```python
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.mixture import BayesianGaussianMixture
import matplotlib.pyplot as plt
import seaborn as sns
```

[117] ✓ 0.0s Python

```python
df = pd.read_csv('D:\Ptich chuoi thoi gian\TH1\Gia SMP va SMPcap 2021.csv')
df.head(5)
```

[118] ✓ 0.0s Python

	Ngay	1	2	3	4	5	6	7	8	9	...	39	40	41	42	43	44	45	46	47	48
0	1/1/2021	964.4	964.4	964.4	964.4	964.4	964.4	964.4	964.4	964.4	...	964.4	964.4	964.4	964.4	964.4	964.4	964.4	964.4	964.4	964.4
1	1/2/2021	1019.7	1019.7	1019.7	1019.7	1019.7	1019.7	1019.7	1019.7	1019.7	...	1019.7	1019.7	1019.7	1019.7	1019.7	1019.7	1019.7	1019.7	1019.7	1019.7
2	1/3/2021	988.4	988.4	988.4	988.4	988.4	988.4	988.4	988.4	988.4	...	988.4	988.4	988.4	988.4	988.4	988.4	988.4	988.4	988.4	988.4
3	1/4/2021	1002.0	1002.0	1002.0	1002.0	1002.0	1002.0	1002.0	1002.0	1002.1	...	1010.8	1010.8	1010.8	1010.8	1010.8	1010.8	1010.8	1010.8	1010.8	1010.8
4	1/5/2021	1061.5	1061.5	1061.5	1061.5	1061.5	1061.5	1061.5	1061.5	1061.5	...	1061.5	1061.5	1061.5	1061.5	1061.5	1061.5	1061.5	1061.5	1061.5	1061.5

5 rows × 49 columns

```python
feats = ['9', '10']
```

✓ 0.0s Python

```python
df[feats].head()
```

✓ 0.0s Python

	9	10
0	964.4	964.4
1	1019.7	1019.7
2	988.4	988.4
3	1002.1	1002.1
4	1061.5	1061.5

1. EDA dữ liệu

1.1. Kiểm tra dữ liệu thiếu

```python
df[feats].isna().mean()
```

[121] ✓ 0.0s Python

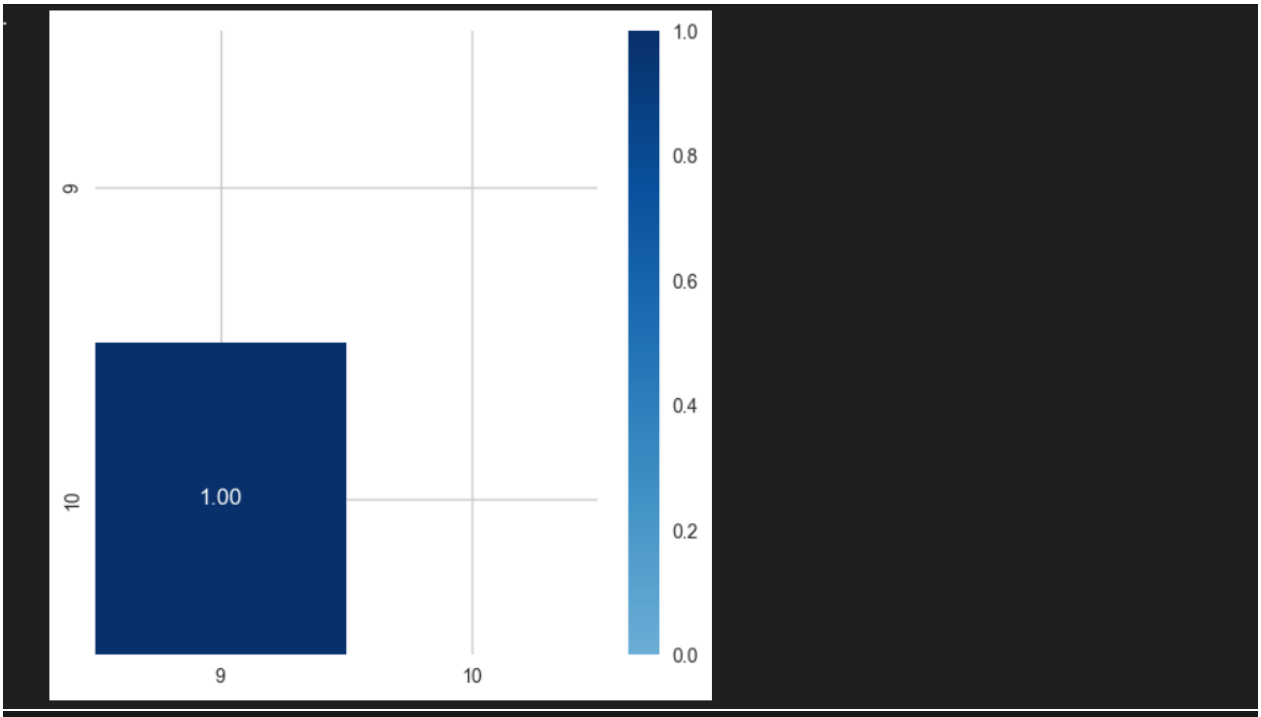
```
9     0.0
10    0.0
dtype: float64
```

- 2 cột đều không có dữ liệu thiếu

1.2. Sử dụng biểu đồ heatmap để kiểm tra độ tương quan của dữ liệu

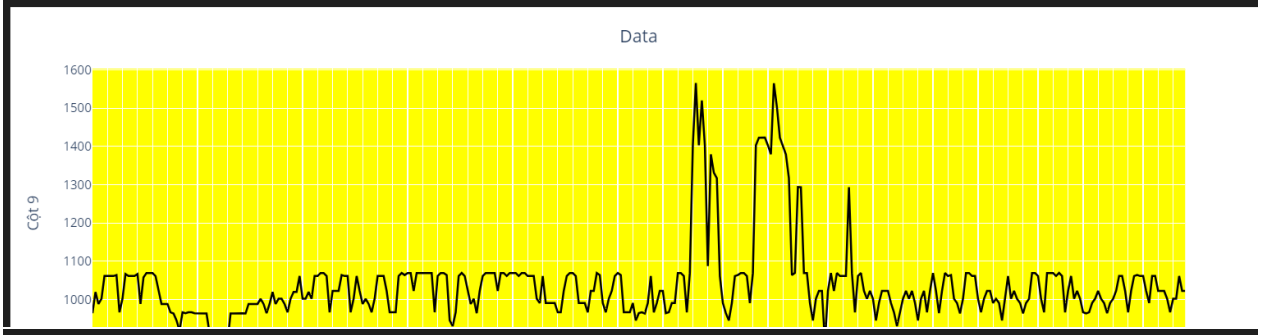
```python
mask = np.triu(np.ones_like(df[feats].corr(), dtype=bool))
cmap = sns.diverging_palette(230, 20, as_cmap=True)
ax = plt.subplots(figsize=(6,6))
sns.heatmap(df[feats].corr(), mask=mask, vmax=1, vmin=0, center=0,annot=True,fmt='.2f', cmap="Blues")
```

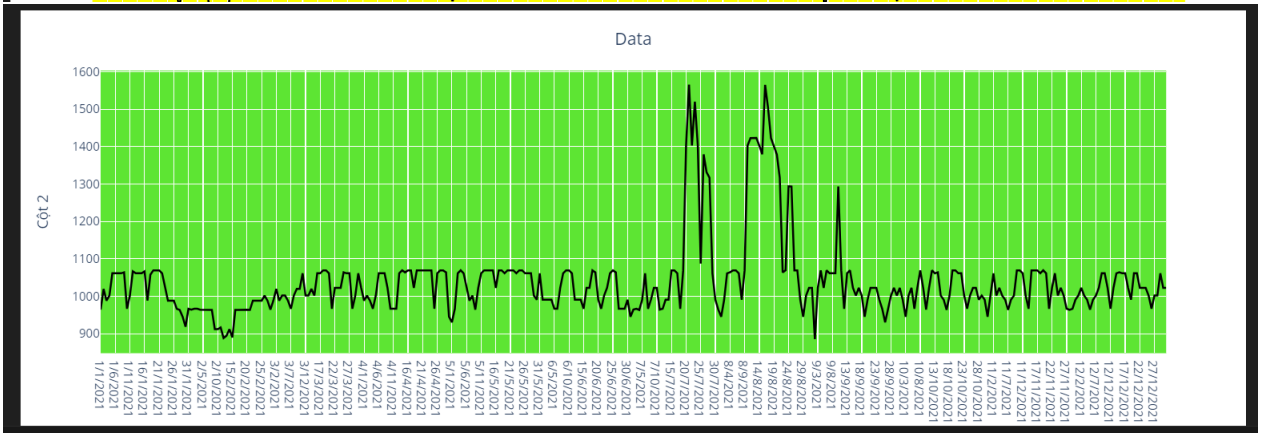
[122] ✓ 0.1s Python

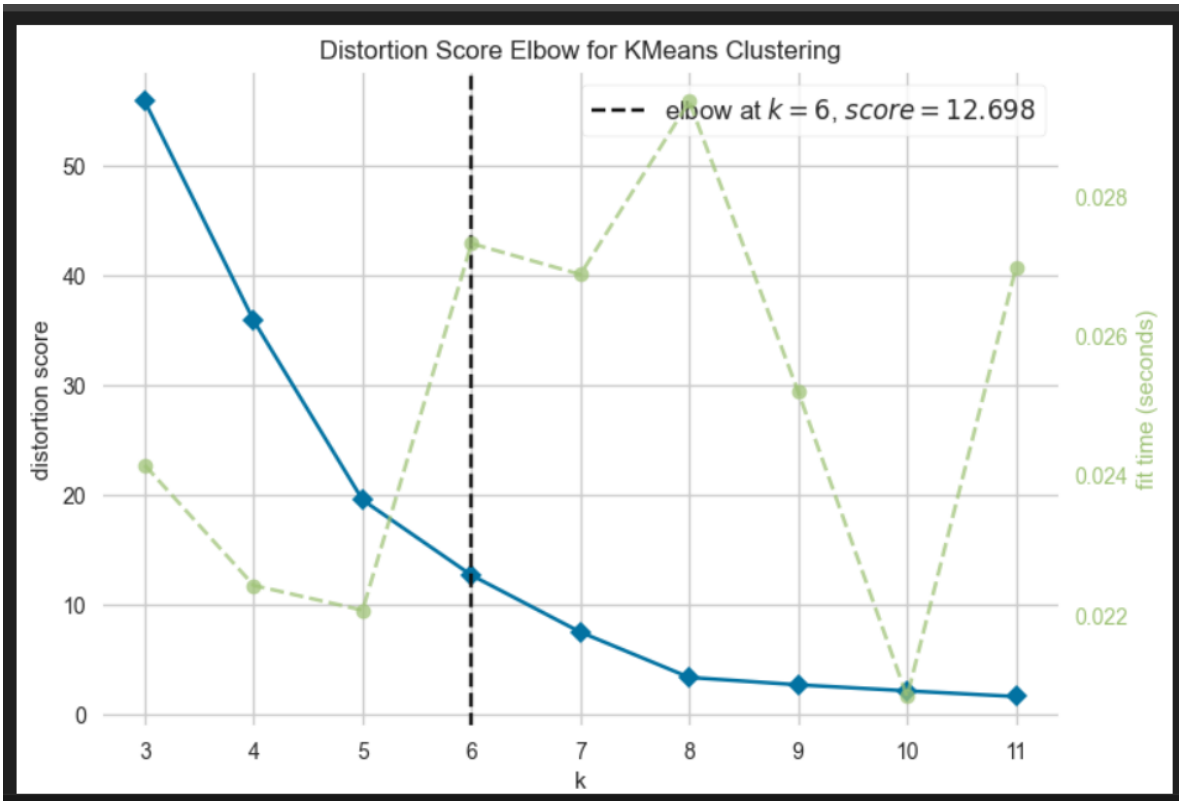



```
import plotly.express as px
fig = px.line(y=df['9'], x=df['Ngay'])
fig.update_traces(line_color='black')
fig.update_layout(xaxis_title="Ngay",
                  yaxis_title="Cột 9",
                  title={'text': "Data", 'y':0.95, 'x':0.5, 'xanchor':'center', 'yanchor':'top'},
                  plot_bgcolor='rgba(600,500,0,3.8)')
```

✓ 0.0s







2. Sử dụng Bayesian Gaussian Mixture

Thử nghiệm với K = 6

```
BGM = BayesianGaussianMixture(n_components=6,covariance_type='full',random_state=1,n_init=12)
preds = BGM.fit_predict(X)
df["Clusters"]= preds
```

✓ 0.7s

```
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 > 0.5)].index
    train_index = np.concatenate((train_index, n_inx))
```

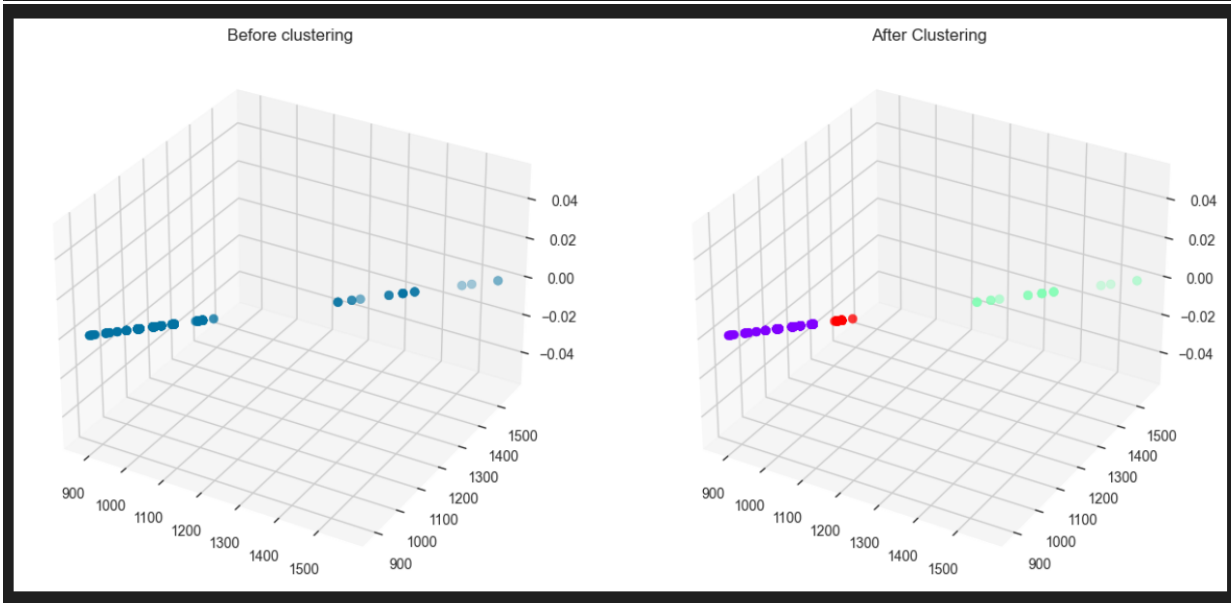
- Tạo bộ phân loại

```
from sklearn.model_selection import StratifiedKFold
import lightgbm as lgb
X_new=df_new.loc[train_index][feats]
y=df_new.loc[train_index]['preds']

params_lgb = {'learning_rate': 0.06,'objective': 'multiclass','boosting': 'gbdt','n_jobs': -1,'verbosity': -1, 'num_classes':7}

model_list=[]

gkf = StratifiedKFold(5)
for fold, (train_idx, valid_idx) in enumerate(gkf.split(X_new,y)):

    tr_dataset = lgb.Dataset(X_new.iloc[train_idx],y.iloc[train_idx],feature_name = feats)
    vl_dataset = lgb.Dataset(X_new.iloc[valid_idx],y.iloc[valid_idx],feature_name = feats)

    model = lgb.train(params = params_lgb,
                      train_set = tr_dataset,
                      valid_sets = vl_dataset,
                      num_boost_round = 5000,
                      callbacks=[ lgb.early_stopping(stopping_rounds=300, verbose=False), lgb.log_evaluation(period=200)])

    model_list.append(model)
```

- Có 3 nhãn tương ứng 3 cụm

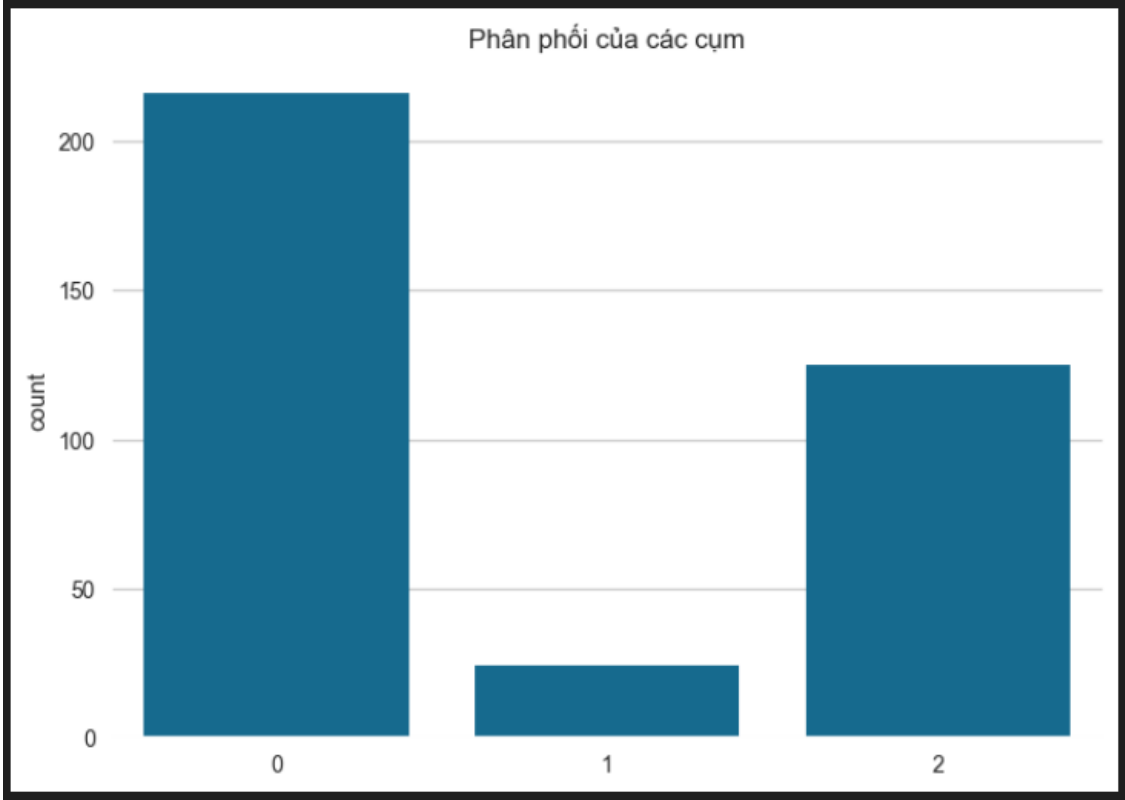
```
fig = plt.figure(figsize=(15,8))
ax = plt.subplot(1,2,1, projection='3d', label="bla")
ax.scatter(df['9'], df['10'], s=40, marker='o', cmap = 'rainbow' )
ax.set_title("Before clustering")
ax = plt.subplot(1,2,2, projection='3d', label="bla")
ax.scatter(df['9'], df['10'], s=40, c=df["Clusters"], marker='o',cmap="rainbow")
ax.set_title("After Clustering")
plt.show()
```

✓ 0.2s



```
pl = sns.countplot(x=np.argmax(lgb_preds,axis=1))
pl.set_title("Phân phối của các cụm")
plt.show()
```

✓ 0.0s



Thử nghiệm với k=4

```
BGM = BayesianGaussianMixture(n_components=4,covariance_type='full',random_state=1,n_init=12)
preds = BGM.fit_predict(X)
df["Clusters"]= preds
```

✓ 0.4s

```
pp=BGM.predict_proba(X)
df_new=pd.DataFrame(X,columns=feats)
df_new[[f'predict_proba_{i}' for i in range(4)]]=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(4):
    n_inx=df_new[(df_new.preds==n) & (df_new.predict_proba > 0.5)].index
    train_index = np.concatenate((train_index, n_inx))
```

✓ 0.0s

- Tạo bộ phân loại

```
from sklearn.model_selection import StratifiedKFold
import lightgbm as lgb
X_new=df_new.loc[train_index][feats]
y=df_new.loc[train_index]['preds']

params_lgb = {'learning_rate': 0.06,'objective': 'multiclass','boosting': 'gbdt','n_jobs': -1,'verbosity': -1, 'num_classes':7}

model_list=[]

gkf = StratifiedKFold(5)
for fold, (train_idx, valid_idx) in enumerate(gkf.split(X_new,y)):

    tr_dataset = lgb.Dataset(X_new.iloc[train_idx],y.iloc[train_idx],feature_name = feats)
    vl_dataset = lgb.Dataset(X_new.iloc[valid_idx],y.iloc[valid_idx],feature_name = feats)

    model = lgb.train(params = params_lgb,
                      train_set = tr_dataset,
                      valid_sets = vl_dataset,
                      num_boost_round = 5000,
                      callbacks=[ lgb.early_stopping(stopping_rounds=300, verbose=False), lgb.log_evaluation(period=200)])

    model_list.append(model)
```

✓ 1.5s

```
200]   valid_0's multi_logloss: 0.000178168
400]   valid_0's multi_logloss: 0.000178168
200]   valid_0's multi_logloss: 0.000223548
400]   valid_0's multi_logloss: 0.000223548
```

```
lgb_preds=0
for model in model_list:
    lgb_preds+=model.predict(df_new[feats])
labels=np.argmax(lgb_preds,axis=1)
```

✓ 0.0s

- Nhãn sau khi phân loại có 3 nhãn

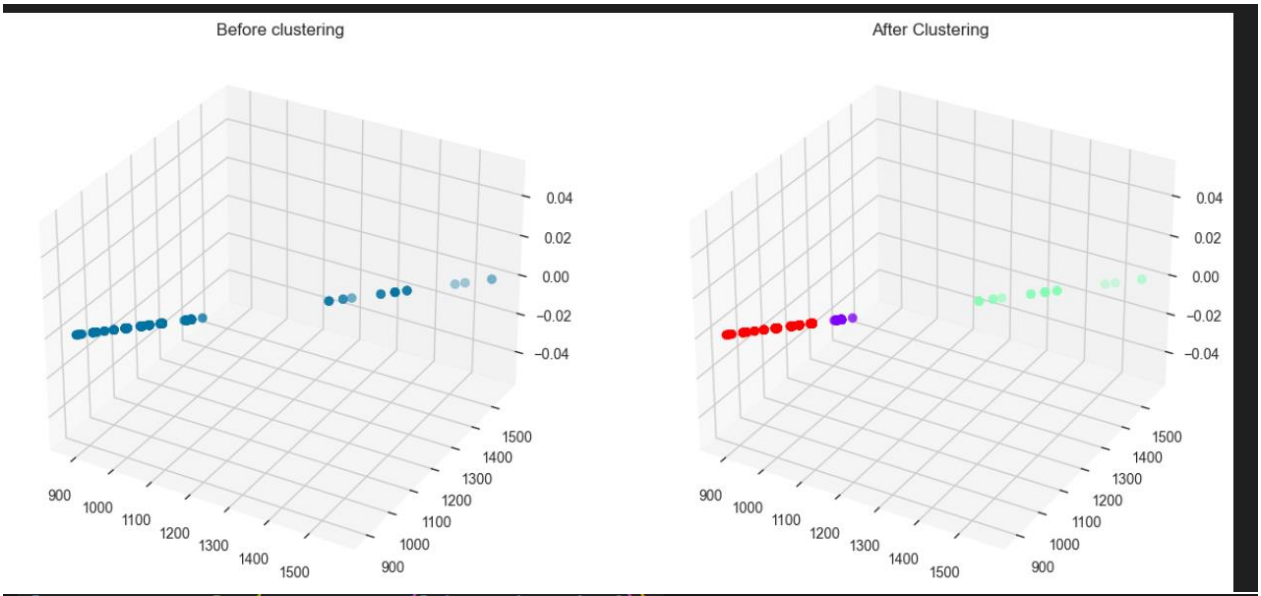
```
u = np.unique(labels)
u
```

✓ 0.0s

```
array([0, 1, 2], dtype=int64)
```

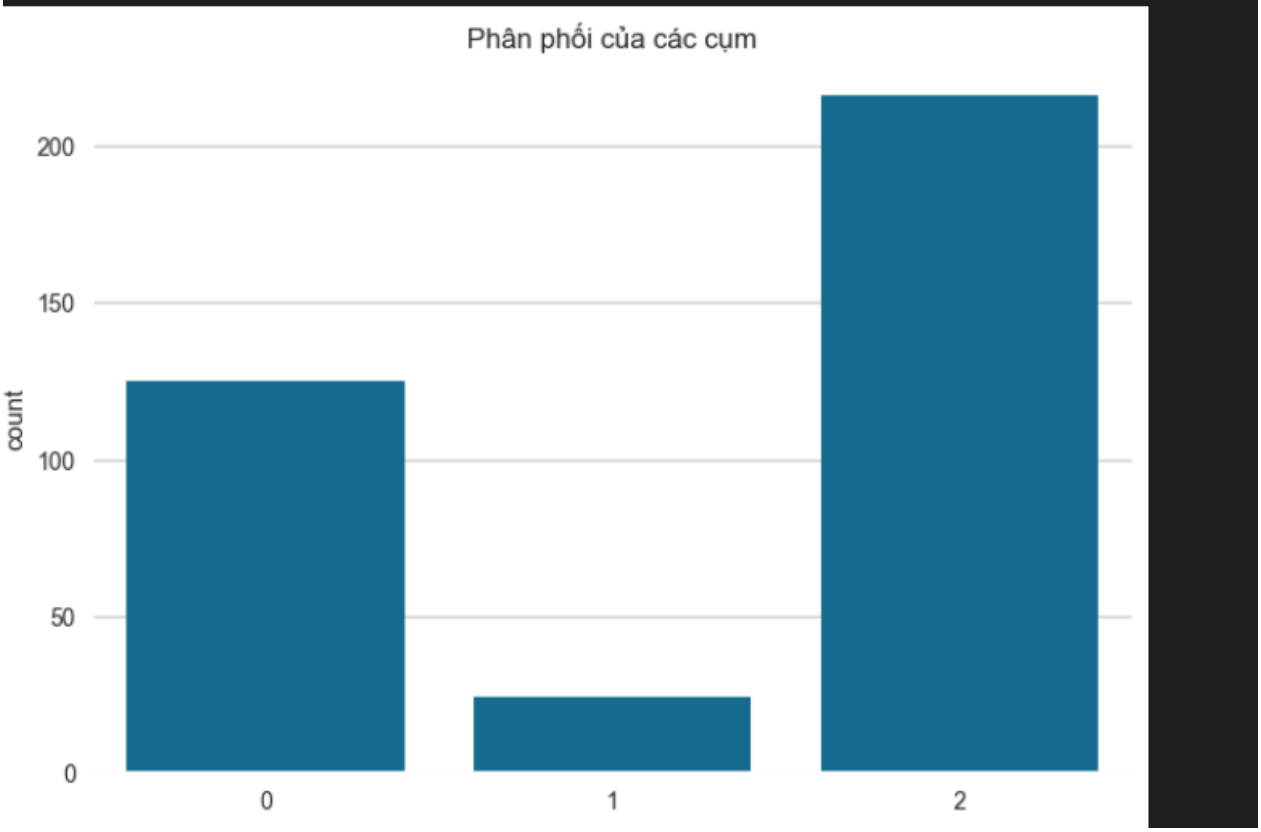
```
fig = plt.figure(figsize=(15,8))
ax = plt.subplot(1,2,1, projection='3d', label="bla")
ax.scatter(df['9'], df['10'], s=40, marker='o', cmap = 'rainbow' )
ax.set_title("Before clustering")
ax = plt.subplot(1,2,2, projection='3d', label="bla")
ax.scatter(df['9'], df['10'], s=40, c=df["Clusters"], marker='o',cmap="rainbow")
ax.set_title("After Clustering")
plt.show()
```

✓ 0.2s




```
pl = sns.countplot(x=np.argmax(lgb_preds,axis=1))
pl.set_title("Phân phối của các cụm")
plt.show()
```

0.0s



Thử nghiệm với K=3

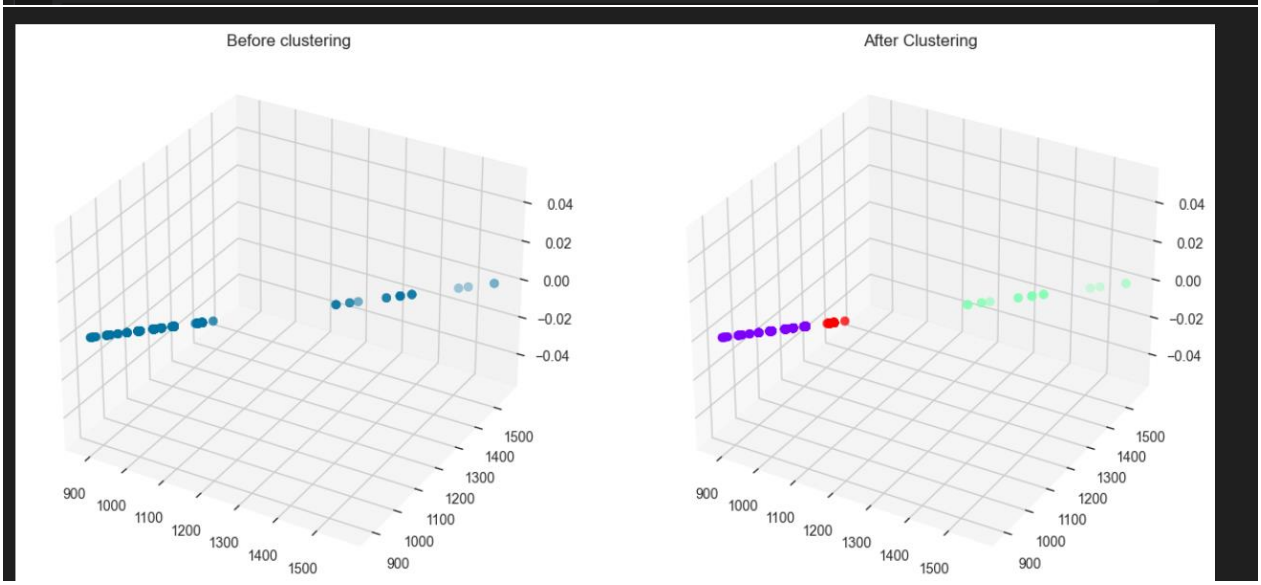
```
BGM = BayesianGaussianMixture(n_components=3,covariance_type='full',random_state=1,n_init=12)
preds = BGM.fit_predict(X)
df["Clusters"]= preds
```

✓ 0.1s

```
pp=BGM.predict_proba(X)
df_new=pd.DataFrame(X,columns=feats)
df_new[[f'predict_proba_{i}' for i in range(3)]]=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(3):
    n_inx=df_new[(df_new.preds==n) & (df_new.predict_proba > 0.5)].index
    train_index = np.concatenate((train_index, n_inx))
```

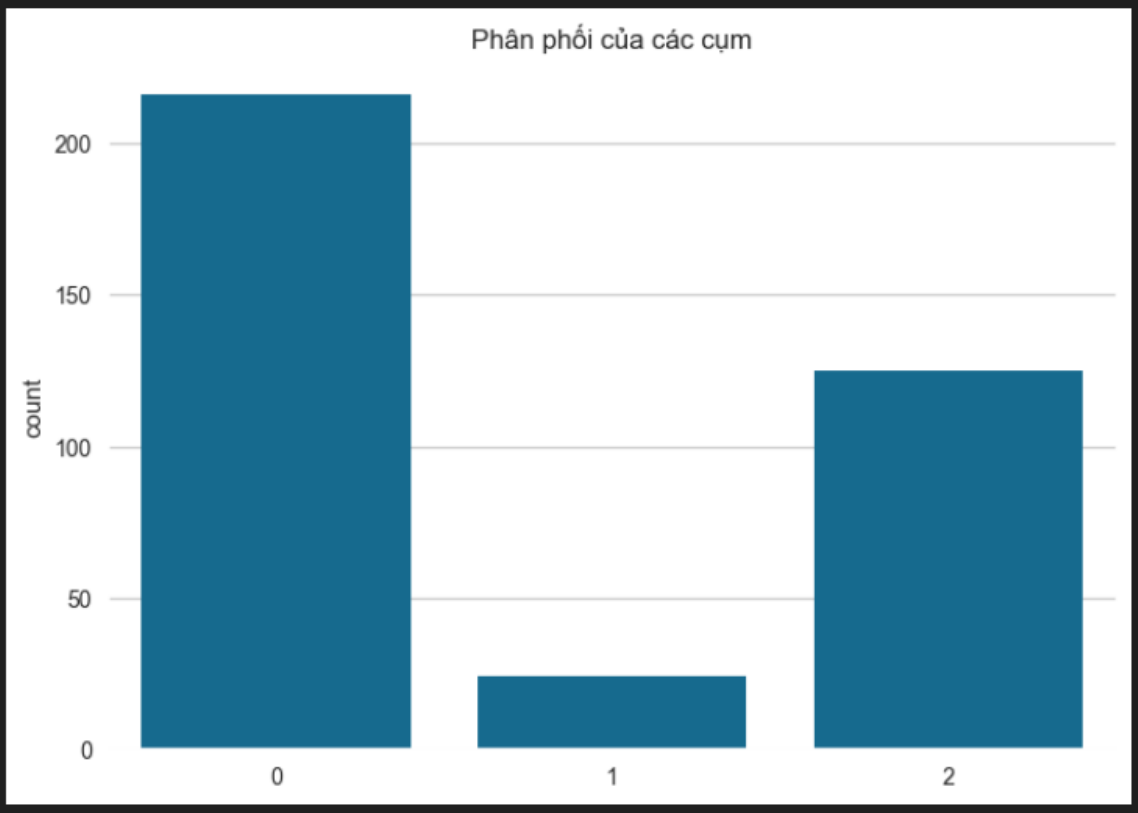
✓ 0.0s

```
from sklearn.model_selection import StratifiedKFold
import lightgbm as lgb
X_new=df_new.loc[train_index][feats]
y=df_new.loc[train_index]['preds']

params_lgb = {'learning_rate': 0.06,'objective': 'multiclass','boosting': 'gbdt','n_jobs': -1,'verbosity': -1, 'num_classes':7}
```



```
pl = sns.countplot(x=np.argmax(lgb_preds,axis=1))
pl.set_title("Phân phối của các cụm")
plt.show()
```

0.0s



3. Sử dụng GRU để dự đoán giá trị tương lai

```
import tensorflow as tf
from keras import Model
from keras.layers import Input, Dense, Dropout
from keras.layers import GRU
```

✓ 0.0s

```
feats = ['Ngay', '9', '10']
```

✓ 0.0s

```
df = df[feats]
```

✓ 0.0s

```
df
```

✓ 0.0s

	Ngay	9	10
0	1/1/2021	964.4	964.4
1	1/2/2021	1019.7	1019.7
2	1/3/2021	988.4	988.4
3	1/4/2021	1002.1	1002.1
4	1/5/2021	1061.5	1061.5
...	...	...	...
360	27/12/2021	1002.0	1002.0

- Chuyển format ngày

+ Code + Markdown

```
df['Ngay'] = pd.to_datetime(df['Ngay'], format='%d/%m/%Y')
df.sort_values(by='Ngay', ascending=True, inplace=True)
df.reset_index(drop=True, inplace=True)
```

✓ 0.0s

- Tạo số dữ liệu test bằng tháng 12

```
test_size = df[df['Ngay'].dt.month==12].shape[0]
test_size
```

✓ 0.0s

```
31
```

Biểu đồ thể hiện training và test trước khi dự đoán

- Biểu đồ thể hiện toàn bộ dữ liệu

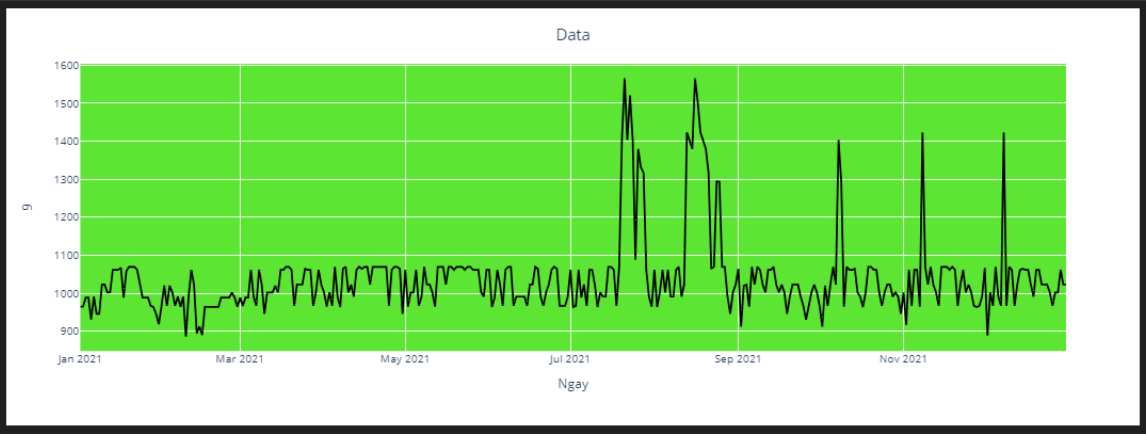
```
import plotly.express as px
fig = px.line(y=df['9'], x=df['Ngay'])
fig.update_traces(line_color='black')
fig.update_layout(xaxis_title="Ngay",
                  yaxis_title="9",
                  title={'text': "Data", 'y':0.95, 'x':0.5, 'xanchor':'center', 'yanchor':'top'},
                  plot_bgcolor='rgba(53,223,0,0.8)')
```

✓ 0.0s

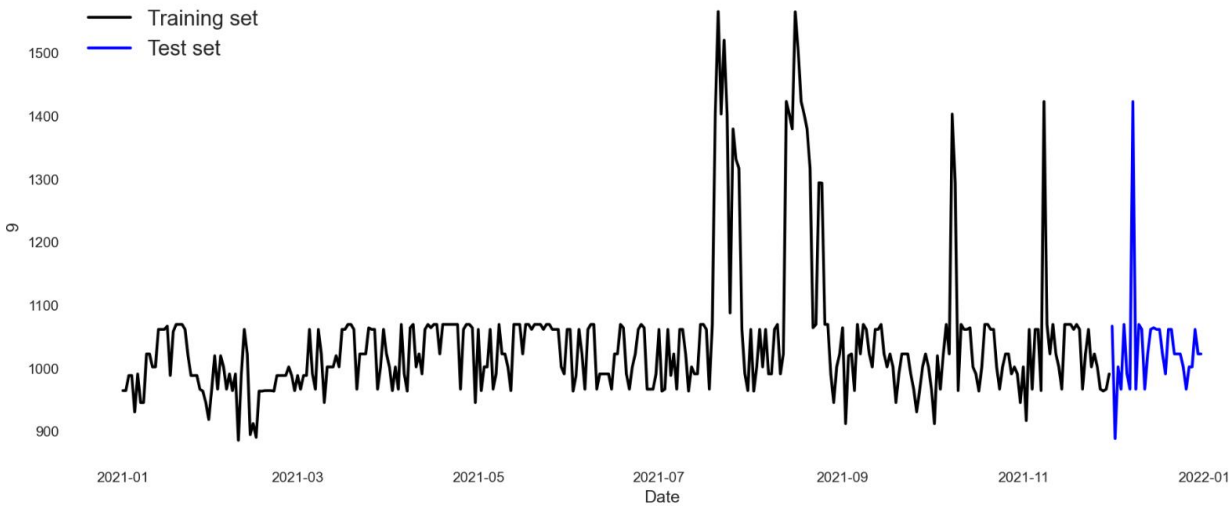
- Biểu đồ thể hiện toàn bộ dữ liệu

```
import plotly.express as px
fig = px.line(y=df['9'], x=df['Ngay'])
fig.update_traces(line_color='black')
fig.update_layout(xaxis_title="Ngay",
                  yaxis_title="9",
                  title={'text': "Data", 'y':0.95, 'x':0.5, 'xanchor':'center', 'yanchor':'top'},
                  plot_bgcolor='rgba(53,223,0,0.8)')
```

0.0s



9



- Sử dụng MinMaxScaler để đưa dữ liệu về khoảng 0-1

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(df[['9']])
```

✓ 0.0s

```
MinMaxScaler()
```

- Chọn window_size = 10 tức là 10 ngày dự đoán cho 1 ngày tiếp theo

```
window_size = 10
```

✓ 0.0s

- Tạo tập training data

```
train_data = df[['9']][:-test_size]
train_data = scaler.transform(train_data)
```

✓ 0.0s

```
X_train = []
y_train = []

for i in range(window_size, len(train_data)):
    X_train.append(train_data[i-window_size:i, 0])
    y_train.append(train_data[i, 0])
```

✓ 0.0s

- Chuyển đổi dữ liệu từ dataframe, series sang numpy array

```
X_train = np.array(X_train)
X_test  = np.array(X_test)
y_train = np.array(y_train)
y_test  = np.array(y_test)
```

✓ 0.0s

```
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test  = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
y_train = np.reshape(y_train, (-1,1))
y_test  = np.reshape(y_test, (-1,1))
```

✓ 0.0s

```
print('X_train Shape: ', X_train.shape)
print('y_train Shape: ', y_train.shape)
print('X_test Shape:  ', X_test.shape)
print('y_test Shape:  ', y_test.shape)
```

✓ 0.0s

- Khởi tạo model

```
def define_model():
    input1 = Input(shape=(window_size,1))
    x = GRU(units = 64, return_sequences=False)(input1)
    x = Dense(32, activation='softmax')(x)
    dnn_output = Dense(1)(x)

    model = Model(inputs=input1, outputs=[dnn_output])
    model.compile(loss='mean_squared_error', optimizer='Nadam')
    model.summary()

    return model
```

✓ 0.0s

- Tạo bộ siêu tham số: epochs=30 (lặp 30 lần)

```
model = define_model()
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.1, verbose=
```

✓ 6.0s

Model: "functional_1"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 10, 1)	0
gru (GRU)	(None, 64)	12,864
dense (Dense)	(None, 32)	2,080
dense_1 (Dense)	(None, 1)	33

Total params: 14,977 (58.50 KB)

```python
from sklearn.metrics import mean_absolute_percentage_error
MAPE = mean_absolute_percentage_error(y_test, y_pred)
```

✓ 0.0s

```python
print("Test Loss:", result)
print("Test MAPE:", MAPE)
```

✓ 0.0s

```
Test Loss: 0.018302038311958313
Test MAPE: 1.8511783593594395
```

```python
y_test_true = scaler.inverse_transform(y_test)
y_test_pred = scaler.inverse_transform(y_pred)
```

✓ 0.0s

- Vẽ biểu đồ sau khi đã tự đoán

```python
plt.figure(figsize=(10, 8), dpi=150)
plt.rcParams['axes.facecolor'] = 'white'
plt.rc('axes',edgecolor='white')
plt.plot(df['Ngay'].iloc[:-test_size], scaler.inverse_transform(train_data), color='black', lw=2)
plt.plot(df['Ngay'].iloc[-test_size:], y_test_true, color='blue', lw=2)
plt.plot(df['Ngay'].iloc[-test_size:], y_test_pred, color='red', lw=2)
plt.title('Prediction', fontsize=15)
plt.xlabel('Date', fontsize=12)
plt.ylabel('9', fontsize=12)
plt.legend(['Training', 'Data thực tế', 'Data dự đoán'], loc='upper left', prop={'size': 15})
plt.grid(color='white')
plt.show()
```

✓ 0.2s