1.Chia kịch bản data

- Dữ liệu được đọc từ tệp 'Gia SMP va SMPcap 2021(Giá thị trường SMP).csv'.

- Sau đó, lấy cột 'Ngày' và cột 2 để phân tích.

- Áp dụng PowerTransformer để biến đổi dữ liệu.

- Mô hình Bayesian Gaussian Mixture (BGM) được sử dụng để phân cụm dữ liệu.

- Chia dữ liệu thành các tập huấn luyện và sử dụng StratifiedKFold, một kỹ thuật chia dữ liệu cross-validation dựa trên một biến mục tiêu phân loại.

- Một mô hình Gradient Boosting được huấn luyện trên dữ liệu và sử dụng để dự đoán nhãn của dữ liệu kiểm tra.

- Chuỗi lợi nhuận được sử dụng để kiểm tra tính stationality của dữ liệu bằng kiểm định Augmented Dickey-Fuller (ADF).

- Cuối cùng, mô hình ARIMA (Autoregressive Integrated Moving Average) được khởi tạo, phù hợp và đánh giá trên dữ liệu lợi nhuận để dự đoán xu hướng tương lai của giá cổ phiếu.

2.Ảnh training , kết quả (đầy đủ code)

```
Go  Run  Terminal  Help    PhanTichChuoiThoiGian
trinhdat.ipynb ●   Gia SMP va SMPcap 2021(Giá thị trường SMP).csv
ThucHanh1 > trinhdat.ipynb > ...
+ Code  + Markdown | Run All  Restart  Clear All Outputs | Variables  Outline  ...      Python 3.12.3

#Importing the Libraries
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
from matplotlib import colors
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import RobustScaler,PowerTransformer
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn import metrics
import lightgbm as lgb
import warnings
import sys
if not sys.warnoptions:
    warnings.simplefilter("ignore")
[76]  ✓ 0.0s                                                      Python
```

```
df = pd.read_csv("../ThucHanh1/Gia SMP va SMPcap 2021(Giá thị trường SMP).csv",encoding='ISO-8859-1',sep=';')
dfs = pd.concat([df["Ngày"], df["2"]], axis=1)
print(dfs)
[77]  ✓ 0.0s                                                      Python

...
           Ngày       2
0    01/01/2021   964.4
1    01/02/2021  1019.7
2    01/03/2021   988.4
3    01/04/2021  1002.0
4    01/05/2021  1061.5
..          ...     ...
360  27/12/2021  1002.0
361  28/12/2021  1002.0
362  29/12/2021  1061.5
363  30/12/2021  1022.6
364  31/12/2021  1022.6

[365 rows x 2 columns]
```

```python
feats= ['2']
```
[78] ✓ 0.0s Python

```python
from sklearn.preprocessing import PowerTransformer

# Chuyển đổi cột '2' thành mảng 2D
X = df['2'].values.reshape(-1, 1)

# Áp dụng PowerTransformer
transformer = PowerTransformer()
X_transformed = transformer.fit_transform(X)
```
[79] ✓ 0.0s Python

```python
BGM = BayesianGaussianMixture(n_components=7,covariance_type='full',random_state=1,n_init=15)
#fit model and predict clusters
preds = BGM.fit_predict(X)

#Adding the Clusters feature to the orignal dataframe.
df["Clusters"]= preds
```
[80] ✓ 0.7s Python

```python
pp=BGM.predict_proba(X)# Calcualting the probabilities of each prediction
df_new=pd.DataFrame(X,columns=feats)
df_new[[f'predict_proba_{i}' for i in range(7)]]=pp # creating new dataframe columns of probabilites
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(7):
    n_inx=df_new[(df_new.preds==n) & (df_new.predict_proba > 0.68)].index
    train_index = np.concatenate((train_index, n_inx))
```
[81] ✓ 0.0s Python

```python
from sklearn.model_selection import StratifiedKFold
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(2)
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)
```
[82] ✓ 1.8s Python

```
[200]	valid_0's multi_logloss: 0.0116669
[200]	valid_0's multi_logloss: 0.0107932
[400]	valid_0's multi_logloss: 0.0107932
[600]	valid_0's multi_logloss: 0.0107932
[800]	valid_0's multi_logloss: 0.0107932
[1000]	valid_0's multi_logloss: 0.0107931
[1200]	valid_0's multi_logloss: 0.0107931
[1400]	valid_0's multi_logloss: 0.0107931
[1600]	valid_0's multi_logloss: 0.0107931
[1800]	valid_0's multi_logloss: 0.0107931
[2000]	valid_0's multi_logloss: 0.0107931
[2200]	valid_0's multi_logloss: 0.010793
[2400]	valid_0's multi_logloss: 0.010793
[2600]	valid_0's multi_logloss: 0.010793
[2800]	valid_0's multi_logloss: 0.010793
[3000]	valid_0's multi_logloss: 0.010793
[3200]	valid_0's multi_logloss: 0.010793
[3400]	valid_0's multi_logloss: 0.010793
[3600]	valid_0's multi_logloss: 0.010793
[3800]	valid_0's multi_logloss: 0.010793
[4000]	valid_0's multi_logloss: 0.010793
[4200]	valid_0's multi_logloss: 0.010793
[4400]	valid_0's multi_logloss: 0.010793
[4600]	valid_0's multi_logloss: 0.010793
[4800]	valid_0's multi_logloss: 0.010793
[5000]	valid_0's multi_logloss: 0.010793
```

```python
lgb_preds=0
for model in model_list:
    lgb_preds+=model.predict(df_new[feats])
```

[83] ✓ 0.1s Python

```python
labels=np.argmax(lgb_preds,axis=1)
```

[84] ✓ 0.0s Python

```python
import numpy as np
# Tính chuỗi return
r_t = np.log(dfs['2'] / dfs['2'].shift(1)).values
```

[85] ✓ 0.0s Python

```python
mean = np.nanmean(r_t)
r_t[0]=mean
r_t[:5]
```

[86] ✓ 0.0s Python

```
array([ 0.00016098,  0.0557576 , -0.03117627,  0.01366581,  0.057685  ])
```

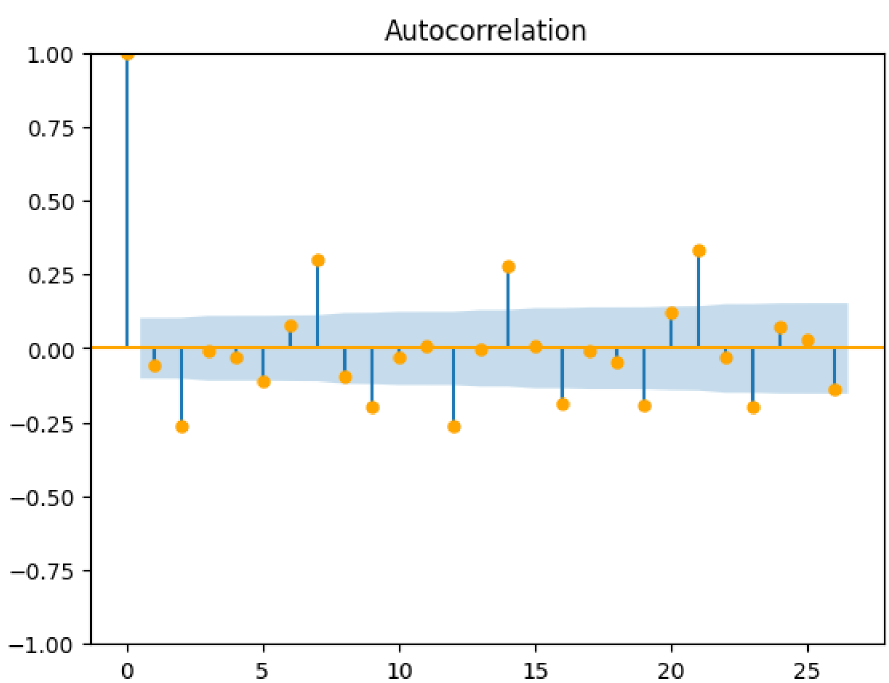
```python
from statsmodels.tsa.stattools import adfuller
result = adfuller(r_t)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
```

[87] ✓ 0.0s Python

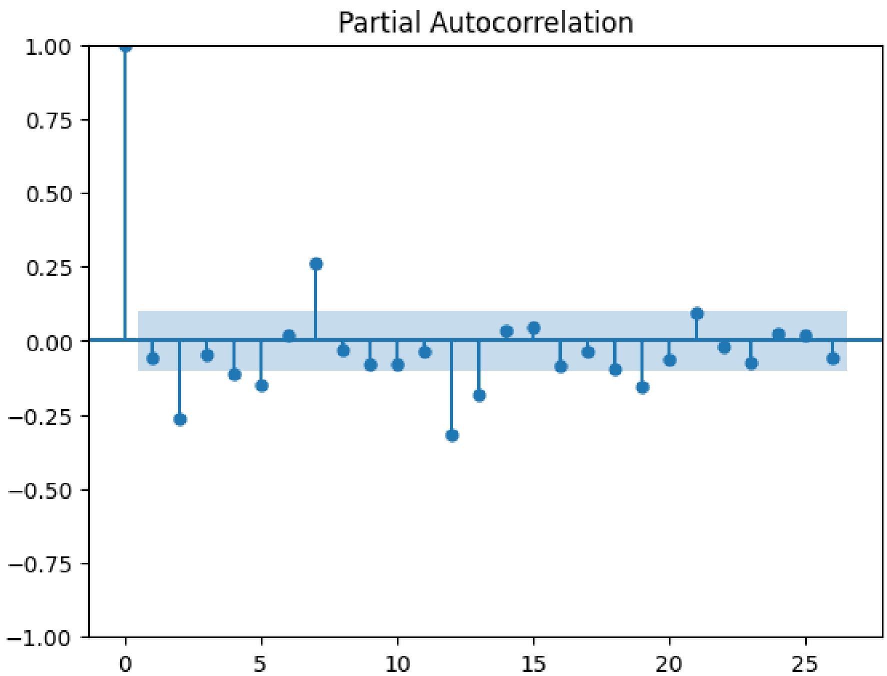
```
ADF Statistic: -9.347967
p-value: 0.000000
Critical Values:
	1%: -3.449
	5%: -2.870
	10%: -2.571
```

```python
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import matplotlib.pyplot as plt
plt.figure(figsize = (8, 6))
ax1 = plot_acf(r_t,color='orange')
```

[88] ✓ 0.2s Python



```python
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import matplotlib.pyplot as plt
plt.figure(figsize = (8, 6))
ax2 = plot_pacf(r_t)
```
[89] ✓ 0.1s Python




```python
from statsmodels.tsa.arima.model import ARIMA

# Khởi tạo và phù hợp với mô hình ARIMA
model_arima = ARIMA(r_t, order=(2, 0, 2))
model_fit = model_arima.fit()

# In ra tóm tắt của mô hình
print(model_fit.summary())
```
[90] ✓ 0.5s Python

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  365
Model:                 ARIMA(2, 0, 2)   Log Likelihood                 564.481
Date:                Tue, 07 May 2024   AIC                          -1116.962
Time:                        11:05:28   BIC                          -1093.563
Sample:                             0   HQIC                         -1107.663
                                - 365
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const       4.927e-05      0.001      0.075      0.940      -0.001       0.001
ar.L1          0.0038      0.068      0.056      0.955      -0.130       0.137
ar.L2          0.5784      0.062      9.279      0.000       0.456       0.701
ma.L1         -0.0795      0.053     -1.492      0.136      -0.184       0.025
ma.L2         -0.8394      0.056    -14.971      0.000      -0.949      -0.729
sigma2         0.0027      0.000     25.368      0.000       0.002       0.003
===================================================================================
Ljung-Box (L1) (Q):                   0.60   Jarque-Bera (JB):               653.37
Prob(Q):                              0.44   Prob(JB):                         0.00
Heteroskedasticity (H):               1.75   Skew:                             0.55
Prob(H) (two-sided):                  0.00   Kurtosis:                         9.46
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

Link github : https://github.com/trinhdat24/ThucHanh1_PhanTichChuoiThoiGian