Import Library

```
import numpy as np    #numpy digunakan untuk membuat beberapa array numpy
import pandas as pd   #pandas berguna untuk membuat tabel terstruktur sehingga data kita akan berada dalam file csv, jadi pandas digunakan untuk menganalisis d
import matplotlib.pyplot as plt   #matplotlib berguna untuk membuat plot dan grafik
import seaborn as sns   #seaborn beerguna untuk membuat beberapa plot
from sklearn.model_selection import train_test_split    #Library ini untuk mengakses function train_test_split
from sklearn.ensemble import RandomForestRegressor    #Library ini untuk memanggil model RandomForestRegressor
from sklearn import metrics   #metrics berguna untuk menemukan skor kesalahan, skor akurasi dll
```

Pengumpulan Data (Data Gathering)

```
# loading data csv ke DataFrame Pandas
gold_data = pd.read_csv('gld_price_data.csv')
```

Exploratory Data Analysis (EDA) dan Persiapan Data (Data Pre-processing)

```
# Print 5 baris pertama dalam DataFrame
gold_data.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

SPX : SPX ini disebut indeks CSMP atau kapitalisasi indeks dari 500 perusahaan yang diperdagangkan secara publik, jadi itu salah satu saham yang tersedia.

GLD : GLD ini adalah nilai dari saham SPX, GLD ini ada GOLD yang mewakili harga emas jadi ini yang akan kita prediksi.

USO : USO ini mewakili harga minyak Amerika Serikat.

SLV : SLV adalah harga perak.

EUR/USD : Fitur tersebut adalah pasangan mata uang Eropa dan dolar Amerika Serikat, jika 1,47 mewakili satu Euro atau satu euro yang sama 1,47 USD

Model Machine Learning ini dengan harga saham tersebut jadi dengan pasangan mata uang EUR/USD dengan nilai SLV (perak) dan SPX dan USO dengan menganalisis nilai-nilai akan mencoba memprediksi nilai emas.

Dalam kasus ini akan memprediksi harga emas berdasarkan harga saham tersebut

```
# Print 5 baris terakhir dari DataFrame
gold_data.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

Mempunyai 2289 nilai dari tanggal 16 Mei 2018, jadi kita mempunyai data sekitar 10 tahun data yang cukup besar

```
# Nomor dari kolom dan baris dari DataFrame
gold_data.shape
```

```
(2290, 6)
```

Data yang dipakai mempunyai 2290 baris dan 6 kolom

```
# Mendapatkan beberapa informasi dasar tentang data
gold_data.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
```

Hasil dari informasi tersebut mempunyai 6 kolom dan dari semua kolom mempunyai 2290 nilai bukan nol yang berarti kita tidak memiliki missing values sehingga tipe datanya telah memberi kita objek untuk tanggal dan tipe data lainya ada di float64

```
# Mengecek missing values (hilang nilai)
gold_data.isnull().sum()    #Function sum() memberi kita jumlah misiing values di setiap kolom
```

```
Date       0
SPX        0
GLD        0
USO        0
SLV        0
EUR/USD    0
dtype: int64
```

```
#Mendapatkan ukuran statistik dari data
gold_data.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

count : count artinya jumlah titik data yang kita miliki yang totalnya 2290

mean : mean adalah nilai rata rata untuk dari semua feature

std : std adalah nilai standar

min : nilai minimum dari setiap feature

25% : artinya adalah 25 presentase nilai kurang dari jumlah nilai tersebut

50% : artinya adalah 50 presentase nilai kurang dari jumlah nilai tersebut

75% : artinya adalah 75 presentase nilai kurang dari jumlah nilai tersebut

max : adalah nilai maksimal dari setiap feature

Korelasi :

1. Korelasi Positif
2. Korelasi Negatif

Saat kita mengerjakan beberapa projek Regresi, kita akan selelau memeriksa korelasi. jadi, ini akan memberitahu kita kolom mana yang berhubungan kolom yang lain

Dalam Korelasi positif kita mengambil 2 variabel, 1 variabel akan meningkat dan variabel lain menurun sehingga jenis seperti itu disebut korelasi positif

Dalam Korelasi negatif jika satu nilai turun maka nilai yang lain naik jadi berbanding terbalik dengan korelasi positif

```
correlation = gold_data.corr()
```

```
<ipython-input-9-b9d572e5c3ef>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default
  correlation = gold_data.corr()
```

```
# Membuat heatmap untuk memahami korelasinya
plt.figure(figsize=(8, 8))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f', annot=True, annot_kws={'size': 16, 'weight': 'bold'}, cmap='Greens')
plt.show()
```

Kita dapat melihat plotnya disini sehingga korelasi negatif memiliki nilai negatif dan korelasi positif memiliki nilai positif, dalam kasus ini nilainya terletak di paling kanan antara -0.6 dan 1.0 berarti keduanya berkorelasi positif

Feature yang menarik disini adalah GLD (emas), jadi kami akan memprediksi harga emas sehingga kami dapat melihat feature mana yang berkorelasi positif, jika dilihat pada kolom SLV dan GLD yang bernilai 0.9 yang berarti berkorelasi artinya jika harga emas naik harga Perak juga akan meningkat

```
#Nilai korelasi dari GLD
print(correlation['GLD'])
```

```
SPX        0.049345
GLD        1.000000
USO       -0.186360
SLV        0.866632
EUR/USD   -0.024375
Name: GLD, dtype: float64
```

Kita bisa melihat korelasi untuk GLD, kita punya korelasi sedikit negatif dan perak berkorelasi positif

```
#Memeriksa distribusi dari harga emas
sns.distplot(gold_data['GLD'], color='Green')
```

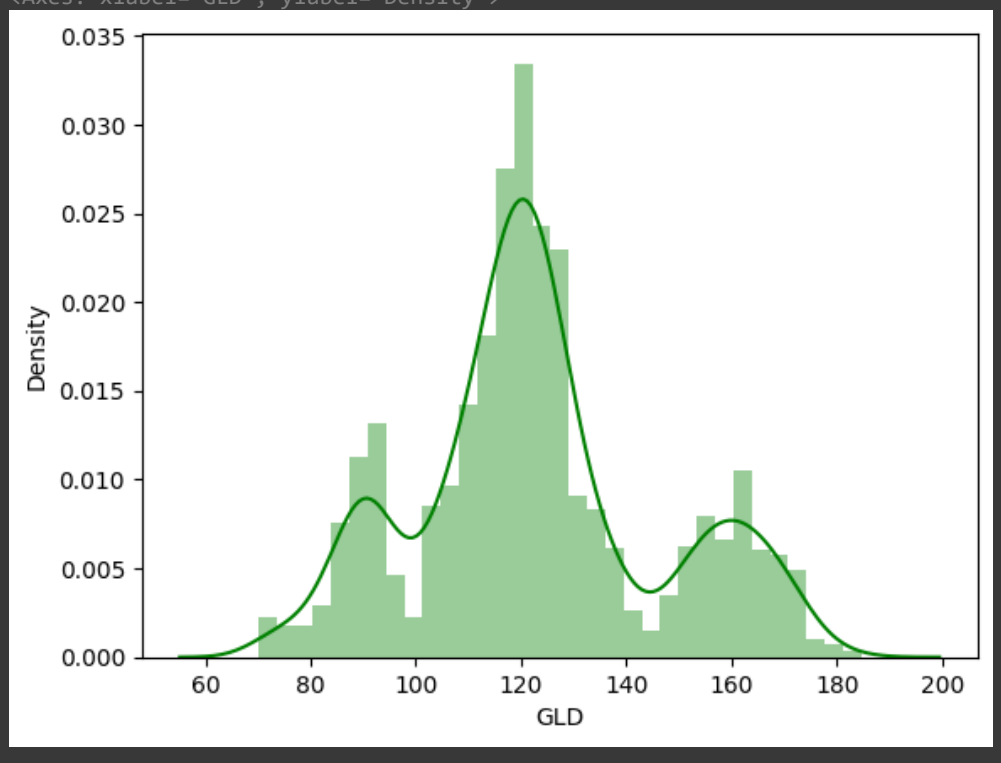
```
<ipython-input-12-01b2a9c7ae53>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(gold_data['GLD'], color='Green')
<Axes: xlabel='GLD', ylabel='Density'>
```



Kita bisa melihat disini nilai terbanyak terletak pada kisaran 120 dan nilai sedikit pada kisaran 180, terdapat pumping sekita 160 tapi disini paling banyak pumping pada 120

Memisahkan Feature dan Target

Targetnya adala harga emas dan Feature nya adalah harga saham SPX, USO dan fitur fitur lainnya.

```
X = gold_data.drop(['Date','GLD'],axis=1)
Y = gold_data['GLD']
```

```
print(X)
```

```
                SPX        USO      SLV   EUR/USD
    0     1447.160034  78.470001  15.1800  1.471692
    1     1447.160034  78.370003  15.2850  1.474491
    2     1411.630005  77.309998  15.1670  1.475492
    3     1416.180054  75.500000  15.0530  1.468299
    4     1390.189941  76.059998  15.5900  1.557099
    ...           ...        ...      ...       ...
    2285  2671.919922  14.060000  15.5100  1.186789
    2286  2697.790039  14.370000  15.5300  1.184722
    2287  2723.070068  14.410000  15.7400  1.191753
    2288  2730.129883  14.380000  15.5600  1.193118
    2289  2725.780029  14.405800  15.4542  1.182033

    [2290 rows x 4 columns]
```

Hasil output diatas sudah tidak ada feature GLD dan Date, dan hanya ada feature feature harga saham yang akan kita gunakan untuk memprediksi nilai harga emas

```
print(Y)
#   Y berisi semua harga emas
```

```
    0        84.860001
    1        85.570000
    2        85.129997
    3        84.769997
    4        86.779999
             ...
    2285    124.589996
    2286    124.330002
    2287    125.180000
    2288    124.489998
    2289    122.543800
    Name: GLD, Length: 2290, dtype: float64
```

Data Training dan Data Testing (Tarining Model dan Test Model)

Kita akan menggunakan algoritma machine learning dengan hasil data pelatihan dan kami akan mengevaluasi regressor menggunakan data pengujian

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

Data yang sudah dipisahkan pada prin(X)akan digunakan pada X_train dan X_test, 80% nilai akan masuk ke X_train dan 20% nilai akan masuk ke X_test dan harga emas yang sesuai akan masuk ke Y_train setelah itu akan masuk juga ke Y_test

Pemilihan dan Pembuatan Model: Random Forest Regressor

```
regressor = RandomForestRegressor(n_estimators=100)
```

```
#Training Model
regressor.fit(X_train, Y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor()
```

Evaluasi/Validasi Model (Evaluate Model)

```
#Prediksi pada Data Uji
train_data_prediction = regressor.predict(X_test)
```

```
print(train_data_prediction)
```

```
    [168.71959976  82.13479995 115.80119994 127.62480038 120.76980128
     154.76829798 150.75699856 126.20120029 117.41779872 125.98090103
     117.00870093 172.13020079 141.662999   167.50149811 115.15179956
     118.00350056 138.79680313 170.22960131 159.38770291 158.79739876
     155.04290013 125.1815006  175.64920071 156.7935027  125.23840039
      93.91919973  77.62870003 120.89589996 119.14650002 167.49569958
      88.20530059 125.06679985  91.2700008  117.74770036 121.02099913
     136.99260092 115.50740109 114.67390073 147.65119979 107.49050046
     104.3877025   87.39979801 126.39410077 117.93109984 153.4200989
     119.57390011 108.46899985 108.15179793  93.32230058 127.15569762
      74.65020056 113.76859957 121.25450024 111.27269905 118.78679899
     120.67039936 159.62780038 166.85100127 146.76649639  85.77339853
```

```
 94.39520025  86.79029884  90.6532001  119.08480059 126.49090062
127.3881001  169.96810033 122.1121995  117.26079915  98.68660019
167.722802   143.00539953 132.09020234 121.18870214 120.71969907
119.74400045 114.5220018  118.19040083 107.46840084 127.93360061
113.97189954 107.18690014 116.66610044 119.80799874  88.79950039
 88.26999884 146.41530257 126.96619976 113.5197003  109.98219852
108.27599902  77.63719864 169.10020185 114.00819913 121.72349931
127.83310221 154.98199826  91.75399936 136.53580044 158.95320288
125.7164007  125.51870085 130.48750209 114.69790152 119.64890007
 92.13749982 110.38679896 167.94519933 156.73639875 114.13779935
106.55970154  79.95739964 113.30810031 125.95400072 107.12679961
119.28450102 156.02870328 160.02249894 120.6229998  133.86790307
101.50499969 117.5898981  119.30040042 113.01700075 102.76349929
160.3357983   99.54480034 147.56969869 125.60040086 169.82469928
125.7287989  127.41559742 127.6028018  113.77629914 113.30720069
123.58919904 102.02899889  89.14840006 124.57319955 101.77529926
107.13199919 113.68830028 117.29990124  98.90319962 121.95710047
162.92409989  87.38609855 106.8139001  117.27820072 127.70730107
124.06930073  80.65689931 120.45380073 157.2828983   87.75979974
110.33229951 118.9612991  172.39489863 102.91949902 105.48450002
122.42400026 157.69299756  87.5163984   92.99890017 112.64170031
177.20230047 114.37430015 119.23490032  94.33060091 125.68620068
166.34560029 114.67350109 116.6510015   88.37839885 148.85640057
120.45699923  89.47510016 112.11980007 117.44940023 118.6414011
 88.23869925  93.90909978 116.82039981 118.41270198 120.19139985
126.79729821 121.94549983 150.63869937 165.04150128 118.54609981
120.3237015  150.80540114 118.47979929 172.70609868 105.03879942
104.95800154 149.46200086 113.66160062 124.79950146 147.35250031
119.63660115 115.27240017 112.71289995 113.46150163 140.91630137
117.81189777 102.82600013 115.88970122 103.84880192  98.67520038
117.24840072  90.69809975  91.49600016 153.61069875 102.71189982
154.88780142 114.38940121 138.51020077  90.08089819 115.50529934
114.53839994 123.19180009 121.85340025 165.37000147  93.00539946
135.89000157 121.37649903 120.83550084 104.61930031 142.59770286
122.25739898 116.6980006  113.51820082 127.10269745 122.62459964
125.81989979 121.2200002   86.74749893 132.385402   145.58340173
 92.7577994  158.66749917 158.37510243 126.18249904 164.52999937
108.7386997  110.49970082 103.63149819  94.34550082 127.57310261
107.12170076 161.69799951 122.01140021 132.07090006 130.5647018
160.71039976  90.08249857 173.9747019  128.10860029 126.74879866
 86.46779901 124.5903994  150.42269735  89.61140051 106.88459999
109.05679988  83.87089907 136.09539924 155.02650256 140.04250369
 74.13600009 151.89750122 126.31490054 126.75919988 127.60709906
108.78259934 156.40770054 114.54980138 116.98540131 125.13639929
154.24860143 121.28759986 156.37339862  92.86480052 125.52950157
125.46560022  88.02880062  92.07289894 126.12679974 128.37900347
```

Nilai diatas adalah hasil prediski oelh model kita, selanjutnya kita perlu membandingakkan nilai prediksi tersebut dengan nilai sebenarnya.

Untuk itu kita perlu menggunakan beberapa matriks

```
error_score = metrics.r2_score(Y_test, train_data_prediction)
print("Error pada kuadrat R : ", error_score)
```

```
Error pada kuadrat R :  0.9891932162643575
```

Kita dapat melihat hasil output diatas adalah 0.98

Tidak ada nilai detail untuk skor Error, kita bisa mengatakan bahwa hasil Error diatas merupakan hasil kinerja Model dengan baik

Untuk mengetahui analisa sebarapa baik model kita, selanjutnya yang akan kita lakukan adalah membandingkan nilai sebenarnya (Y_test) dan nilai prediksi (train_data_prediction) dengan memplotnya di dalam beberapa grafik dan melihat sebarapa dekat nilai-nilainya

Sebelum itu kita akan ubah Y_test menjadi list karena jika tidak akan error

```
Y_test = list(Y_test)
```

Sesudah di ubah ke list langkah selanjutnya melakukan perbandingan

Penggunaan Model dengan data baru untuk prediksi

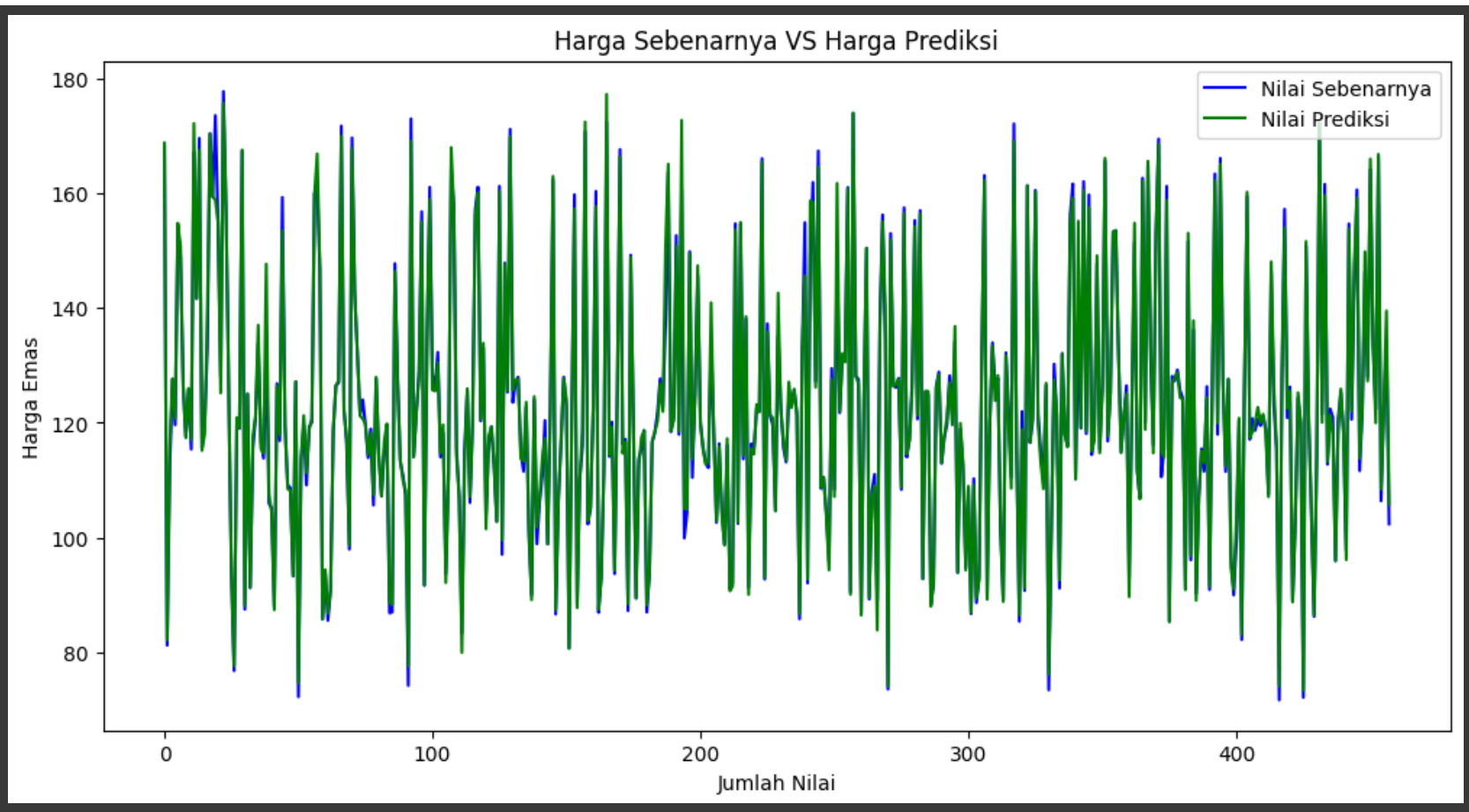
```
# Mengatur ukuran gambar grafik yang lebih besar
plt.figure(figsize=(12, 6))

# Plot prediksi nilai sebenarnya dan nilai prediksi
plt.plot(Y_test, color='blue', label='Nilai Sebenarnya')
plt.plot(train_data_prediction, color='green', label='Nilai Prediksi')

# Membuat judul dan label
plt.title('Harga Sebenarnya VS Harga Prediksi')
plt.xlabel('Jumlah Nilai')
plt.ylabel('Harga Emas')

# Menambahkan legend
plt.legend()

# Menampilkan plot
plt.show()
```



Kita dapat melihat kotak kecil di diatas samping kanan yang merupakan label harga sebenarnya warna biru dan harga prediksi warna hijau.

Sekarang kita bisa melihat harga sebenarnya dan harga prediksi, keduanya sangat dekat satu sama lain sehingga harga sebenarnya agak sedikit lebih banyak dari nilai prediksi tidak lain adalah hasil dari output diatas yang hasilnya 0,98

Output diatas artinya kita berhasil menggunakan Forest Regressor untuk menentukan harga emas berdasarkan beberapa harga saham lainnya