Đọc data, chọn 3 cột: 6,7,8

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```
✓ 0.0s

```
df = pd.read_csv('data.csv', encoding='latin-1', sep=';')
```
✓ 0.0s

- Sử dụng 3 cột 6,7,8

```
columns = ['6', '7', '8']
```
✓ 0.0s

```
data = df[columns]
```
✓ 0.0s

Sử dụng HMM để phân tích chuỗi với 2 trạng thái ẩn

```
from hmmlearn.hmm import GaussianHMM
```
✓ 0.0s

- Lấy ra các quan sát từ dữ liệu: observations

```
observations = data.values
```
✓ 0.0s

- Định nghĩa mô hình HMM với 2 trạng thái ẩn

```
model = GaussianHMM(n_components=2, covariance_type="diag", n_iter=1000)
```
✓ 0.0s

```
model.fit(observations)
```
✓ 0.0s

```
GaussianHMM
GaussianHMM(n_components=2, n_iter=1000)
```

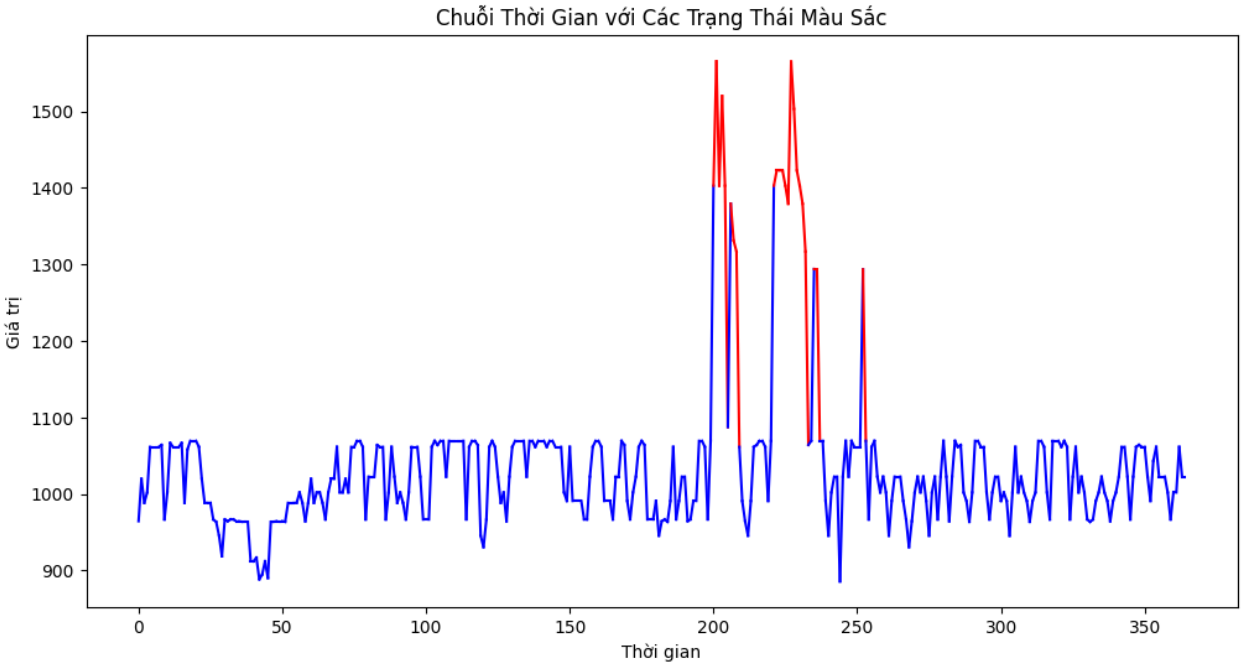
```
hidden_states = model.predict(observations)
```
✓ 0.0s

Vẽ biểu đồ đường

```
plt.figure(figsize=(12, 6))

# Vẽ biểu đồ đường với màu sắc dựa trên giá trị số nguyên
for i in range(len(data) - 1):
    plt.plot(data.index[i:i+2], data['6'][i:i+2], color='blue' if hidden_states[i] == 0 else 'red')

plt.xlabel('Thời gian')
plt.ylabel('Giá trị')
plt.title('Chuỗi Thời Gian với Các Trạng Thái Màu Sắc')
plt.show()
```
✓ 0.4s



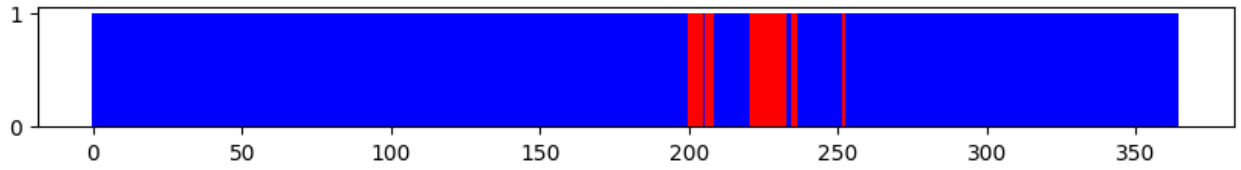
Vẽ biểu đồ hiển thị trạng thái của dữ liệu trong chuỗi thời gian, trong đó mỗi trạng thái được mã hóa bằng một màu sắc cụ thể.

```
state2color = {}
state2color['0'] = 'blue'
state2color['1'] = 'red'

def plot_price(samples, state2color):
    colors = [state2color[str(x)] for x in samples]
    x = np.arange(0, len(colors))
    y = np.ones(len(colors))
    plt.figure(figsize=(10,1))
    plt.bar(x, y, color=colors, width=1)

plot_price(hidden_states, state2color)
```

✓ 0.3s



Sử dụng HMM với 1 số điều chỉnh về transmat, start prob, emission prob

```python
transmat = np.array([[0.9, 0.1],
                     [0.7, 0.3]])

# Xác suất bắt đầu cho 2 trạng thái
start_prob = np.array([0.6, 0.4])

emission_probs = np.array([[0.4, 0.4, 0.1, 0.1],
                           [0.1, 0.1, 0.6, 0.2]])

# Tạo mô hình HMM với 2 trạng thái
model = GaussianHMM(n_components=2, covariance_type="diag", n_iter=1000)
model.startprob_ = start_prob
model.transmat_ = transmat
model.emissionprob_ = emission_probs
```

✓ 0.0s

Tối ưu hóa các tham số của nó sao cho phù hợp nhất với dữ liệu quan sát

```python
model.fit(observations)
```

✓ 0.0s

```
Even though the 'startprob_' attribute is set, it will be overwritten during initialization because 'init_params' contains 's'
Even though the 'transmat_' attribute is set, it will be overwritten during initialization because 'init_params' contains 't'
```

```
GaussianHMM
GaussianHMM(n_components=2, n_iter=1000)
```

Dự đoán trạng thái ẩn cho các quan sát được cung cấp bằng cách sử dụng mô hình HMM đã được huấn luyện

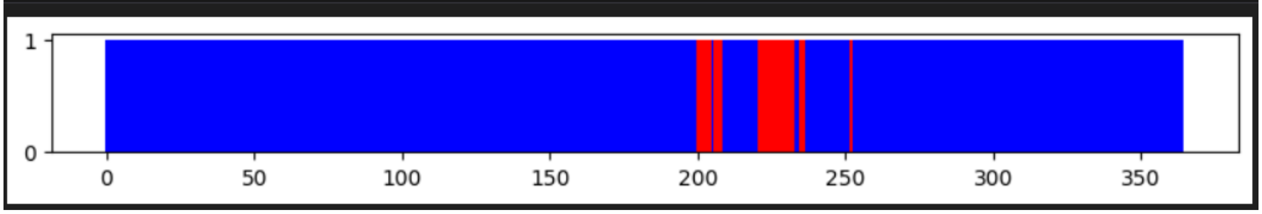
```python
hidden_states = model.predict(observations)
```
✓ 0.0s

```python
hidden_states
```
✓ 0.0s

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

Biểu đồ hiển thị trạng thái ẩn của dữ liệu trong mô hình HMM

```python
plot_price(hidden_states, state2color)
```
✓ 0.3s



Kalman Filter

tạo ra một bộ lọc Kalman với các tham số được chỉ định

```
from pykalman import KalmanFilter
```
✓ 0.0s

+ Code + Markdown

```
kf = KalmanFilter(initial_state_mean=np.mean(observations, axis=0), n_dim_obs=3)

state_means, state_covariances = kf.em(observations, n_iter=5).filter(observations)
```
✓ 0.5s

Thực hiện ước lượng và lọc dữ liệu bằng cách sử dụng thuật toán EM trong bộ lọc Kalman. Với:

observations là dữ liệu quan sát được sử dụng cho việc ước lượng và lọc.

n_iter=5 là số lần lặp của thuật toán EM.

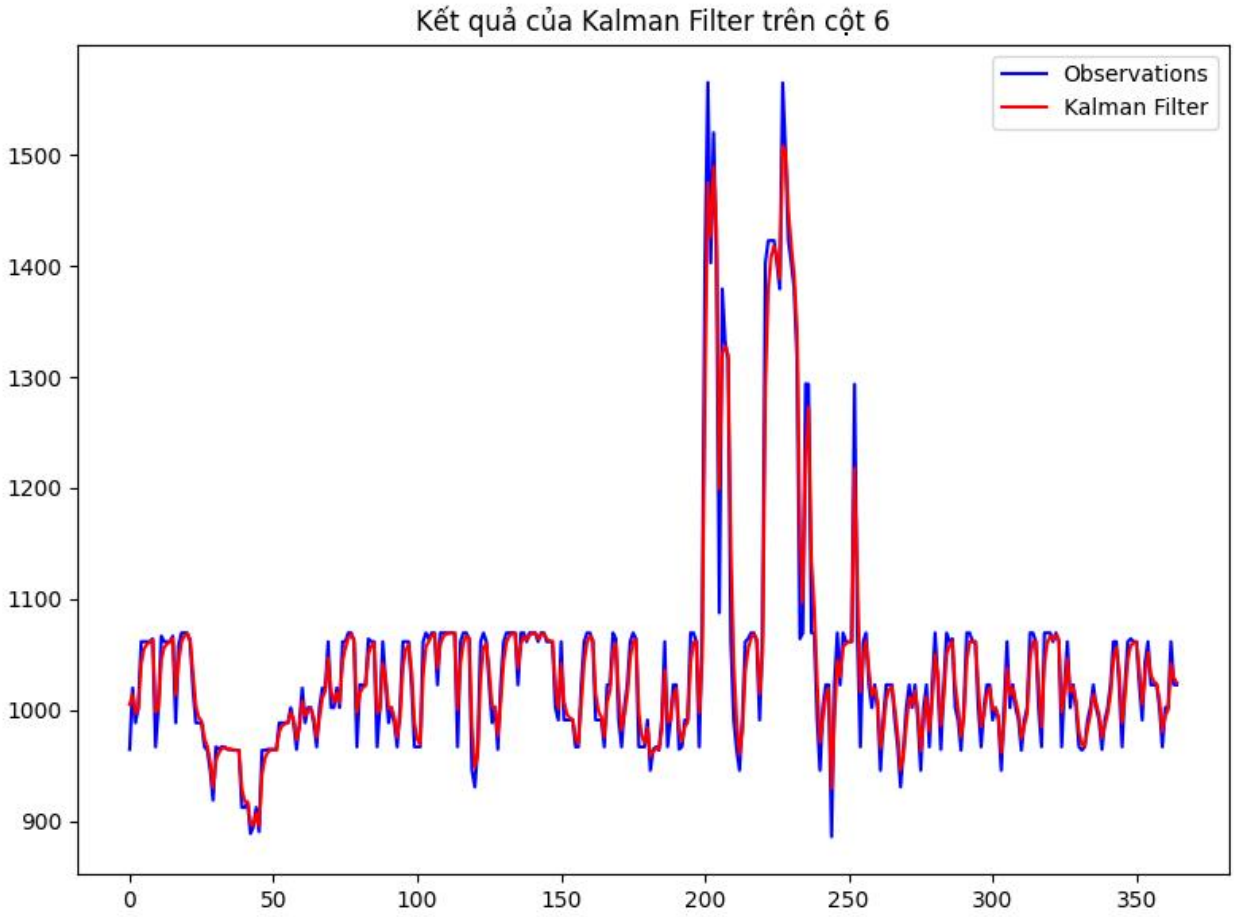
Sau khi hoàn thành, state_means chứa ước lượng trung bình của trạng thái tại mỗi thời điểm, và state_covariances chứa ma trận hiệp phương sai của trạng thái tại mỗi thời điểm.

Biểu đồ so sánh giữa dữ liệu quan sát và ước lượng của bộ lọc Kalman trên cột đầu tiên của dữ liệu

```
plt.figure(figsize=(8, 6))

plt.plot(range(observations.shape[0]), observations[:, 0], 'blue', label='Observations')
plt.plot(range(observations.shape[0]), state_means[:, 0], 'red', label='Kalman Filter')
plt.title(f'Kết quả của Kalman Filter trên cột 6')
plt.legend()

plt.tight_layout()
plt.show()
```
✓ 0.1s



Với: màu xanh: dữ liệu quan sát

màu đỏ: dữ liệu sau khi được lọc