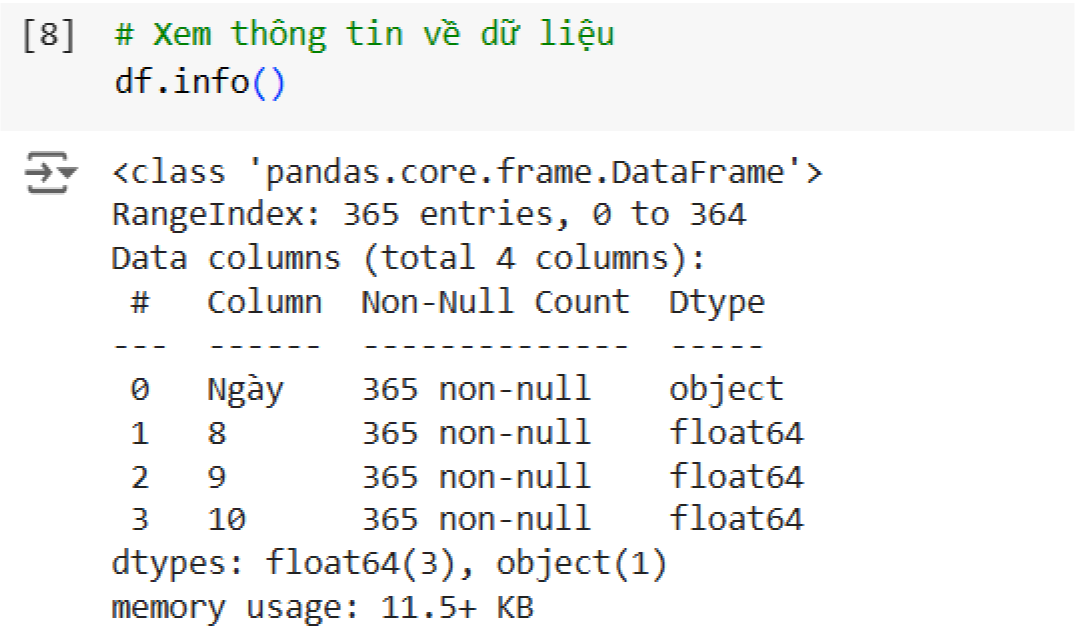
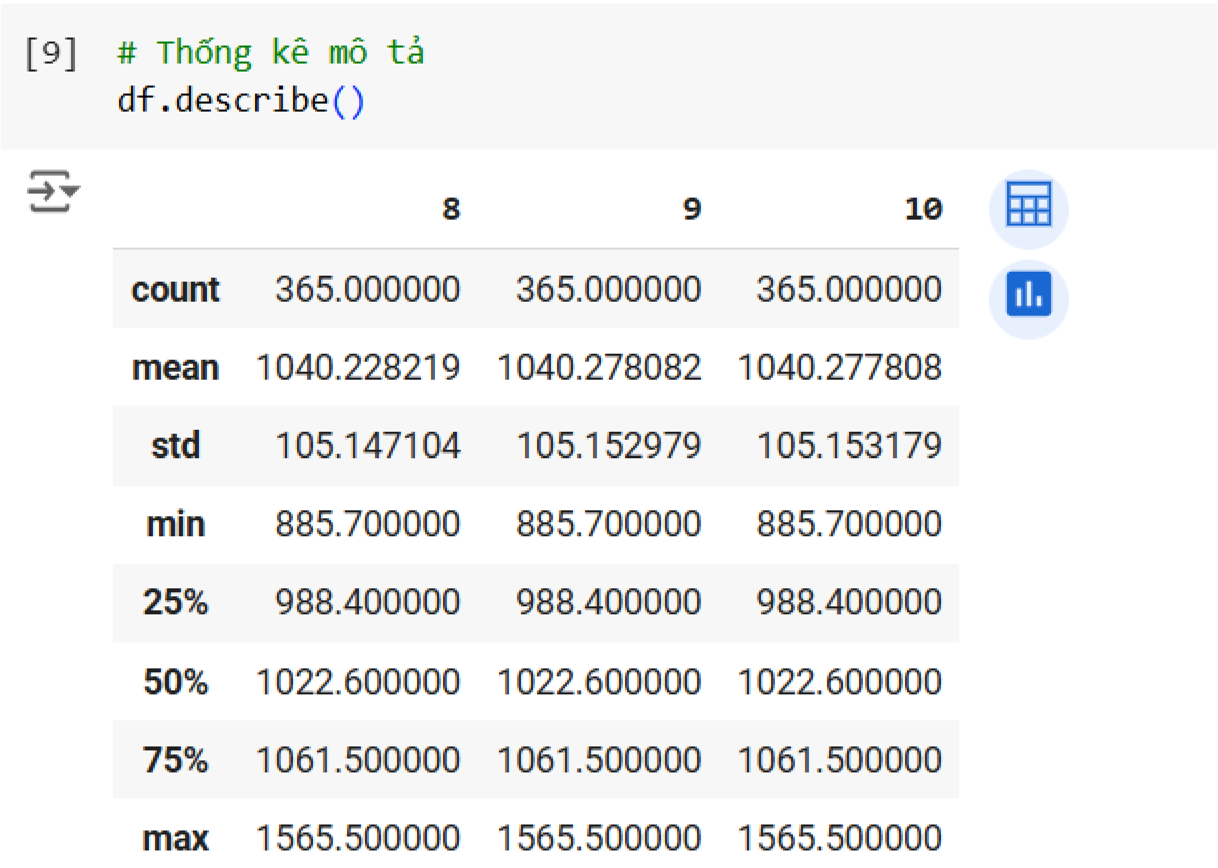
EDA:



```
[8] # Xem thông tin về dữ liệu
    df.info()
```

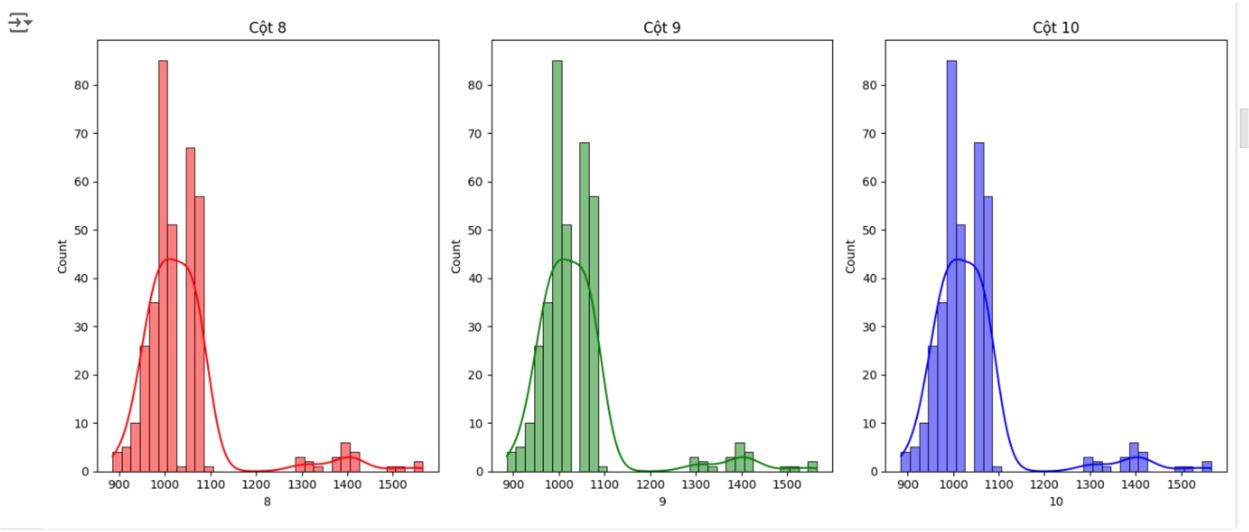
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 365 entries, 0 to 364
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Ngày    365 non-null    object 
 1   8       365 non-null    float64
 2   9       365 non-null    float64
 3   10      365 non-null    float64
dtypes: float64(3), object(1)
memory usage: 11.5+ KB
```

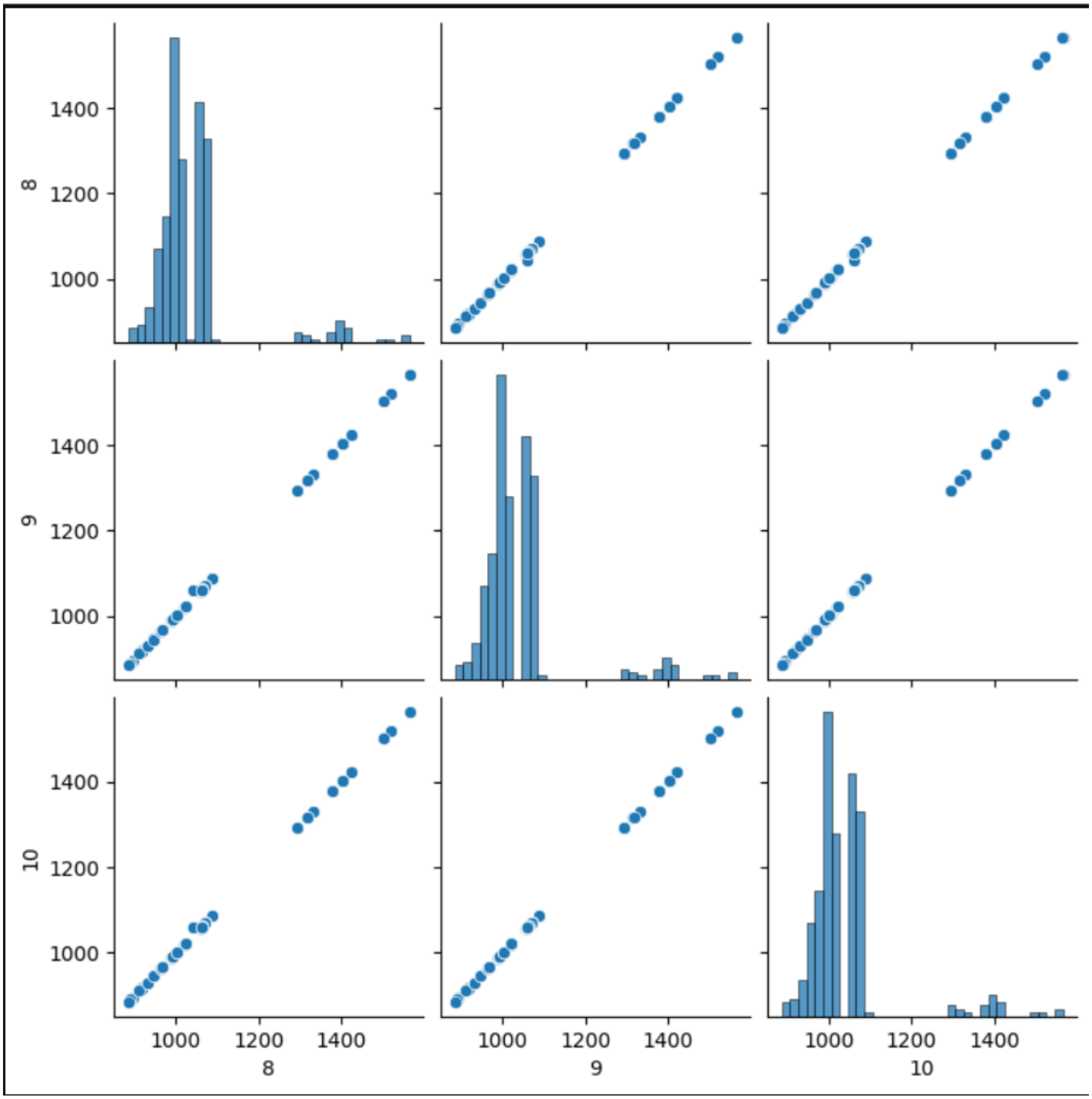


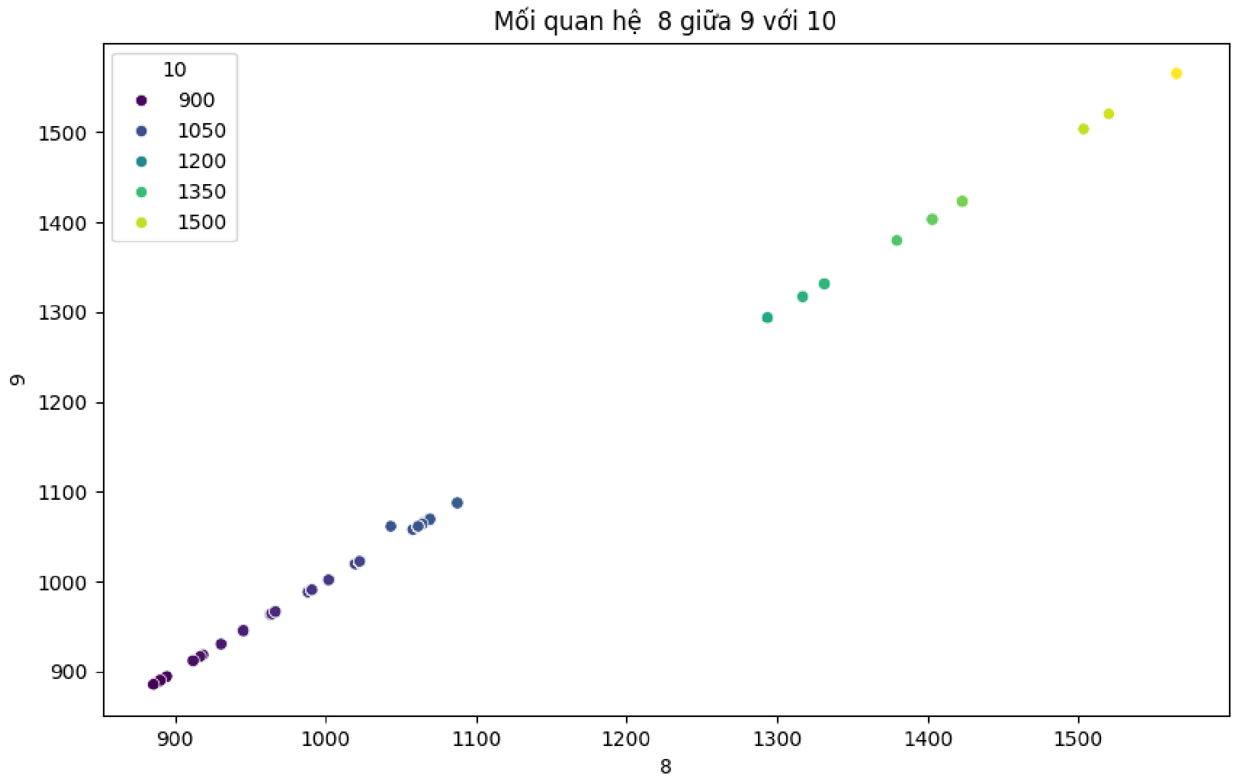
```
[9] # Thống kê mô tả
    df.describe()
```

	8	9	10
count	365.000000	365.000000	365.000000
mean	1040.228219	1040.278082	1040.277808
std	105.147104	105.152979	105.153179
min	885.700000	885.700000	885.700000
25%	988.400000	988.400000	988.400000
50%	1022.600000	1022.600000	1022.600000
75%	1061.500000	1061.500000	1061.500000
max	1565.500000	1565.500000	1565.500000

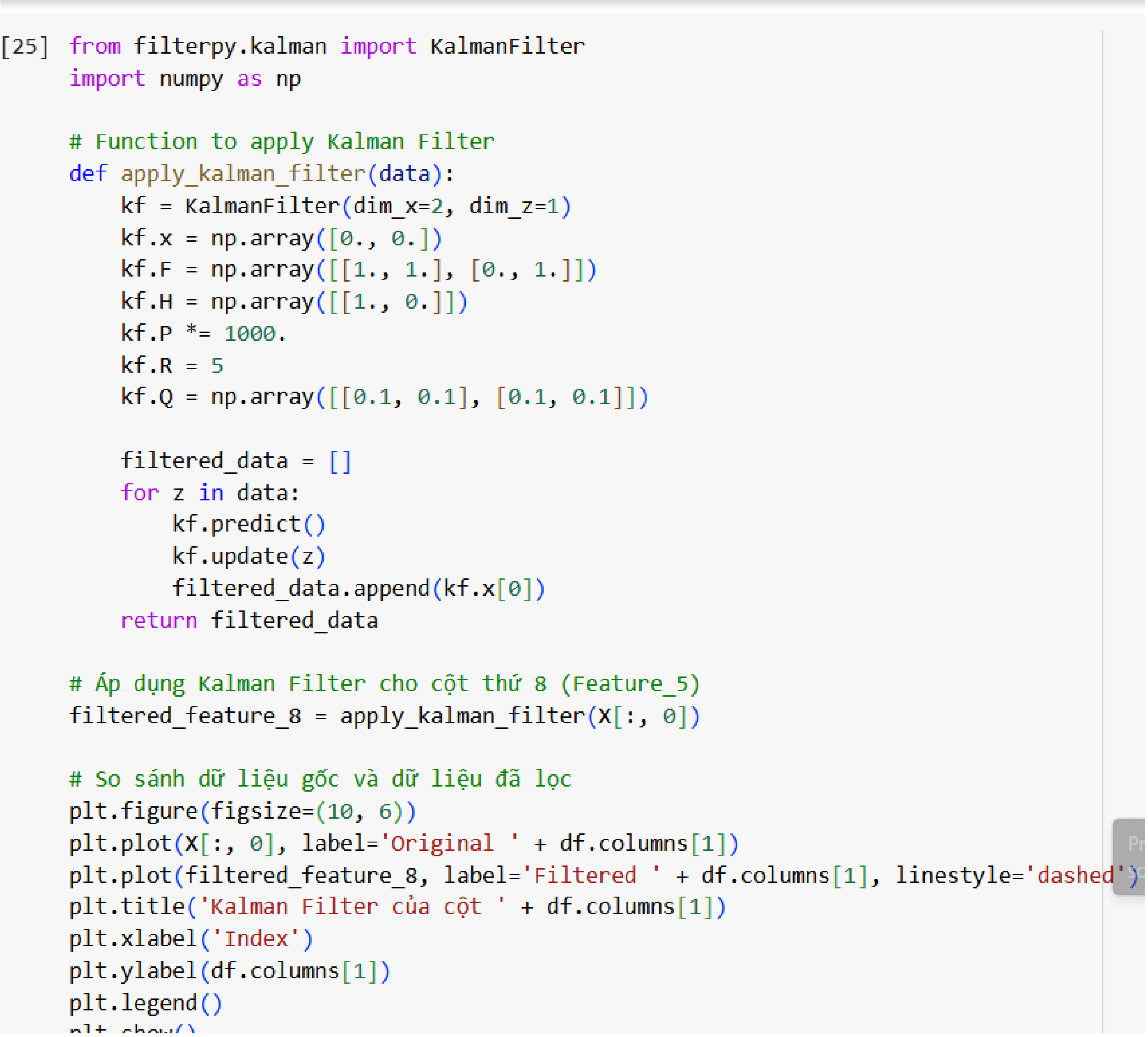
Phân phối dữ liệu của các cột:



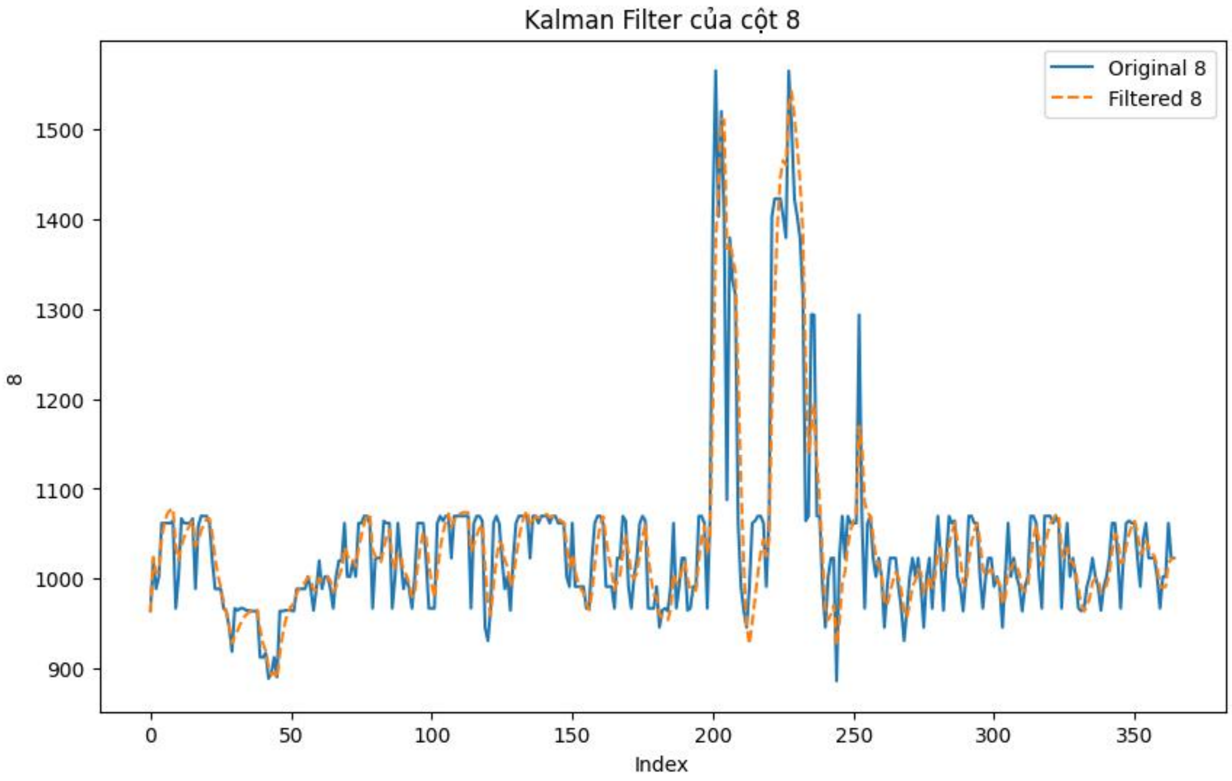

8
1400
1200
1000
9
1400
1200
1000
10
1400
1200
1000
1000
1200
1400
8
1000
1200
1400
9
1000
1200
1400
10

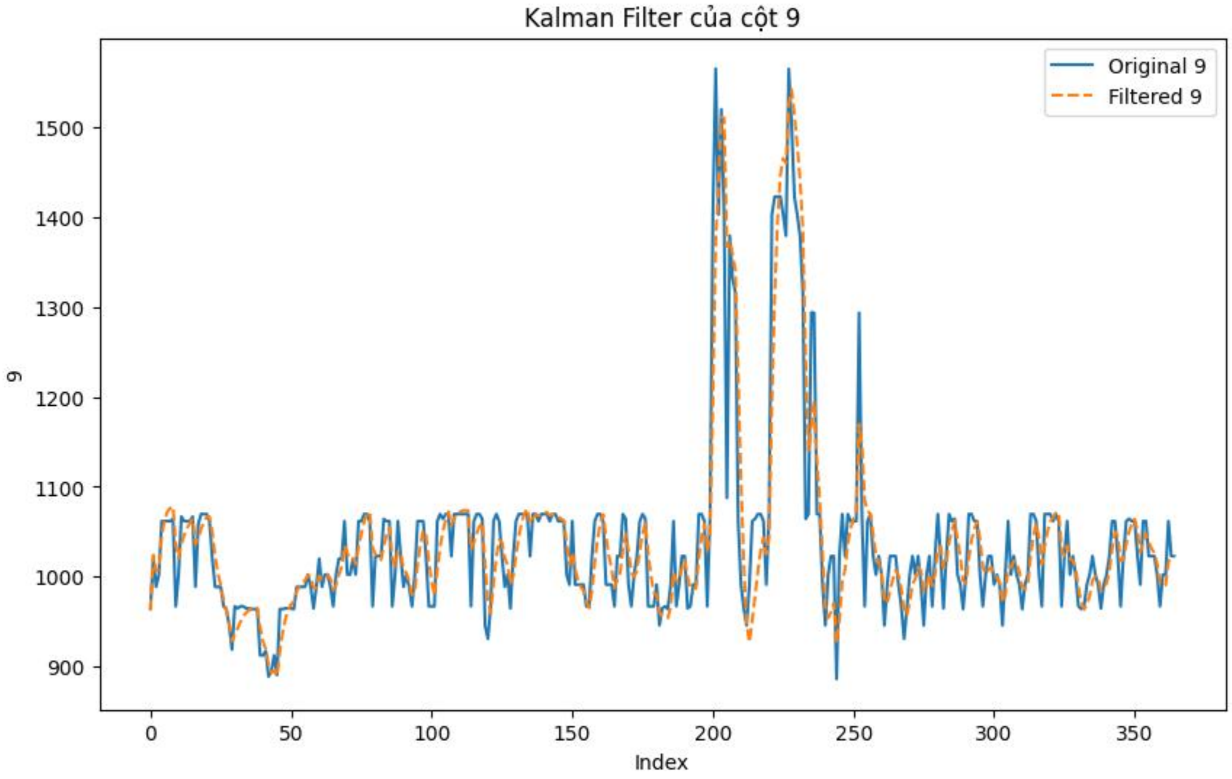

Mối quan hệ 8 giữa 9 với 10
10
900
1050
1200
1350
1500
1500
1400
1300
1200
1100
1000
900
9
900
1000
1100
1200
1300
1400
1500
8

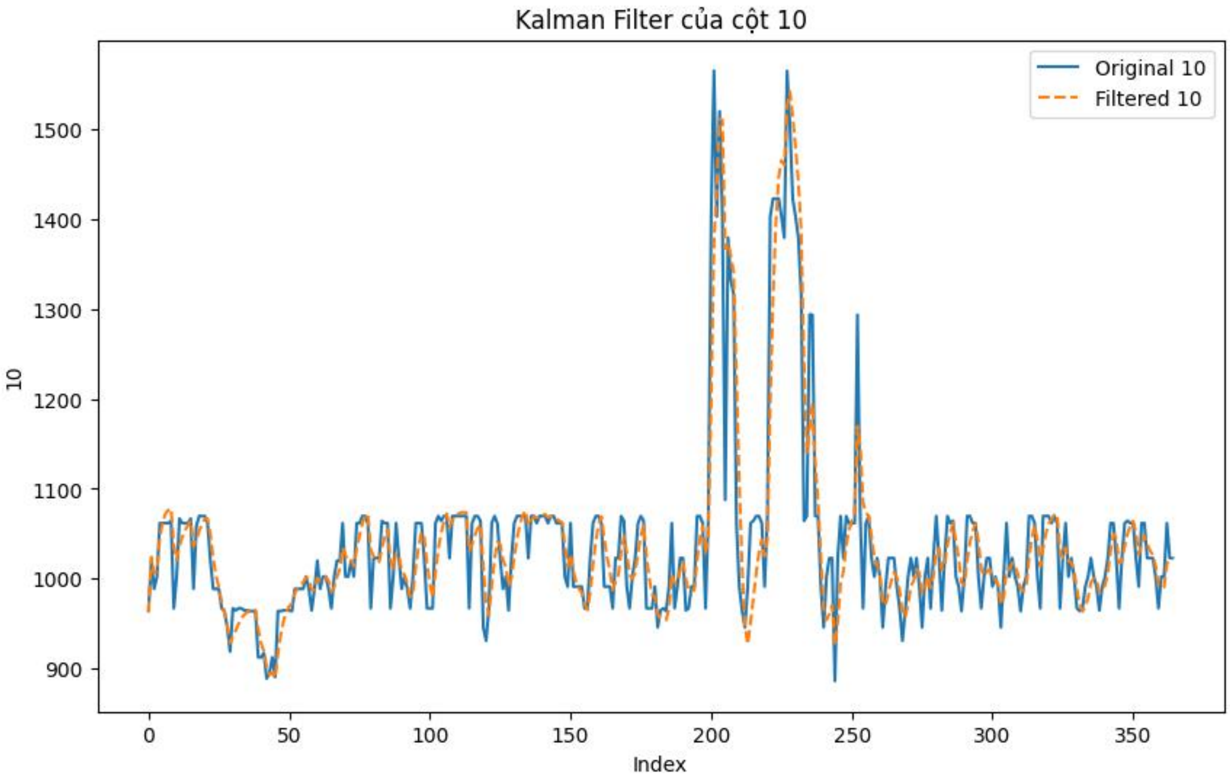
KALMAN:



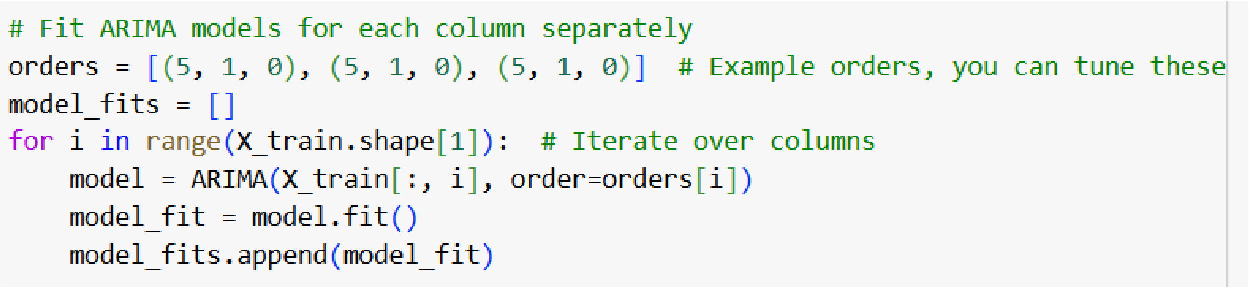
```
[25] from filterpy.kalman import KalmanFilter
     import numpy as np

     # Function to apply Kalman Filter
     def apply_kalman_filter(data):
         kf = KalmanFilter(dim_x=2, dim_z=1)
         kf.x = np.array([0., 0.])
         kf.F = np.array([[1., 1.], [0., 1.]])
         kf.H = np.array([[1., 0.]])
         kf.P *= 1000.
         kf.R = 5
         kf.Q = np.array([[0.1, 0.1], [0.1, 0.1]])

         filtered_data = []
         for z in data:
             kf.predict()
             kf.update(z)
             filtered_data.append(kf.x[0])
         return filtered_data

     # Áp dụng Kalman Filter cho cột thứ 8 (Feature_5)
     filtered_feature_8 = apply_kalman_filter(X[:, 0])

     # So sánh dữ liệu gốc và dữ liệu đã lọc
     plt.figure(figsize=(10, 6))
     plt.plot(X[:, 0], label='Original ' + df.columns[1])
     plt.plot(filtered_feature_8, label='Filtered ' + df.columns[1], linestyle='dashed')
     plt.title('Kalman Filter của cột ' + df.columns[1])
     plt.xlabel('Index')
     plt.ylabel(df.columns[1])
     plt.legend()
     plt.show()
```


Kalman Filter của cột 8
Original 8
Filtered 8
8
1500
1400
1300
1200
1100
1000
900
0
50
100
150
200
250
300
350
Index

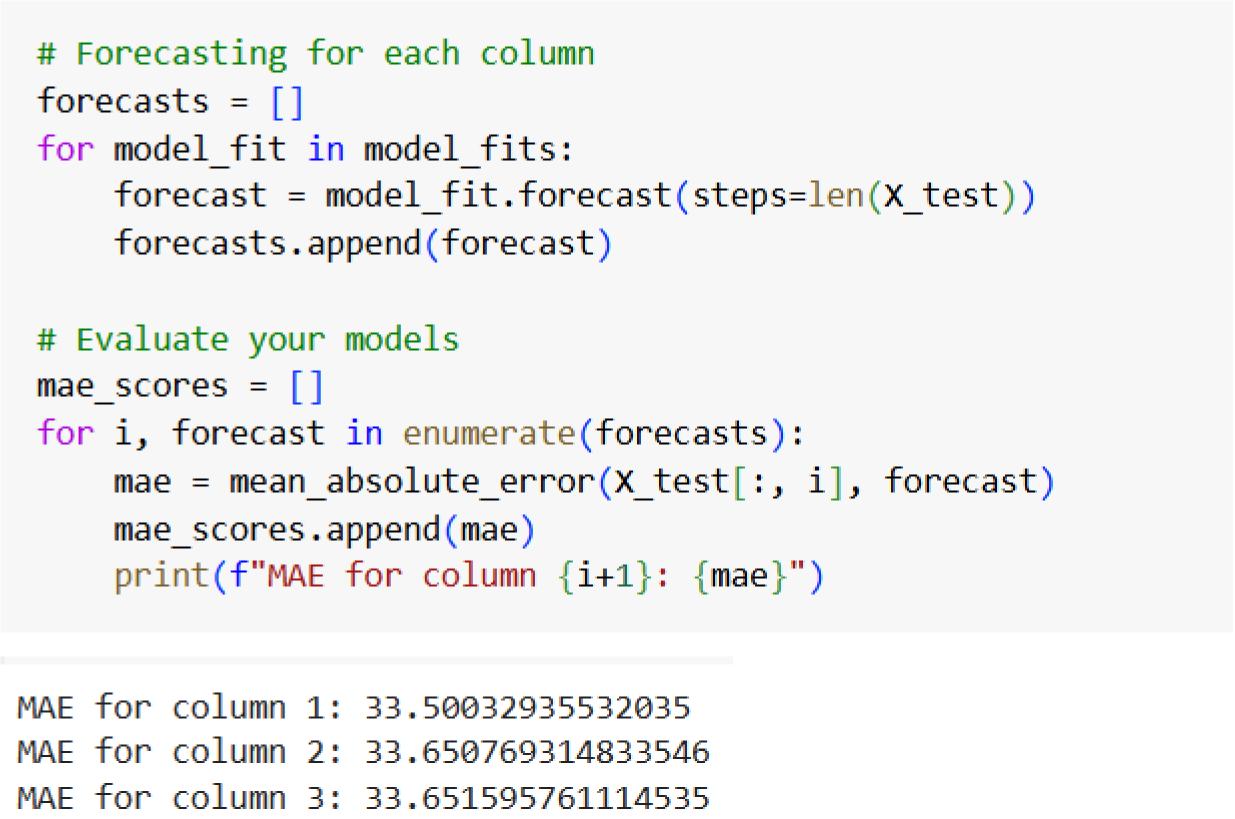

Kalman Filter của cột 9
Original 9
Filtered 9
9
1500
1400
1300
1200
1100
1000
900
0
50
100
150
200
250
300
350
Index



Arima:

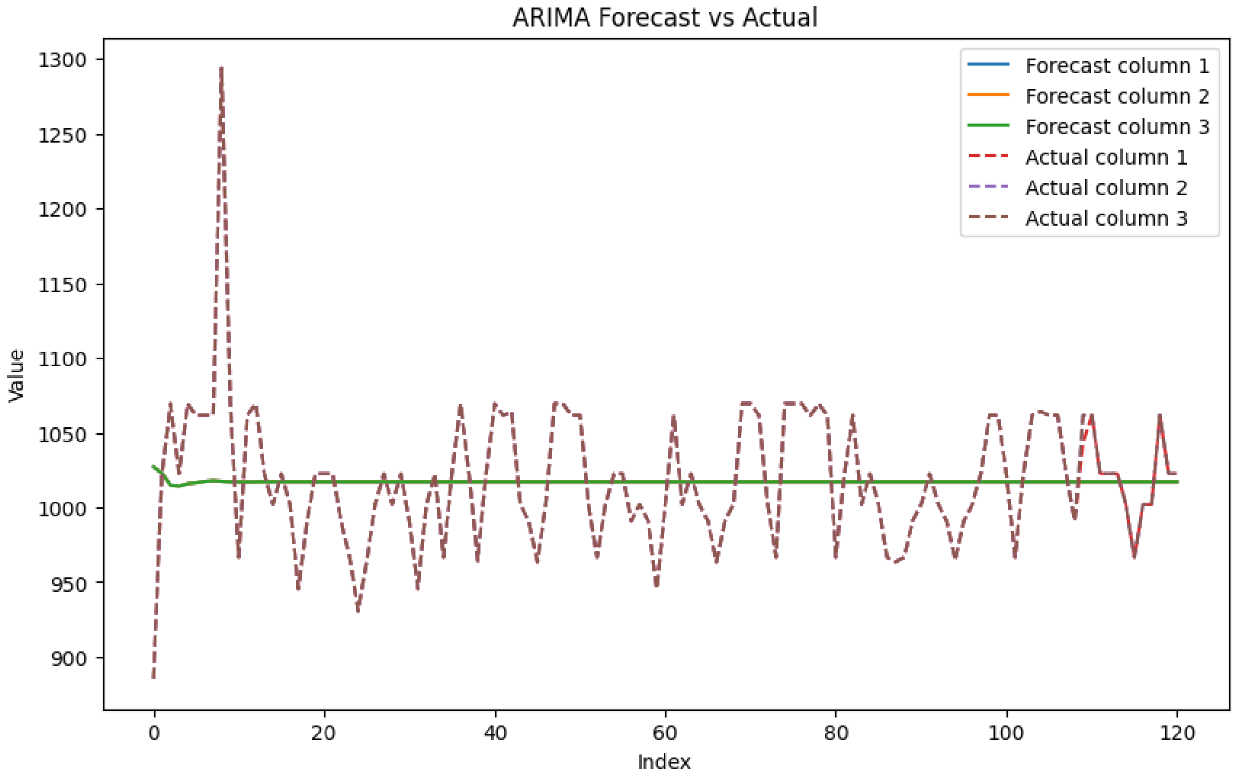


```
# Fit ARIMA models for each column separately
orders = [(5, 1, 0), (5, 1, 0), (5, 1, 0)]  # Example orders, you can tune these
model_fits = []
for i in range(X_train.shape[1]):  # Iterate over columns
    model = ARIMA(X_train[:, i], order=orders[i])
    model_fit = model.fit()
    model_fits.append(model_fit)
```



```python
# Forecasting for each column
forecasts = []
for model_fit in model_fits:
    forecast = model_fit.forecast(steps=len(X_test))
    forecasts.append(forecast)

# Evaluate your models
mae_scores = []
for i, forecast in enumerate(forecasts):
    mae = mean_absolute_error(X_test[:, i], forecast)
    mae_scores.append(mae)
    print(f"MAE for column {i+1}: {mae}")
```

```
MAE for column 1: 33.50032935532035
MAE for column 2: 33.650769314833546
MAE for column 3: 33.651595761114535
```



CuongNgD203/TH3_TimeSeries (github.com)