



Gold Price Prediction

발표자 : 김홍범

DataSet

◆ <https://www.kaggle.com/datasets/altruistdelhite04/gold-price-data>

◆ `gld_price_data.csv`

◆ 여러 다른 지표들을 바탕으로 금값을 예측

Gold Price Data



Data Code (38) Discussion (1)

About Dataset

Data Overview: This data file is a Comma separated value file format with 2290 rows and 7 columns. It contains 5 columns which are numerical in datatype and one column in Date format. Clearly the data shows value of the variables SPX,GLD,USO,SLV,EUR/USD against the dates in the date column.

Usability ⓘ

3.53

License

Unknown

Expected update frequency

Not specified

Data Explorer

Version 1 (130.56 kB)

gld_price_data.csv

`gld_price_data.csv` (130.56 kB)



Detail Compact Column

6 of 6 columns ▾

Date

SPX

GLD

US



DataSet

```
[Info]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2290 entries, 0 to 2289
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  -
 0   Date        2290 non-null   object
 1   SPX         2290 non-null   float64
 2   GLD         2290 non-null   float64
 3   USO         2290 non-null   float64
 4   SLV         2290 non-null   float64
 5   EUR/USD     2290 non-null   float64
dtypes: float64(5), object(1)
memory usage: 107.5+ KB
None
```

data_frame.info()

- column의 종류
- Dtype

DataSet

[Head]

	Date	SPX	GLD	USO	SLV	EUR/USD
0	1/2/2008	1447.160034	84.860001	78.470001	15.180	1.471692
1	1/3/2008	1447.160034	85.570000	78.370003	15.285	1.474491
2	1/4/2008	1411.630005	85.129997	77.309998	15.167	1.475492
3	1/7/2008	1416.180054	84.769997	75.500000	15.053	1.468299
4	1/8/2008	1390.189941	86.779999	76.059998	15.590	1.557099

[Tail]

	Date	SPX	GLD	USO	SLV	EUR/USD
2285	5/8/2018	2671.919922	124.589996	14.0600	15.5100	1.186789
2286	5/9/2018	2697.790039	124.330002	14.3700	15.5300	1.184722
2287	5/10/2018	2723.070068	125.180000	14.4100	15.7400	1.191753
2288	5/14/2018	2730.129883	124.489998	14.3800	15.5600	1.193118
2289	5/16/2018	2725.780029	122.543800	14.4058	15.4542	1.182033

.head()

.tail()

DataSet

[describe]

	SPX	GLD	USO	SLV	EUR/USD
count	2290.000000	2290.000000	2290.000000	2290.000000	2290.000000
mean	1654.315776	122.732875	31.842221	20.084997	1.283653
std	519.111540	23.283346	19.523517	7.092566	0.131547
min	676.530029	70.000000	7.960000	8.850000	1.039047
25%	1239.874969	109.725000	14.380000	15.570000	1.171313
50%	1551.434998	120.580002	33.869999	17.268500	1.303297
75%	2073.010070	132.840004	37.827501	22.882500	1.369971
max	2872.870117	184.589996	117.480003	47.259998	1.598798

.describe()

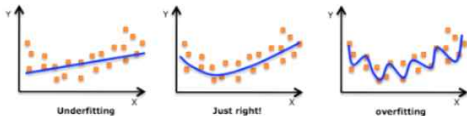
- mean : 평균
- std : 표준편차
- min : 최솟값
- max : 최댓값

고려사항

- 결측치

- Dataset의 결측, 누락, 손실 등

- 과적합



- 정규화/표준화

- scaling

- 검증

- KFold, Stratified KFold

DataSet

```
[isnull]      [isna]
Date          0    Date          0
SPX           0    SPX           0
GLD           0    GLD           0
USO           0    USO           0
SLV           0    SLV           0
EUR/USD       0    EUR/USD       0
dtype: int64     dtype: int64
```

```
.isnull().sum()
.isna().sum()
```

결측 데이터의 수를
합산

Visualization

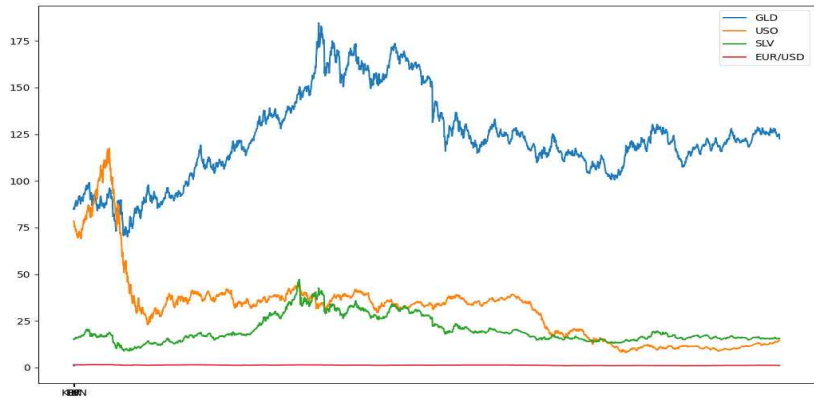
```
df = data_frame.drop(['Date'], axis=1)

plt.plot(df)
plt.show()

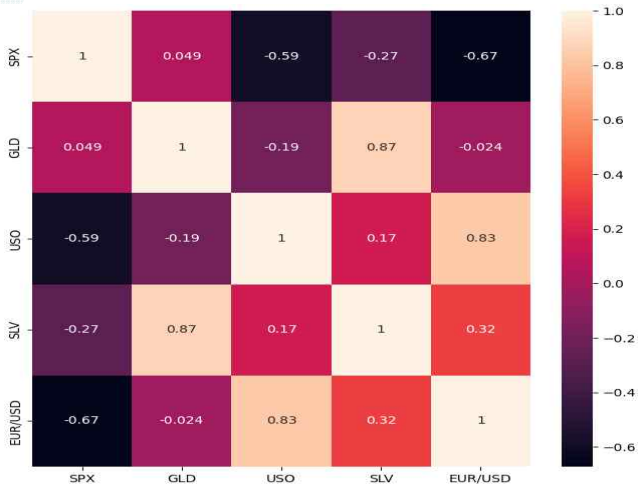
correlation = df.corr()
plt.figure(figsize=(8, 8))
sns.heatmap(correlation, annot=True)
plt.show()
```



Visualization



Visualization



Train Test Split

```
from sklearn.model_selection import train_test_split
```

```
X = df.drop(['GLD'], axis=1)
```

```
Y = df['GLD']
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

X : 문제지 Y : 정답지

X_train + Y_train

X_test + Y_test

Scaling

from sklearn.preprocessing import ---

◆ StandardScaler

- 평균이 0, 분산이 1 인 정규 분포
- 이상치가 존재한다면, 평균과 분산에 크게 영향을 줌

◆ MinMaxScaler

- 모든 데이터를 0과 1사이의 값으로 스케일

◆ MaxAbsScaler

- 절대값이 0과 1사이의 값이 되도록 스케일

Scaling

```
from sklearn.preprocessing import StandardScaler
```

```
std_scaler = StandardScaler()
```

```
std_scaler.fit(X_train)
```

```
X_train_scaled = std_scaler.transform(X_train)
```

```
X_test_scaled = std_scaler.transform(X_test)
```

```
plt.plot(X_train_scaled) # scaled data
```

```
print(X_train_scaled, "\n\n")
```

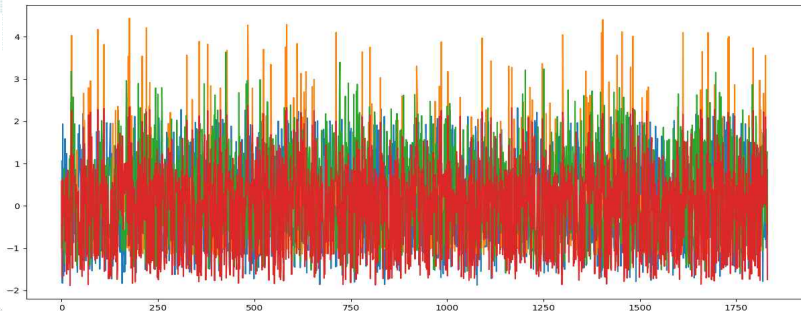
```
plt.show()
```

MinMaxScaler()

MaxAbsScaler()

RobustScaler()

```
[[-0.98462976  0.40800594 -0.45358541  0.58865226]
 [ 1.06599085 -1.10546089 -0.60775875 -1.69473821]
 [-1.82781083 -0.38661608 -1.03633275 -0.21860468]
 ...
 [-0.70024302  0.3467614  1.03157202  0.55478557]
 [-0.36162586  0.10334014  1.28475614  0.16734672]
 [ 0.8530957  -0.69699095 -0.63604747 -1.74527372]]
```



Regressor

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
```

```
DT = DecisionTreeRegressor(random_state=2)
DT.fit(X_train, Y_train)
not_scaled_DTscore = DT.score(X_test, Y_test)
DT.fit(X_train_scaled, Y_train)
scaled_DTscore = DT.score(X_test_scaled, Y_test)
```

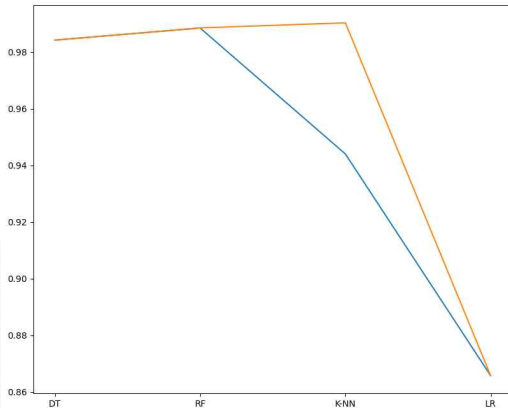
```
RF = RandomForestRegressor(n_estimators=100, random_state=2)
RF.fit(X_train, Y_train)
not_scaled_RFscore = RF.score(X_test, Y_test)
RF.fit(X_train_scaled, Y_train)
scaled_RFscore = RF.score(X_test_scaled, Y_test)
```

```
KN = KNeighborsRegressor(n_neighbors=2)
KN.fit(X_train, Y_train)
not_scaled_KNscore = KN.score(X_test, Y_test)
KN.fit(X_train_scaled, Y_train)
scaled_KNscore = KN.score(X_test_scaled, Y_test)

LR = LinearRegression()
LR.fit(X_train, Y_train)
not_scaled_LRscore = LR.score(X_test, Y_test)
LR.fit(X_train_scaled, Y_train)
scaled_LRscore = LR.score(X_test_scaled, Y_test)
```

random_state : 난수 seed
n_estimators : 생성할 트리의 수

Regressor



Not scaled 스코어: DT=0.98, RF=0.99, K-NN=0.94, LR=0.87

Scaled 스코어: DT=0.98, RF=0.99, K-NN=0.99, LR=0.87

Validation : KFold & Stratified KFold

```
from sklearn.model_selection import cross_val_score

basic_score = cross_val_score(RF, X_train_scaled, Y_train)
print('K폴드 검증 Scores : ', basic_score)
```

`cross_val_score(estimator, feature_X, label_Y, scoring, cv)`

`estimator` : 알고리즘

`feature` : X

`label` : Y

`scoring` : 지표 종류

`cv` : 횟수

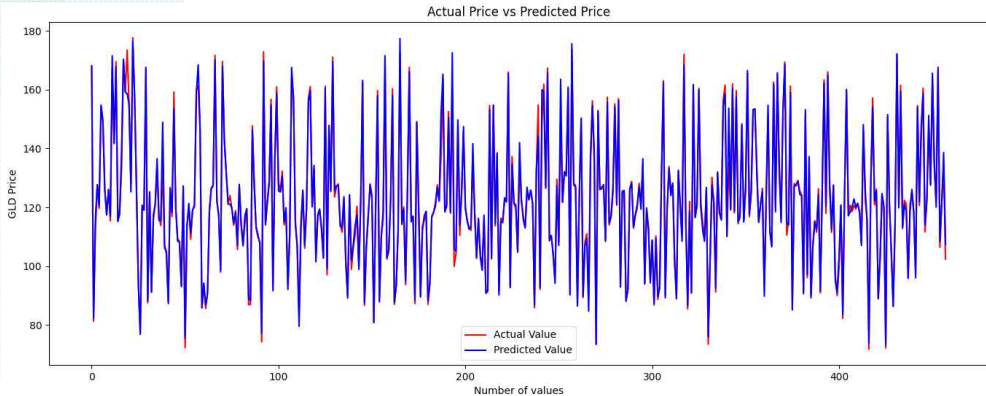
```
K폴드 검증 Scores : [0.99002462 0.99031065 0.9852354 0.98658362 0.99006894]
```

Predict

```
test_data_prediction = RF.predict(X_test_scaled)
Y_test = list(Y_test)

plt.plot(Y_test, color='red', label='Actual Value')
plt.plot(test_data_prediction, color='blue', label='Predicted Value')
plt.title('Actual Price vs Predicted Price')
plt.xlabel('Number of values')
plt.ylabel('GLD Price')
plt.legend()
plt.show()
```

Predict

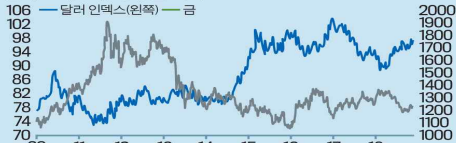


아쉬운점

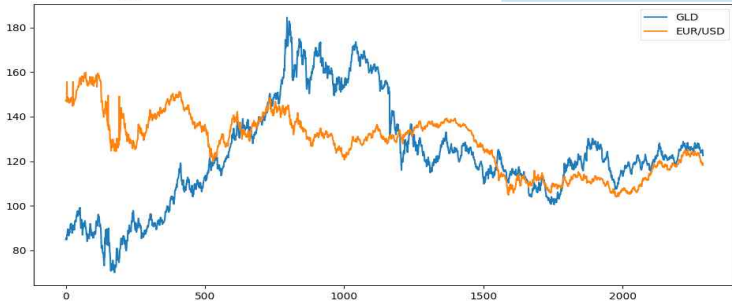
```
import pandas as pd
import matplotlib.pyplot as plt
# import pandas_profiling as pp
import seaborn as sns

data_frame = pd.read_csv('gld_price_data.csv')
```

달러 인덱스와 국제 금 시세



*세계의 주요 6개 통화(유로 · 엔 · 파운드 · 캐나다 달러 · 스웨덴 크로네 · 스위스 프랑) 대비 달러화의 상대적 가치를 가중평균한 지표, 자료 : 톰슨로이터



참고링크

◆ 전처리 기초 : <https://datascienceschool.net> - 머신러닝편 2.1 데이터 전처리 기초

◆ pandas : <https://javapp.tistory.com/161>

◆ 스케일링



▪ <https://jaaamj.tistory.com/20>

◆ random forest : <https://woolulu.tistory.com/28>

◆ 검증

