1. Gaussian

- Ảnh train

```
# thử nghiệm với k=6
BGM = BayesianGaussianMixture(n_components=6,covariance_type='full',random_state=1,n_init=12)
# fit model and predict clusters
preds = BGM.fit_predict(X)
#Adding the Clusters feature to the orignal dataframe.
df["Clusters"]= preds
```

[8] ✓ 3.9s Python

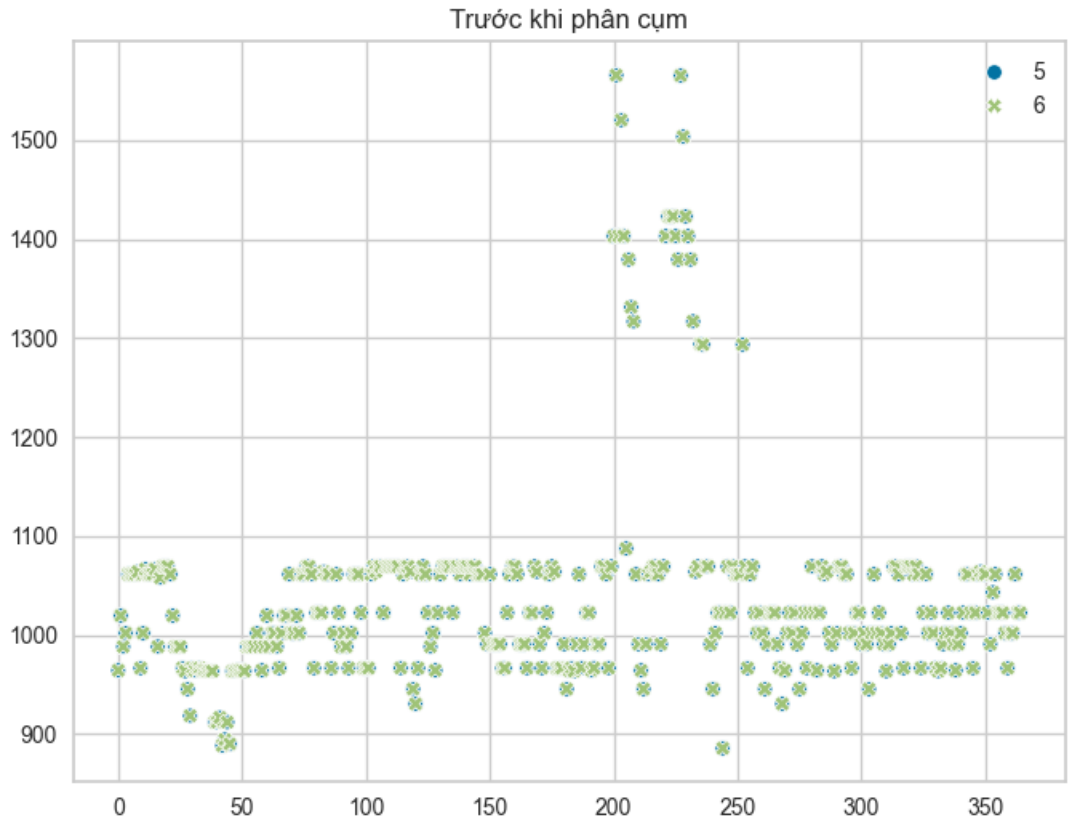
```
pp=BGM.predict_proba(X)
df_new=pd.DataFrame(X,columns=feats)
df_new[[f'predict_proba_{i}' for i in range(6)]]=pp
df_new['preds']=preds
df_new['predict_proba']=np.max(pp,axis=1)
df_new['predict']=np.argmax(pp,axis=1)

train_index=np.array([])
for n in range(6):
    n_inx=df_new[(df_new.preds==n) & (df_new.predict_proba > 0.68)].index
    train_index = np.concatenate((train_index, n_inx))
```

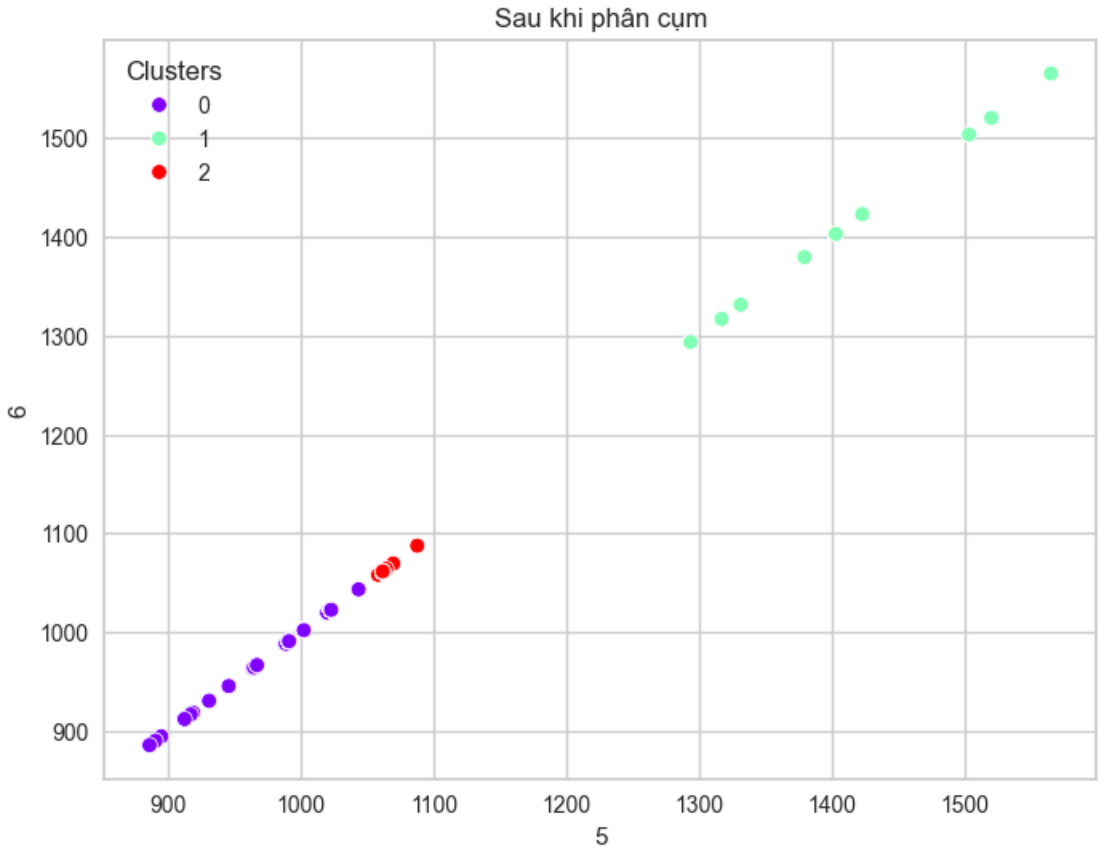
[9] ✓ 0.0s Python

- Ảnh kết quả

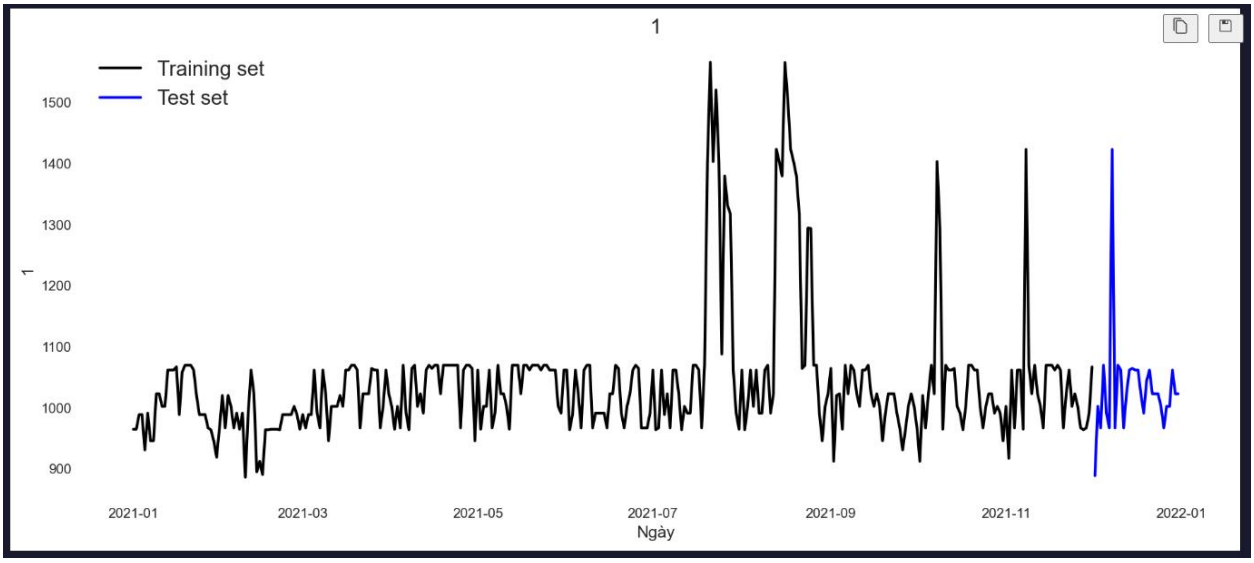
```
[200]   valid_0's multi_logloss: 0.000151726
[400]   valid_0's multi_logloss: 0.000151726
[200]   valid_0's multi_logloss: 0.000153686
[400]   valid_0's multi_logloss: 0.000153686
[200]   valid_0's multi_logloss: 0.000154034
[400]   valid_0's multi_logloss: 0.000154034
[200]   valid_0's multi_logloss: 0.000156754
[400]   valid_0's multi_logloss: 0.000156754
[200]   valid_0's multi_logloss: 0.000153261
[400]   valid_0's multi_logloss: 0.000153261
[200]   valid_0's multi_logloss: 0.00015524
[400]   valid_0's multi_logloss: 0.00015524
[200]   valid_0's multi_logloss: 0.000155472
[400]   valid_0's multi_logloss: 0.000155472
[200]   valid_0's multi_logloss: 9.92419e-06
[400]   valid_0's multi_logloss: 9.8222e-06
[600]   valid_0's multi_logloss: 9.75475e-06
[800]   valid_0's multi_logloss: 9.70667e-06
[1000]  valid_0's multi_logloss: 9.67072e-06
[1200]  valid_0's multi_logloss: 9.64285e-06
[1400]  valid_0's multi_logloss: 9.62062e-06
[1600]  valid_0's multi_logloss: 9.60248e-06
[1800]  valid_0's multi_logloss: 9.5874e-06
[2000]  valid_0's multi_logloss: 9.57468e-06
[2200]  valid_0's multi_logloss: 9.5638e-06
...
[400]   valid_0's multi_logloss: 0.000157332
[200]   valid_0's multi_logloss: 0.0119762
[400]   valid_0's multi_logloss: 0.0119762
[200]   valid_0's multi_logloss: 0.02382
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

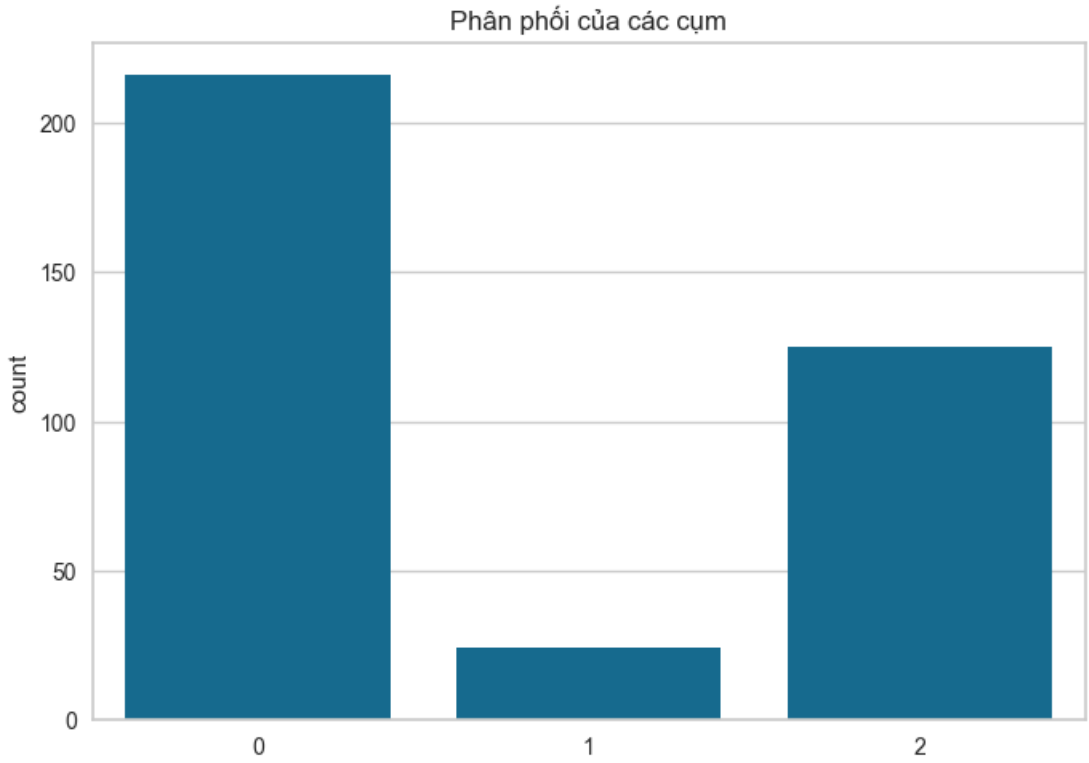
2. EDA

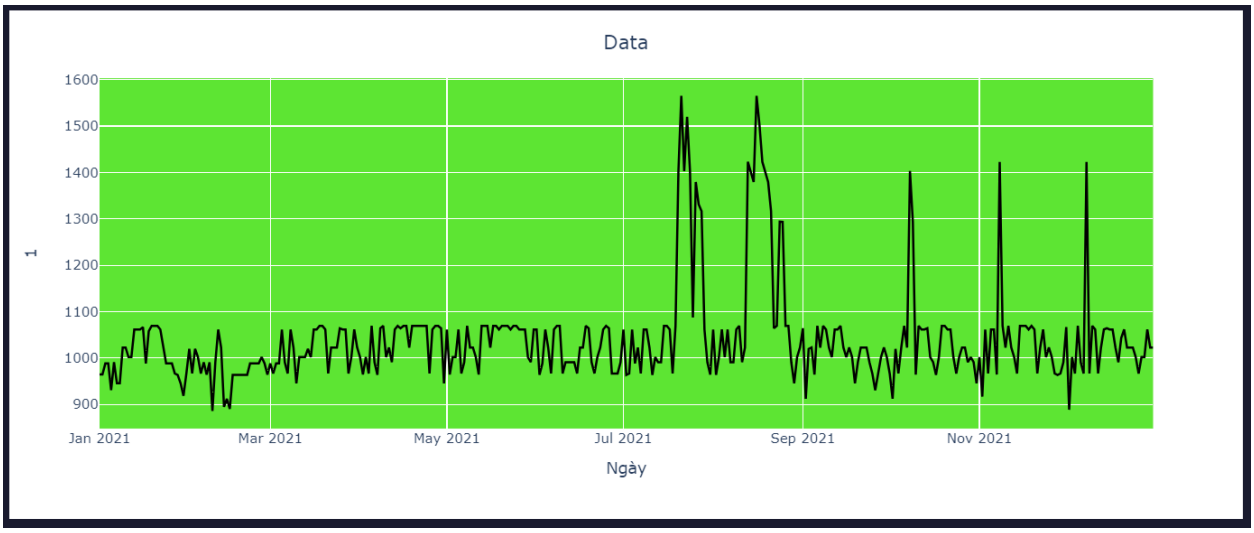

Trước khi phân cụm
5
6
1500
1400
1300
1200
1100
1000
900
0
50
100
150
200
250
300
350



- Biểu đồ thể hiện train và test trước khi dự đoán




Phân phối của các cụm
count
200
150
100
50
0
0
1
2

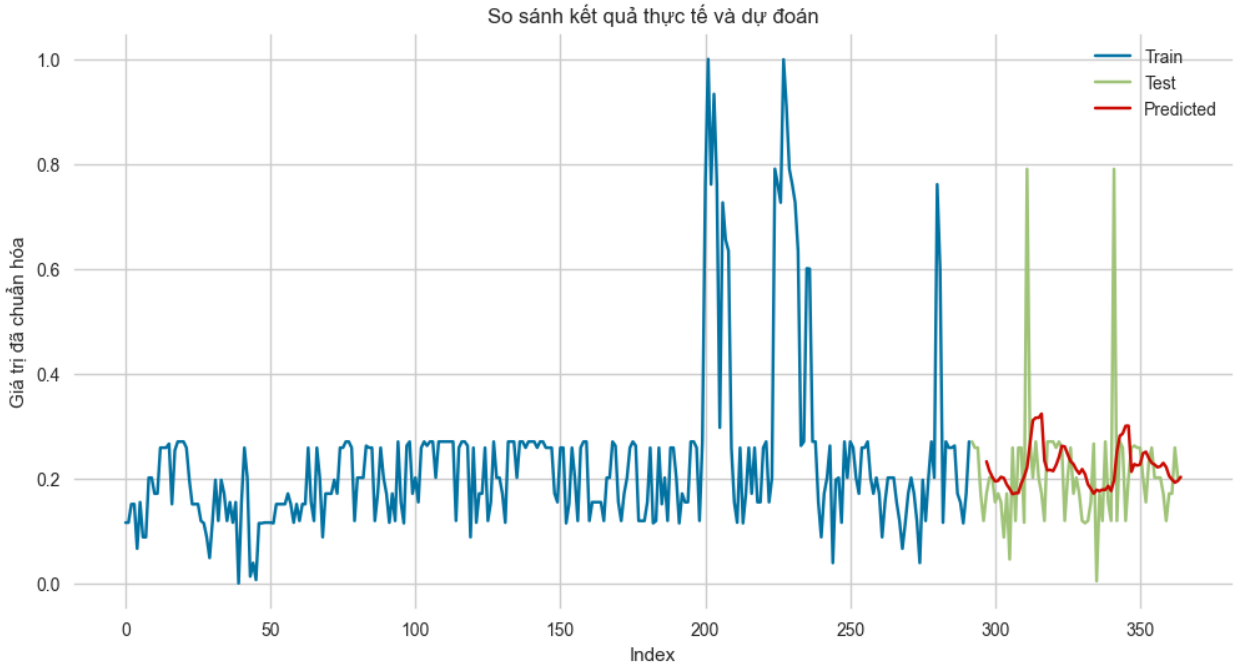

Data
1
1600
1500
1400
1300
1200
1100
1000
900
Jan 2021
Mar 2021
May 2021
Jul 2021
Sep 2021
Nov 2021
Ngày

3. LSTM

```python
# Chọn các cột cần thiết
feats = ['5', '6']
selected_data = df[feats]
print(selected_data.head(20))

# Sử dụng MinMaxScaler để chuẩn hóa các cột đã chọn về khoảng 0-1
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(selected_data)
scaled_df = pd.DataFrame(scaled_data, columns=feats)
print(scaled_df.head(20))

# Chuẩn bị dữ liệu cho LSTM
window_size = 5

def create_windowed_dataset(data, window_size):
    X, y = [], []
    for i in range(len(data) - window_size):
        X.append(data[i:i + window_size])
        y.append(data[i + window_size])
    return np.array(X), np.array(y)

# Chia dữ liệu thành train và test
split_ratio = 0.8
split_index = int(len(scaled_df) * split_ratio)
train_data = scaled_df.values[:split_index]
test_data = scaled_df.values[split_index:]

# Tạo dữ liệu train và test theo cửa sổ thời gian
X_train, y_train = create_windowed_dataset(train_data, window_size)
X_test, y_test = create_windowed_dataset(test_data, window_size)
```

```python
# Thay đổi hình dạng của dữ liệu để phù hợp với LSTM
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], X_train.shape[2]))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], X_test.shape[2]))

# Xây dựng mô hình LSTM
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(window_size, X_train.shape[2])))
model.add(LSTM(50))
model.add(Dense(X_train.shape[2]))

model.compile(optimizer='adam', loss='mean_squared_error')
model.summary()

# Huấn luyện mô hình
model.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.2)

# Dự đoán trên dữ liệu test
predictions = model.predict(X_test)

# Vẽ biểu đồ so sánh kết quả thực tế và dự đoán
plt.figure(figsize=(12, 6))
plt.plot(range(len(train_data)), train_data[:, 0], label='Train')
plt.plot(range(len(train_data), len(train_data) + len(test_data)), test_data[:, 0], label='Test')
plt.plot(range(len(train_data) + window_size, len(train_data) + window_size + len(predictions)), predictions[:, 0], label='Predicted')
plt.title('So sánh kết quả thực tế và dự đoán')
plt.xlabel('Index')
plt.ylabel('Giá trị đã chuẩn hóa')
plt.legend()
plt.show()
```

- Kết quả



Link github:[trinhdat24/ThucHanh2_PhanTichChuoiThoiGian (github.com)]