여 변환된 데이터 학습.

```
kpss_diffs = pm.arima.ndiffs(train, alpha=0.05, test='kpss', max_d=5)
adf_diffs = pm.arima.ndiffs(train, alpha=0.05, test='adf', max_d=5)
n_diffs = max(kpss_diffs, adf_diffs)

print(f"Optimized 'd' = {n_diffs}")
```

Optimized 'd' = 1

2. ARIMA

모델 학습 및 최적화

```
# 직접 지정해주어 모델링 실시

model = auto_arima(y=train, # 데이터
d=n_diffs, # 차분 (d), 기본값 = None
start_p= 0, # 시작 p값, 기본값 = 2
max_p = 5, # p 최대값, 기본값 = 5
start_q= 0, # 시작 q값, 기본값 = 2
max_q = 5, # q 최대값, 기본값 = 5
m=1, # season의 주기, 기본값 = 1
seasonal=False, # sARIMA를 실시, 기본값 = True
stepwise=True, # stepwise algorithm, 기본값 = True
trace=True) # 각 step을 출력할지, 기본값 = False

model2 = auto_arima(train, d=1, seasonal=False, trace=True)
```

2. ARIMA

모델 학습 및 최적화

model

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=30145.105, Time=0.10 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=30144.206, Time=0.14 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=30144.181, Time=1.17 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=30144.938, Time=0.04 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=30146.167, Time=1.68 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=30146.139, Time=1.62 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=30144.886, Time=4.23 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=30144.121, Time=0.65 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=30146.110, Time=1.02 sec
 ARIMA(0,1,2)(0,0,0)[0]             : AIC=30146.092, Time=0.48 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=30144.142, Time=0.11 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=30144.861, Time=1.42 sec

Best model:  ARIMA(0,1,1)(0,0,0)[0]
Total fit time: 12.672 seconds
```

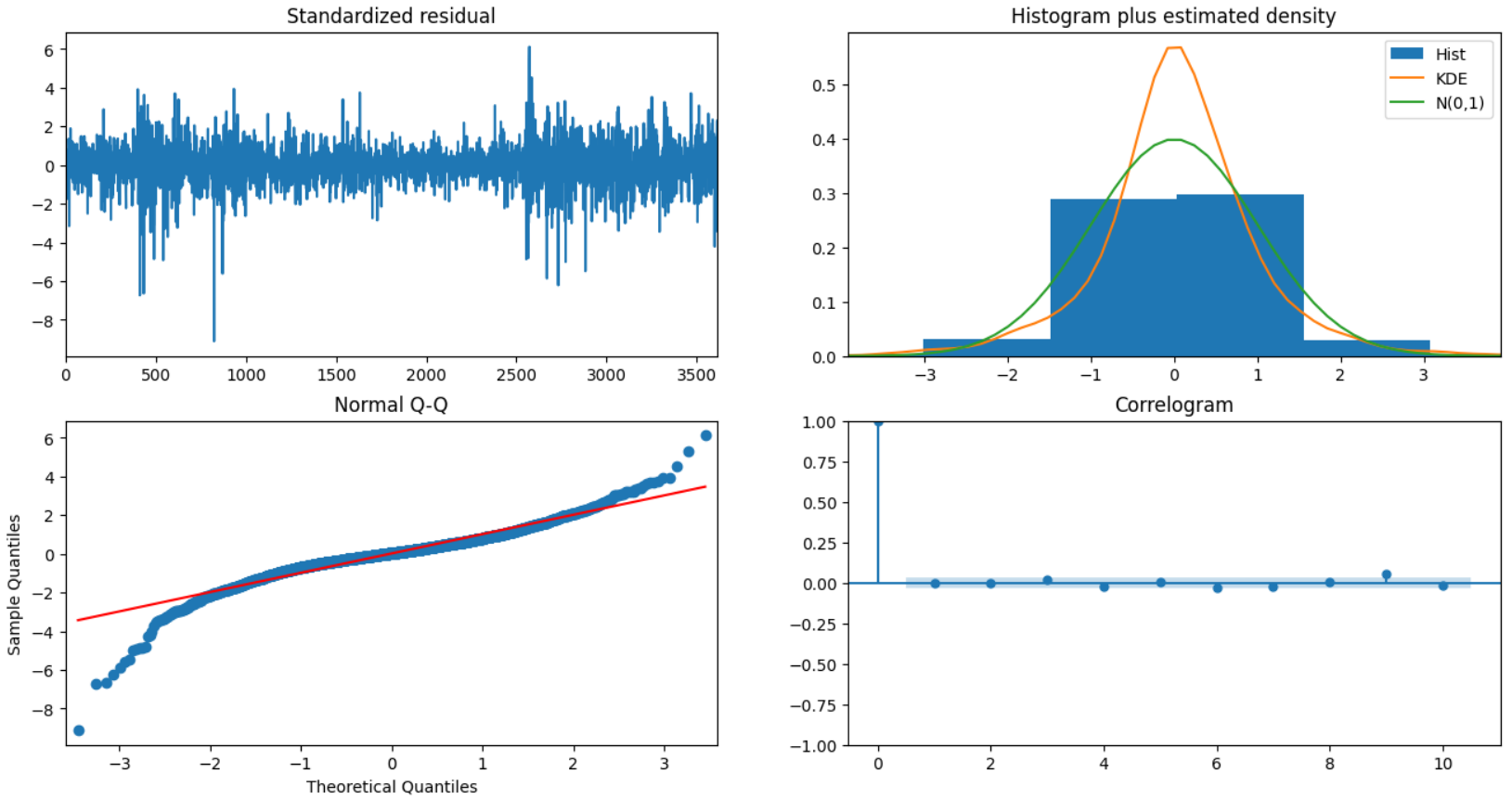
model2

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=30144.904, Time=7.70 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=30145.105, Time=0.09 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=30144.206, Time=0.20 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=30144.181, Time=0.91 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=30144.938, Time=0.10 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=30146.167, Time=1.48 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=30146.139, Time=1.42 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=30144.886, Time=4.10 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=30144.121, Time=0.45 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=30146.110, Time=0.87 sec
 ARIMA(0,1,2)(0,0,0)[0]             : AIC=30146.092, Time=0.43 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=30144.142, Time=0.09 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=30144.861, Time=1.38 sec

Best model:  ARIMA(0,1,1)(0,0,0)[0]
Total fit time: 19.231 seconds
```

2. ARIMA

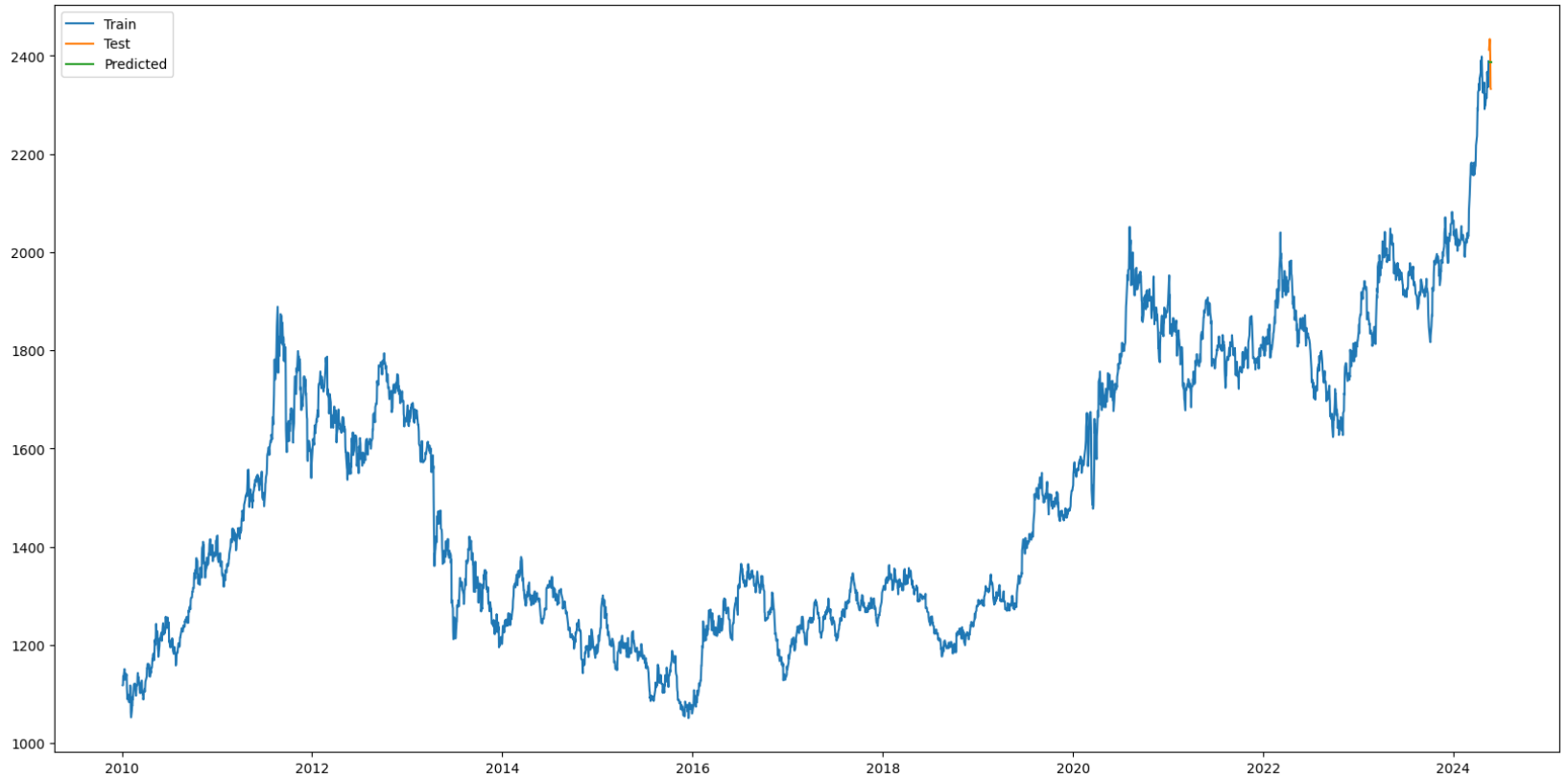
모델 학습 및 최적화

```
model2.plot_diagnostics(figsize=(16,8))
plt.show()
```



2. ARIMA

모델 학습 및 최적화




```
actual prices : [2380.0, 2412.199951, 2433.899902, 2421.699951, 2389.199951, 2335.0, 2332.5]
predicted prices : [2387.40019722 2387.40019722 2387.40019722 2387.40019722 2387.40019722
 2387.40019722 2387.40019722]
MAE : 31.72850825480899
```

3. HMM

데이터 전처리

- **결측치 처리:** 결측치가 존재하는 데이터 포인트 식별, 결측치 제거를 통한 데이터 정리

- **파생 변수 생성:**
  1. HMM 모델에서 사용할 데이터 생성
  2. 각 데이터셋의 'Close' 값만 추출하여 새로운 데이터프레임 생성

각 금 가격의 종가를 예측하는 task 라서 각기 종가만 사용하기로 정함

VIF(**분산 팽창 인자) 를 이용하여 다중 공산성 확인**

3. HMM

데이터 전처리

•VIF(분산 팽창 인자) 를 이용하여 다중 공산성 확인

Initial VIF Results (Before Stationarity Check):

	Variable	VIF
0	GC	538.244641
1	CL	1069.506131
2	BZ	1236.357163
3	NG	16.164231
4	SI	358.757757
5	HG	378.456906
6	USDKRW	2399.046359
7	USDEUR	3537.847559
8	USDCNY	1994.951350
9	BTCUSD	29.902723
10	ETHUSD	35.239365

다중 공산성이 매우 높아 feature을 줄일 필요가 있음

3. HMM

데이터 전처리

GC 정상성 없음, 차분 진행. 1번째 차분
GC 정상성 확보됨. 차분 횟수: 1
CL 정상성 없음, 차분 진행. 1번째 차분
CL 정상성 확보됨. 차분 횟수: 1
BZ 정상성 없음, 차분 진행. 1번째 차분
BZ 정상성 확보됨. 차분 횟수: 1
NG 정상성 없음, 차분 진행. 1번째 차분
NG 정상성 확보됨. 차분 횟수: 1
SI 정상성 없음, 차분 진행. 1번째 차분
SI 정상성 확보됨. 차분 횟수: 1
HG 정상성 없음, 차분 진행. 1번째 차분
HG 정상성 확보됨. 차분 횟수: 1
USDKRW 정상성 없음, 차분 진행. 1번째 차분
USDKRW 정상성 확보됨. 차분 횟수: 1
USDEUR 정상성 없음, 차분 진행. 1번째 차분
USDEUR 정상성 확보됨. 차분 횟수: 1
USDCNY 정상성 없음, 차분 진행. 1번째 차분
USDCNY 정상성 확보됨. 차분 횟수: 1
BTCUSD 정상성 없음, 차분 진행. 1번째 차분
BTCUSD 정상성 확보됨. 차분 횟수: 1
ETHUSD 정상성 없음, 차분 진행. 1번째 차분
ETHUSD 정상성 확보됨. 차분 횟수: 1

dicky-fuller 을 이용해 정상성 검정

```python
def check_stationarity(series):
    result = adfuller(series.dropna())  # NaN 값 제거
    return {'ADF Statistic': result[0], 'p-value': result[1], 'Critical Values': result[4]}
```

•가우시안 마르코프 모델 사용에는 데이터의 정상성이 보장되어야 하기 때문에, 코드를 이용해 정상성 확인및 차분으로 정상성 확보

3. HMM

데이터 전처리

정상성을 확보한 데이터의 다중 공산성 확인

Final VIF Results (After Stationarity Check):

	Variable	VIF
0	GC	2.508886
1	CL	1.607999
2	BZ	1.711675
3	NG	1.008714
4	SI	2.658282
5	HG	1.261598
6	USDKRW	1.410137
7	USDEUR	1.345436
8	USDCNY	1.199697
9	BTCUSD	2.277007
10	ETHUSD	2.285869

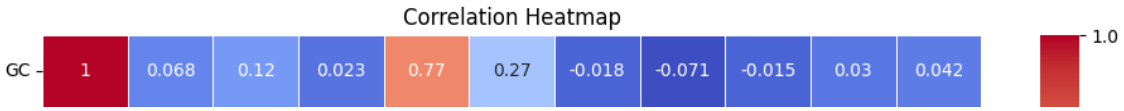
정상적인 범위(10보다작음) 안으로 다 들어옴

3. HMM

데이터 전처리

상관계수가 너무 낮을 경우 모델에 필요가 없으므로

금 종가와의 상관계수가 0.15보다 작을경우 제거




```python
# 금 종가와의 상관계수가 0.15보다 낮은 경우 제거
threshold = 0.15
low_corr_cols = corr.index[abs(corr['GC']) < threshold].tolist()
if 'GC' in low_corr_cols:
    low_corr_cols.remove('GC')  # 금 종가는 제외
data.drop(columns=low_corr_cols, inplace=True)
data = data.fillna(0)
```

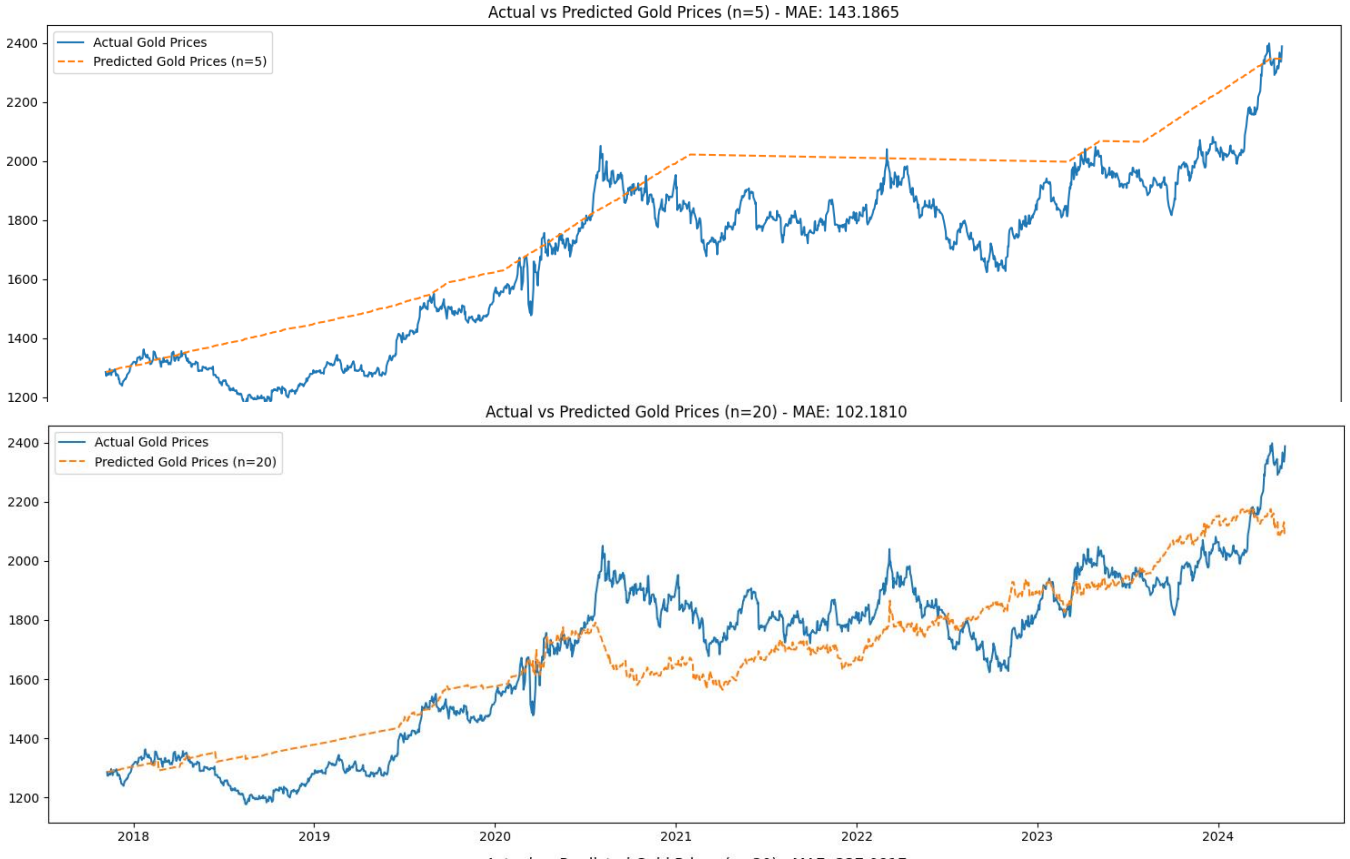
3. HMM

모델 훈련

가우시안 히든 마르코프 모델을 사용하여, 히든 layer 별로 train 후 비교해 최적의 모델 선정,

여러 번 실험해 최적의 모델 저장 (mae 가 가장 낮은 모델)

차분을 이용하여 데이터를 사용 했으니 Mae 를 구하거나 plot 을 그릴 때는 다시 원래의 값으로 복원




```
    # 예측된 차분을 실제 가격으로 복원
    predicted_prices = np.r_[data_origin['GC'].iloc[0], predicted_diffs].cumsum()
```

3. HMM

모델 테스트

가우시안 히든 마르코프 모델을 사용하여, 히든 layer 별로 train 후 비교해 최적의 모델 선정

	Date	Predicted_Gold_Price	Actual_Gold_Price
0	2024-05-16	2405.218587	2380.000000
2	2024-05-18	2375.465482	2412.199951
3	2024-05-19	2348.967408	2433.899902
4	2024-05-20	2347.151661	2433.899902
5	2024-05-21	2344.755917	2421.699951
6	2024-05-22	2343.605001	2389.199951
7	2024-05-23	2341.789254	2335.000000
8	2024-05-24	2342.576033	2332.500000
9	2024-05-25	2343.362813	2332.500000
10	2024-05-26	2344.149592	2332.500000
11	2024-05-27	2344.936371	2332.500000

MAE: 37.16066683180179

4. XgBoost

데이터 전처리

•**결측치 처리:**결측치가 존재하는 데이터 포인트 식별, 후방 채우기 방법으로 결측치 보완

•**파생 변수 생성** : 날짜 관련 변수 생성 (년, 월, 일, 요일), 5일, 15일 이동 평균 및 이동 표준편차 기울기 계산, 기타 통계적 특성 (평균, 표준편차, 최솟값, 최댓값 등)

```
GC=GC.reset_index()
GC.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3618 entries, 0 to 3617
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       3618 non-null   datetime64[ns]
 1   Open       3614 non-null   float64
 2   High       3614 non-null   float64
 3   Low        3614 non-null   float64
 4   Close      3614 non-null   float64
 5   Adj Close  3614 non-null   float64
 6   Volume     3614 non-null   float64
dtypes: datetime64[ns](1), float64(6)
memory usage: 198.0 KB
```

```
# 뒤의 값으로 채우기

GC=GC.fillna(method='bfill')
GC.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3618 entries, 0 to 3617
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       3618 non-null   datetime64[ns]
 1   Open       3618 non-null   float64
 2   High       3618 non-null   float64
 3   Low        3618 non-null   float64
 4   Close      3618 non-null   float64
 5   Adj Close  3618 non-null   float64
 6   Volume     3618 non-null   float64
dtypes: datetime64[ns](1), float64(6)
memory usage: 198.0 KB
```

4. XgBoost

데이터 전처리

- **결측치 처리:** 결측치가 존재하는 데이터 포인트 식별, 후방 채우기 방법으로 결측치 보완

- **파생 변수 생성** : 날짜 관련 변수 생성 (년, 월, 일, 요일), 5일, 15일 이동 평균 및 이동 표준편차 기울기 계산, 기타 통계적 특성 (평균, 표준편차, 최솟값, 최댓값 등)

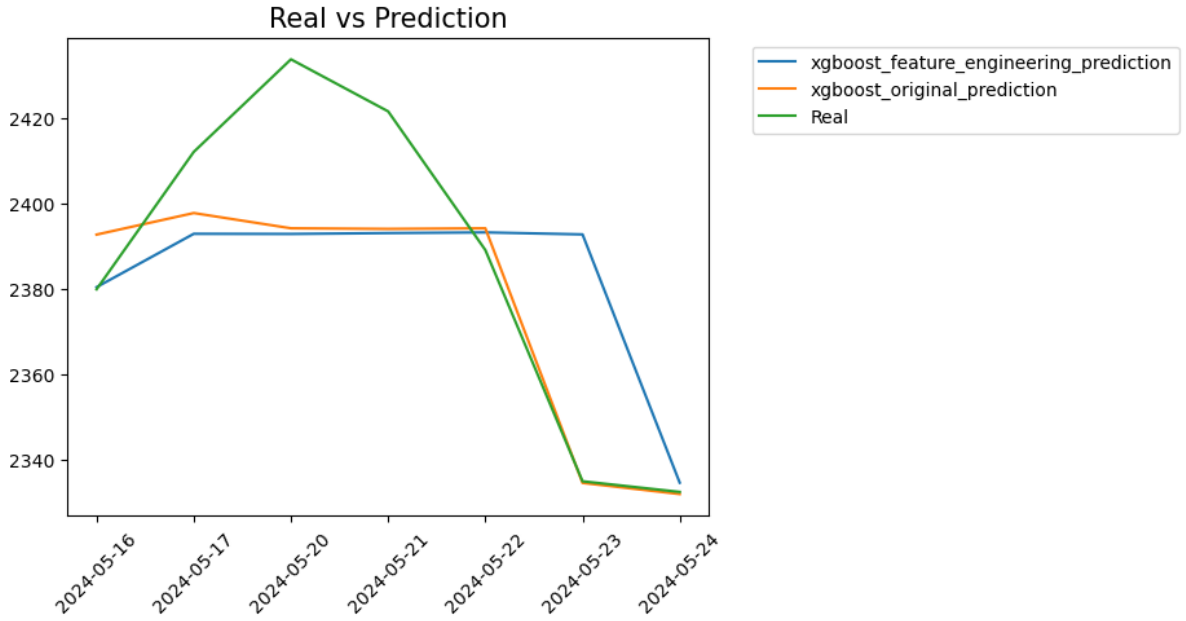
```
dt=GC['Date'].astype('str')
month_data=pd.to_datetime(dt)
GC['month']=month_data.dt.month
year_data=pd.to_datetime(dt)
GC['year']=year_data.dt.year
GC['day']=year_data.dt.day
GC['wd']=year_data.dt.weekday   ## 한 주의 요일을 나타내는 단어

GC['slope5'] = GC['Close'].rolling(5).apply(get_slope, raw=True)
GC['slope15'] = GC['Close'].rolling(15).apply(get_slope, raw=True)

GC['std5'] = GC['Close'].rolling(5).std(raw=True)
GC['std15'] = GC['Close'].rolling(15).std(raw=True)

GC['mean5'] = GC['Close'].rolling(5).mean(raw=True)
GC['mean15'] = GC['Close'].rolling(15).mean(raw=True)

GC['skew5'] = GC['Close'].rolling(5).skew()
GC['skew15'] = GC['Close'].rolling(15).skew()

GC['kurt5'] = GC['Close'].rolling(5).kurt()
GC['kurt15'] = GC['Close'].rolling(15).kurt()

GC['min5'] = GC['Close'].rolling(5).min()
GC['min15'] = GC['Close'].rolling(15).min()

GC['max5'] = GC['Close'].rolling(5).max()
GC['max15'] = GC['Close'].rolling(15).max()

print('파생변수 완료')
GC_data=GC.iloc[15:,:]                  # 이동평균 15일 이전 값들을 None값이 들어 있기 때문에 없애기
GC_data.drop(['Date'],inplace=True,axis=1)
```

4. XgBoost

모델 결과

- **하이퍼 파라미터 튜닝:** 5월 1일부터 5월 15일값들을 validation set으로 놓고 튜닝

```
(hjb) iai@iai-MS-7D76:~/heo/project_gold$ python xgboost_test.py
파생변수를 포함하고 학습한 xgboost model:
MAE:  14.379333410156278
파생변수를 포함하지 않고 학습한 xgboost model:
MAE:  14.323869904017881
(hjb) iai@iai-MS-7D76:~/heo/project_gold$
[64] 0:bash*
```



5. Conclusion

다양한 모델을 통한

1. 연구 결과 요약

이번 프로젝트에서는 다양한 원자재, 환율, 암호화폐 데이터를 활용하여 금 선물 가격을 예측하는 모델을 개발했습니다. 데이터 수집부터 전처리, 파생 변수 생성, 모델 학습 및 평가에 이르기까지의 과정을 통해 예측 모델의 성능을 향상시키기 위한 다양한 기법을 적용했습니다.

5. Conclusion

다양한 모델을 통한

2. 주요 발견 사항

•데이터 전처리의 중요성

- 결측치 제거 및 파생 변수 생성을 통해 데이터의 질을 향상시켰습니다. 특히, 5일 및 15일 이동 평균과 표준편차 등의 통계적 특성을 활용한 모델은 더 나은 예측 성능을 보였습니다.

•모델 성능

ARIMA 모델

- 시계열 분석을 통해 금 선물 가격의 트렌드와 계절성을 반영한 ARIMA 모델을 구축했습니다. ARIMA 모델은 데이터의 자기상관성을 활용하여 단기 예측에 유리한 성능을 보였습니다.

HMM 모델

- HMM(Hidden Markov Model)을 사용하여 금 선물 가격 예측의 잠재적인 상태 전이를 분석했습니다. HMM 모델은 다양한 시장 상태를 반영할 수 있는 유용한 도구임을 확인했습니다.

XgBoost 모델

- XGBoost 모델을 통해 파생 변수의 유효성을 검증하고, 최적의 하이퍼파라미터를 찾기 위한 Grid Search를 수행하여 모델 성능을 극대화했습니다.

5. Conclusion

다양한 모델을 통한

3. 성과 및 한계:

성과

- ARIMA, HMM, XGBoost 모델을 결합하여 금 선물 가격 예측 모델을 구축하였고, 이를 통해 다양한 변수와 상태를 고려한 예측을 가능하게 했습니다.
- Mean Absolute Error (MAE) 등의 성능 지표를 통해 모델의 예측 정확도를 평가하였으며, 이는 향후 연구 및 실무 적용에 유용한 기준을 제공했습니다.

한계

- 일부 데이터의 변동성이 큰 관계로 예측의 불확실성이 존재하였습니다. 이는 추가적인 데이터 수집 및 분석을 통해 보완될 필요가 있습니다.
- ARIMA 모델의 경우, 장기 예측에 대한 성능이 제한적일 수 있습니다.
- HMM 모델의 복잡성으로 인해 실시간 예측에는 다소 시간이 소요될 수 있습니다.

5. Conclusion

다양한 모델을 통한

4. 향후 연구 방향:

다양한 모델 적용:

- 향후 연구에서는 딥러닝 기반의 시계열 모델(LSTM, GRU 등)을 적용하여 예측 성능을 더욱 향상시킬 계획입니다.
- 강화 학습을 통한 최적 거래 전략 개발 등 다양한 접근 방식을 탐구할 예정입니다.

추가 데이터 및 변수:

- 더 많은 원자재 및 금융 데이터를 포함하여 모델의 예측 범위를 확장할 것입니다.
- 외부 요인(정책 변화, 경제 지표 등)을 고려한 모델링을 통해 예측의 정밀도를 높이겠습니다.

5. Conclusion

다양한 모델을 통한

5. 실무 적용 가능성

•본 연구의 결과는 금융 시장에서의 실시간 예측 및 리스크 관리 시스템에 직접적으로 활용될 수 있습니다.
•다양한 금융 자산의 가격 변동성을 예측하는 데 유용한 도구로서의 가치를 지니고 있으며, 이를 통해 투자 전략 수립 및 의사 결정 지원 시스템을 개선할 수 있습니다.