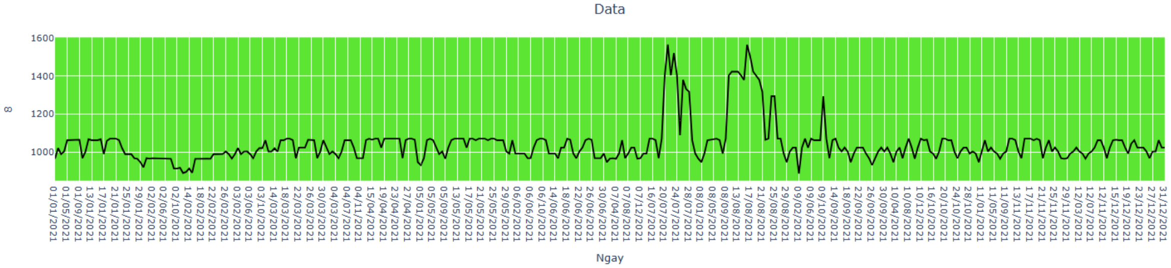
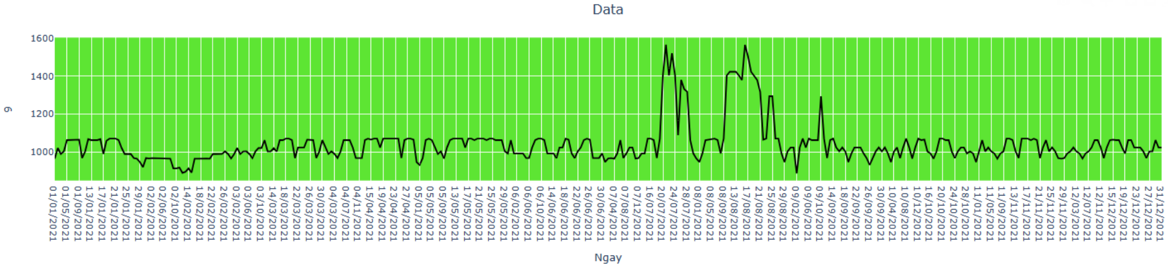
```
[83]: import plotly.express as px
      fig = px.line(y=df['8'], x=df['Ngay'])
      fig.update_traces(line_color='black')
      fig.update_layout(xaxis_title="Ngay",
                        yaxis_title="8",
                        title={'text': "Data", 'y':0.95, 'x':0.5, 'xanchor':'center', 'yanchor':'top'},
                        plot_bgcolor='rgba(53,223,0,0.8)')
```



Ngay

```
[84]: 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)')
```



```
[85]: transformer = StandardScaler()
      X = transformer.fit_transform(np.array(df[feats]))
```

```
[86]: X[:5]
```

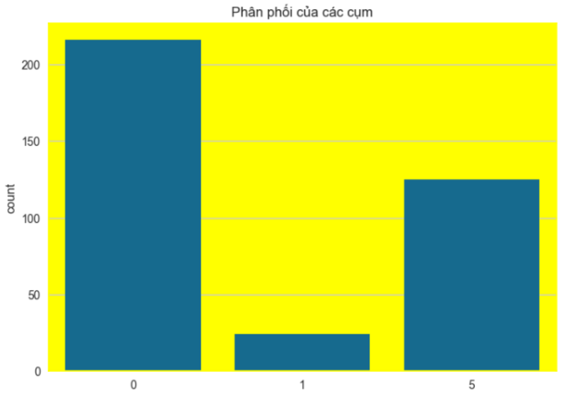
```
[91]: lgb_preds=0
      for model in model_list:
          lgb_preds+=model.predict(df_new[feats])
```

```
[92]: labels=np.argmax(lgb_preds,axis=1)
```

```
[93]: u = np.unique(labels)
      u
```

```
[93]: array([0, 1, 5], dtype=int64)
```

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



[98]:

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

[99]:

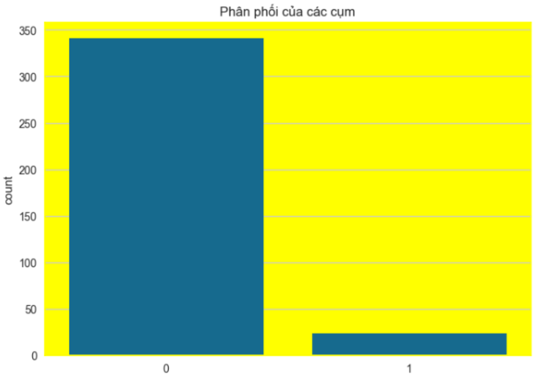
```
u = np.unique(labels)
u
```

[99]:

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

[100]:

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



[101]:

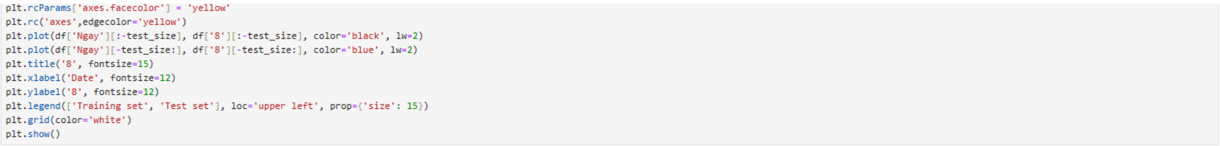
```
BGM = BayesianGaussianMixture(n_components=3,covariance_type='full',random_state=1,n_init=12)
preds = BGM.fit_predict(X)
df["Clusters"]= preds
```

[102]:

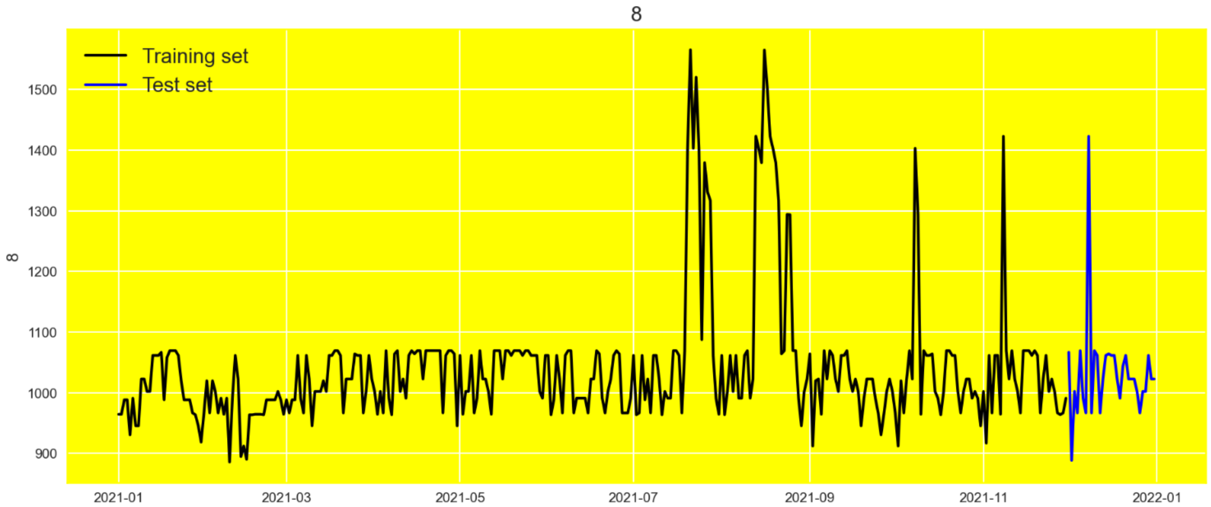
```
pp=BGM.predict_proba(X)
df_new=pd.DataFrame(X,columns=feats)
df_new[[f'predict_proba_{i}' for i in range(3)]]=pp
df_new['preds']=preds
df_new['predict_proba']=np.max(pp,axis=1)
df_new['predict']=np.argmax(pp,axis=1)

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))
```

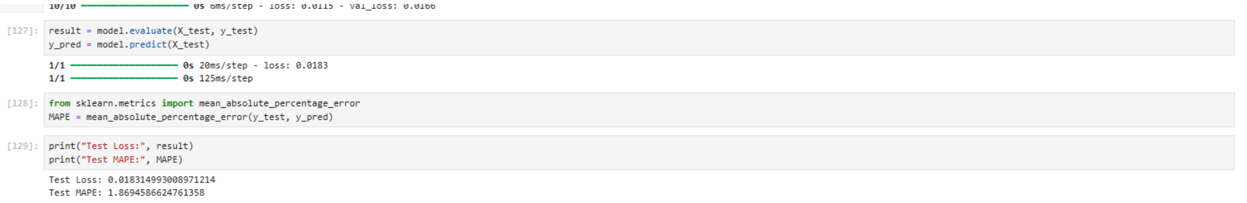
[103]:

```
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)
```



```
plt.rcParams['axes.facecolor'] = 'yellow'
plt.rc('axes',edgecolor='yellow')
plt.plot(df['Ngay'][:-test_size], df['8'][:-test_size], color='black', lw=2)
plt.plot(df['Ngay'][-test_size:], df['8'][-test_size:], color='blue', lw=2)
plt.title('8', fontsize=15)
plt.xlabel('Date', fontsize=12)
plt.ylabel('8', fontsize=12)
plt.legend(['Training set', 'Test set'], loc='upper left', prop={'size': 15})
plt.grid(color='white')
plt.show()
```





```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0115 - val_loss: 0.0166
```

```
[127]: result = model.evaluate(X_test, y_test)
       y_pred = model.predict(X_test)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - loss: 0.0183
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 125ms/step
```

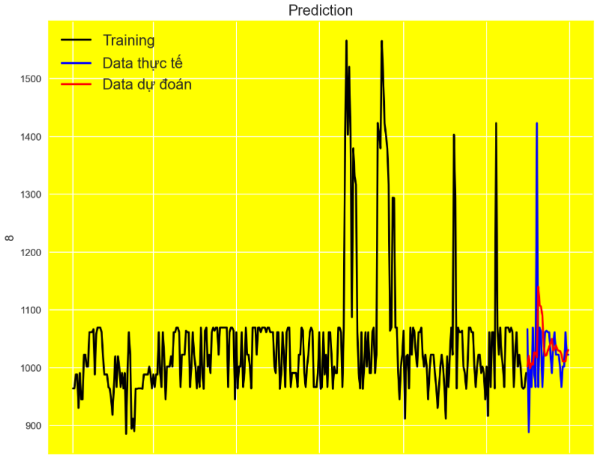
```
[128]: from sklearn.metrics import mean_absolute_percentage_error
       MAPE = mean_absolute_percentage_error(y_test, y_pred)
```

```
[129]: print("Test Loss:", result)
       print("Test MAPE:", MAPE)
```

```
Test Loss: 0.018314993008971214
Test MAPE: 1.8694586624761358
```

[131]:

```
plt.figure(figsize=(10, 8), dpi=150)
plt.rcParams['axes.facecolor'] = 'yellow'
plt.rc('axes',edgecolor='yellow')
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('8', fontsize=12)
plt.legend(['Training', 'Data thực tế', 'Data dự đoán'], loc='upper left', prop={'size': 15})
plt.grid(color='white')
plt.show()
```



[125]:

```
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
```

[126]:

```
model = define_model()
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.1, verbose=1)
```

Model: "functional_3"

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 10, 1)	0
gru_1 (GRU)	(None, 64)	12,864
dense_2 (Dense)	(None, 32)	2,080
dense_3 (Dense)	(None, 1)	33