PROJECT : MACHINE LEARNING IN PREDICTING FINANCIAL CRISIS

Contents

1. DataSet

1.1 Nguồn dữ liệu

- Dữ liệu bao gồm 1 vài biến về các chỉ số kinh tế, tài chính vĩ mô đo lường sức khỏe tài chính với tổng cộng của 270 quốc gia (bao gồm cả các khu tự trị và các khu vực khác trên toàn thế giới) lấy từ năm 1970 - 2023 được lấy từ trang web của World Bank (Ngân hàng Thế giới) là một kho tài liệu phong phú và đa dạng, cung cấp nhiều nguồn dữ liệu về kinh tế, phát triển, và các lĩnh vực liên quan.

1.2 Sơ lược về các biến.

- Để dự đoán khủng hoảng tài chính 1 cách tối ưu và phù hợp và chọn ra các biến phù hợp cho mô hình trước tiên cần tìm hiểu nguyên nhân khủng hoảng tài chính theo Radelet và Sachs (1998) , Chang và Velasco (2001) đã nhấn mạnh tầm quan trọng của thanh khoản tạm thời trong quá trình nợ nước ngoài tăng nhanh, ngoài ra các biến quan trọng quyết định khủng hoảng tài chính khác như tỉ lệ thất nghiệp, tỉ lệ lạm phát và cả mức tăng trưởng GDP qua các năm cũng được đề cập trong các tài liệu khác trước đó, dưới đây là các biến số trong mô hình.
- **Country Code**: Mã quốc gia
- **Country Name:** Tên quốc gia
- **Year** : Năm
- **short_term_deb**t: tỷ lệ nợ ngắn hạn trên tổng lượng dự trữ quốc gia.
- **external_debt:** tỷ lệ tổng nợ nước ngoài trên cho GNI của quốc gia.
- **total_reserves** : tỷ lệ tổng lượng dự trữ trên cho nợ nước ngoài.
- **cpi :** tỷ lệ lạm phát
- **unemployment** : tỷ lệ thất nghiệp
- **gdp_percentage** : % tăng trưởng GDP qua các năm
- **financial crisis** : biến giả dùng để dự đoán khủng hoảng tài chính với 0 là không xảy ra khủng hoảng tài chính và 1 là khủng hoảng tài chính.

  Để xây dựng Target column (**financial crisis**) tôi đã liệt kê các cuộc khủng hoảng tài chính lớn trên thế giới và các quốc gia bị khủng hoảng tài chính trong từng thời kỳ.
- Khủng Hoảng Tài Chính Châu Á (1997-1998)
- Khủng hoảng kinh tế ấn độ (1991)
- Khủng hoảng tài chính toàn cầu (2007-2008)
- Khủng hoảng tài chính nợ công ở trung quốc (2020-2021)
- Khủng hoảng kinh tế ở Siri Lanka (2022)
- Khủng hoảng tài chính Myanmar (2021-2022)
- Khủng hoảng tài chính Châu Âu (2010-2022)
- Khủng hoảng tài chính Ireland(2008-2010)

- Khủng hoảng tài chính Hy lạp (2010)
- Khủng Hoảng Tài Chính ở Bồ Đào Nha (2010-2014)
- Khủng Hoảng Tài Chính ở Tây Ban Nha (2008-2014)
- Khủng Hoảng Tài Chính ở Ý (2011-2013)
- Khủng Hoảng Tài Chính ở Châu Phi (1990s)
- Khủng Hoảng Tài Chính ở Zimbabwe (2000s)
- Khủng Hoảng Tài Chính ở Sudan (2010s)
- Khủng Hoảng Tài Chính ở Nam Phi (2020-2021)
- Khủng Hoảng Tài Chính ở Argentina (2001-2002)
- Khủng Hoảng Tài Chính ở Mexico (1994)
- Khủng Hoảng Tài Chính ở Úc (1990s)

	Country Code	Country Name	Year	short_term_debt	external_debt	total_reserves	cpi	unemployment	gdp_percentage	financial crisis
8259	AUS	Australia	1991	NaN	NaN	NaN	3.176675	9.586	-0.381875	1
8275	BRA	Brazil	1991	280.163990	30.470574	7.740140	432.786662	NaN	1.032190	1
8291	COL	Colombia	1991	25.607833	36.574066	39.339442	30.387820	10.120	2.001608	1
8355	IND	India	1991	92.831035	31.899926	8.975578	13.870246	NaN	1.056831	1
8420	NGA	Nigeria	1991	18.354982	NaN	13.953032	13.006973	NaN	0.358353	1
...	...	...	...	...	...	...	...	...	...	...
16630	LKA	Sri Lanka	2022	NaN	80.995176	NaN	49.721102	4.528	-7.823977	1
16644	MDV	Maldives	2022	54.216928	71.171517	20.838453	2.333143	NaN	13.906676	1
16652	MMR	Myanmar	2022	NaN	20.598582	NaN	NaN	NaN	4.037493	1
16670	NPL	Nepal	2022	3.453984	22.256532	101.958244	7.650792	NaN	5.613193	1
16676	PAK	Pakistan	2022	88.332273	34.359847	7.819767	19.873860	NaN	4.705803	1

177 rows × 10 columns

Hình 1: Dataframe của những quan sát xuất hiện khủng hoảng tài chính.

Ứng với mỗi cuộc khủng hoảng tài chính sẽ có 1 hoặc nhiều quốc gia xảy ra khủng hoảng tài chính và với quốc gia bị khủng hoảng thì **financial crisis** = 1 và ngược lại thì **financial crisis** = 0.

2. Wrangling Data

Vì dữ liệu lấy từ World Bank (Ngân hàng Thế giới) bao gồm 270 quốc gia bao gồm các khu tự trị và vùng lãnh thổ độc lập,... nên sẽ rất khó trong việc xử lý và dự đoán khủng hoảng tài chính vì thế nên trong nghiên cứu này ta chỉ lọc ra 193 quốc gia được thế giới công nhận và lọc từ năm 1991-2023.

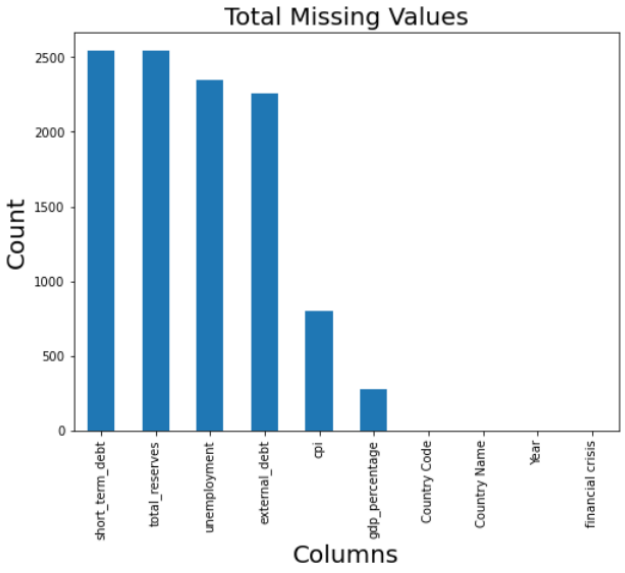
```
# Định nghĩa danh sách các quốc gia trên thế giới
recognized_countries = [
    "Afghanistan", "Albania", "Algeria", "Andorra", "Angola", "Antigua and Barbuda", "Argentina", "Armenia", "Australia", "Austri
    "Azerbaijan", "Bahamas", "Bahrain", "Bangladesh", "Barbados", "Belarus", "Belgium", "Belize", "Benin", "Bhutan",
    "Bolivia", "Bosnia and Herzegovina", "Botswana", "Brazil", "Brunei", "Bulgaria", "Burkina Faso", "Burundi", "Cabo Verde", "Ca
    "Cameroon", "Canada", "Central African Republic", "Chad", "Chile", "China", "Colombia", "Comoros", "Congo, Democratic Republi
    "Costa Rica", "Croatia", "Cuba", "Cyprus", "Czech Republic", "Denmark", "Djibouti", "Dominica", "Dominican Republic", "East T
    "Ecuador", "Egypt", "El Salvador", "Equatorial Guinea", "Eritrea", "Estonia", "Eswatini", "Ethiopia", "Fiji", "Finland",
    "France", "Gabon", "Gambia", "Georgia", "Germany", "Ghana", "Greece", "Grenada", "Guatemala", "Guinea",
    "Guinea-Bissau", "Guyana", "Haiti", "Honduras", "Hungary", "Iceland", "India", "Indonesia", "Iran", "Iraq",
    "Ireland", "Israel", "Italy", "Ivory Coast", "Jamaica", "Japan", "Jordan", "Kazakhstan", "Kenya", "Kiribati",
    "Korea, North", "Korea, South", "Kosovo", "Kuwait", "Kyrgyzstan", "Laos", "Latvia", "Lebanon", "Lesotho", "Liberia",
    "Libya", "Liechtenstein", "Lithuania", "Luxembourg", "Madagascar", "Malawi", "Malaysia", "Maldives", "Mali", "Malta",
    "Marshall Islands", "Mauritania", "Mauritius", "Mexico", "Micronesia", "Moldova", "Monaco", "Mongolia", "Montenegro", "Moroc
    "Mozambique", "Myanmar", "Namibia", "Nauru", "Nepal", "Netherlands", "New Zealand", "Nicaragua", "Niger", "Nigeria",
    "North Macedonia", "Norway", "Oman", "Pakistan", "Palau", "Palestine", "Panama", "Papua New Guinea", "Paraguay", "Peru",
    "Philippines", "Poland", "Portugal", "Qatar", "Romania", "Russia", "Rwanda", "Saint Kitts and Nevis", "Saint Lucia", "Saint V
    "Samoa", "San Marino", "Sao Tome and Principe", "Saudi Arabia", "Senegal", "Serbia", "Seychelles", "Sierra Leone", "Singapore
    "Slovenia", "Solomon Islands", "Somalia", "South Africa", "South Sudan", "Spain", "Sri Lanka", "Sudan", "Suriname", "Sweden",
    "Switzerland", "Syria", "Taiwan", "Tajikistan", "Tanzania", "Thailand", "Togo", "Tonga", "Trinidad and Tobago", "Tunisia",
    "Turkey", "Turkmenistan", "Tuvalu", "Uganda", "Ukraine", "United Arab Emirates", "United Kingdom", "United States", "Uruguay"
    "Vanuatu", "Vatican City (Holy See)","Russian Federation", "Venezuela, RB", "Viet Nam", "Yemen", "Zambia", "Zimbabwe","Korea,
]
```

```
# Lọc ra các dòng có "Country Name" trong danh sách quốc gia được công nhận trên thế giới
melted_df = melted_df[melted_df["Country Name"].isin(recognized_countries)]

# Lọc dữ liệu từ năm 1991 đến 2022 và chuyển đổi kiểu dữ liệu của cột "Year" sang số nguyên
melted_df["Year"] = melted_df["Year"].astype(int)
melted_df = melted_df[(melted_df["Year"] >= 1991) & (melted_df["Year"] <= 2023)]
```

2.1 Xử lý Missing data

- Dữ liệu trên World Bank (Ngân hàng Thế giới) hay xuất hiện giá trị NA vì 1 số lí do chẳng hạn như việc một số quốc gia hoặc khu vực không thu thập đầy đủ dữ liệu cần thiết hoặc không có khả năng báo cáo chính xác. Điều này có thể do hạn chế về tài chính, kỹ thuật, hoặc nhân lực. Ngoài ra, dữ liệu không được thu thập và báo cáo một cách đồng bộ giữa các quốc gia hoặc vùng lãnh thổ. Do đó, có thể có khoảng thời gian không có dữ liệu cho một số chỉ số.

In [253]:
```
import matplotlib.pyplot as plt
total = merged_df.isnull().sum().sort_values(ascending=False)
total_select = total.head(20)
total_select.plot(kind="bar", figsize = (8,6), fontsize = 10)

plt.xlabel("Columns", fontsize = 20)
plt.ylabel("Count", fontsize = 20)
plt.title("Total Missing Values", fontsize = 20)
```



Biểu đồ 1: Biểu đồ thể hiện số lượng missing data trong từng biến

- Qua biểu đồ 1, ta có thể thấy số lượng missing data là vô cùng lớn chiếm phần nhiều trong bộ dữ liệu để xử lý những giá trị này có rất nhiều phương pháp nhưng ở đây tôi sẽ sử dụngK-Nearest Neighbors (KNN) Imputation để dự đoán và điền các giá trị thiếu. KNN Imputation điền giá trị thiếu bằng cách tìm các điểm dữ liệu gần nhất trong không gian đa chiều và sử dụng trung bình hoặc trung vị của các điểm này để ước lượng.

```
# Khởi tạo KNNImputer với n_neighbors=5
imputer = KNNImputer(n_neighbors=5)
# Áp dụng KNNImputer cho các cột dữ liệu số
cleaned_data = pd.DataFrame(imputer.fit_transform(numeric_df), columns=numeric_df.columns)
```

- Kiểm tra lại số lượng missing data trong dataset.

```
missing_values = cleaned_data.isna().sum()
missing_values
```

Out[256]:

```
Country Code        0
Country Name        0
Year                0
short_term_debt     0
external_debt       0
total_reserves      0
cpi                 0
unemployment        0
gdp_percentage      0
financial crisis    0
dtype: int64
```

2.2 Xử lí Outlier

- Trước khi xử lí Outlier ta cần xử lí các giá trị trùng lặp (duplicated value)

In [38]:

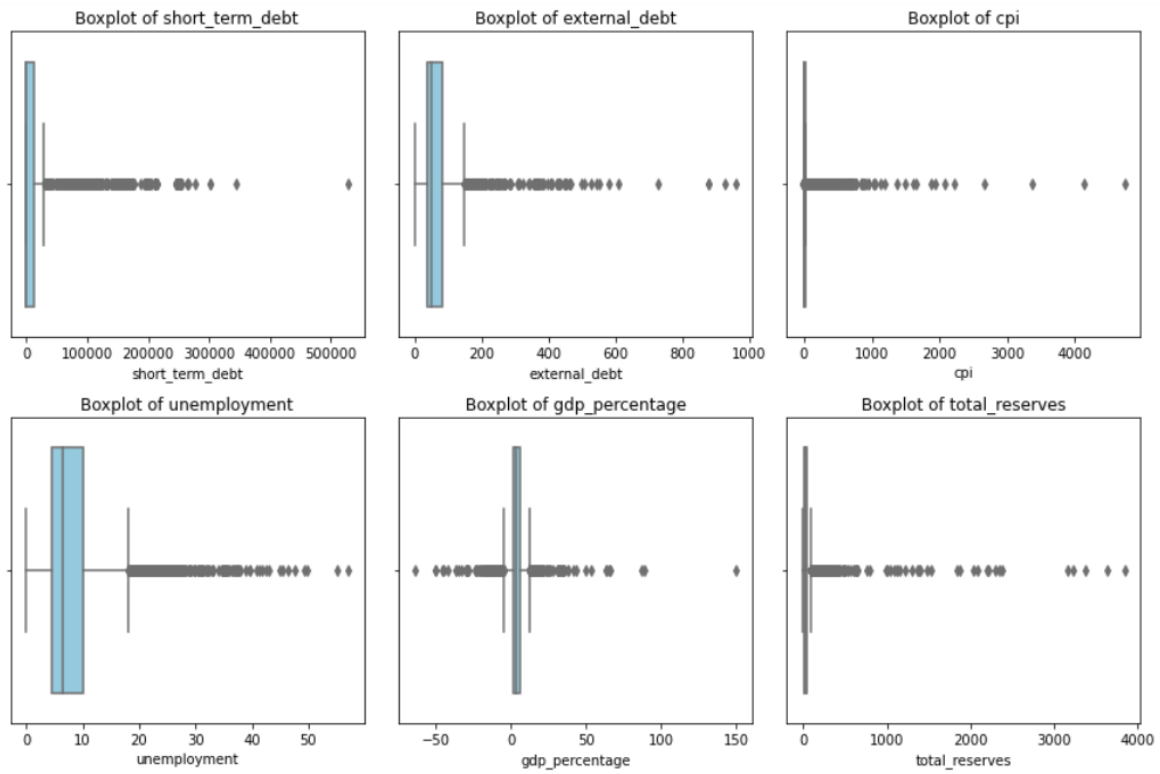
```
# xóa giá trị bị trùng lặp chỉ giữ lại giá trị đầu tiên
cleaned_data = cleaned_data.drop_duplicates(subset=[col for col in cleaned_data.columns if col != 'Year'])
# Kiểm tra xem còn giá trị trùng lặp không
duplicates = cleaned_data.duplicated(subset=[col for col in cleaned_data.columns if col != 'Year']).sum()

print(f'Số lượng giá trị trùng lặp còn lại: {duplicates}')
```

```
Số lượng giá trị trùng lặp còn lại: 0
```

- Có nhiều phương pháp để phát hiện ra những giá trị ngoại lệ như sử dụng z-score hay trực quan hóa dữ liệu, và sử dụng boxplot và trong nghiên cứu này tôi sẽ sử dụng boxplot.

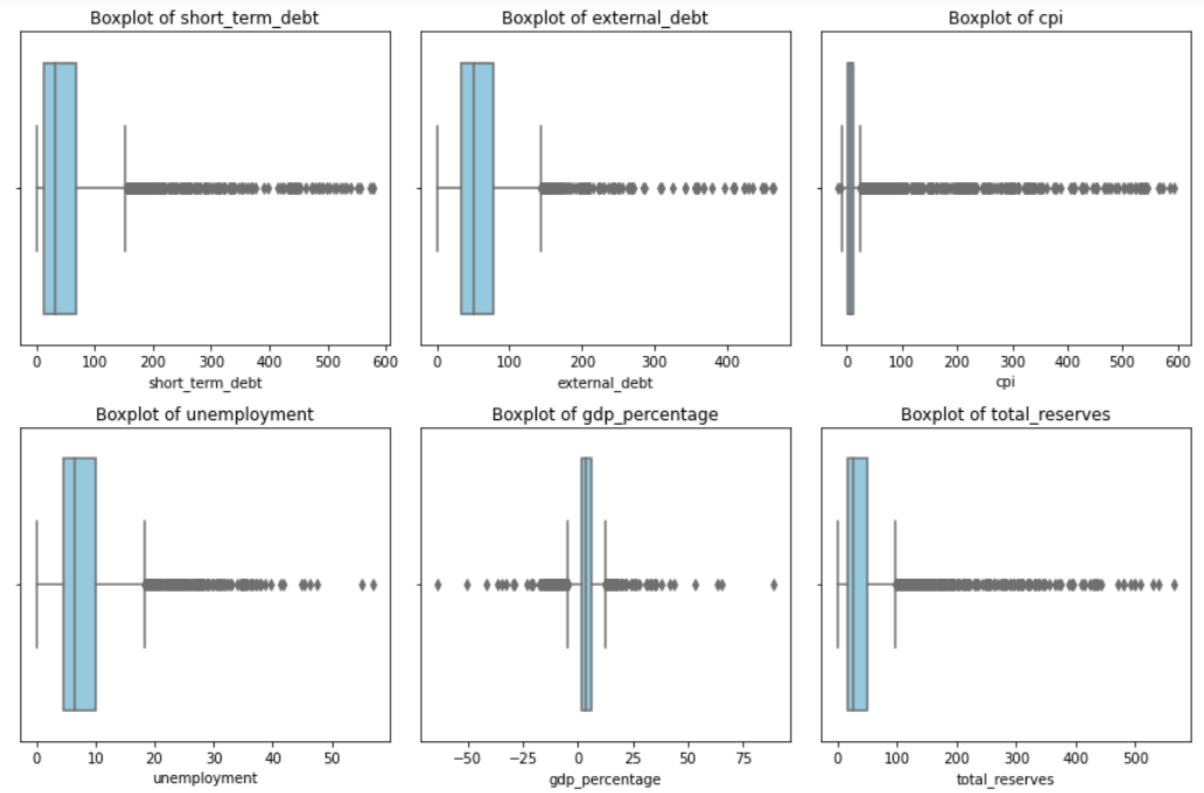
In [11]:

```
# xác định outlier
import seaborn as sns
import matplotlib.pyplot as plt
# Đặt kích thước của figure và số lượng subplot
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(12, 8))
# Flatten ma trận các axes thành một list để dễ truy cập
axes = axes.flatten()
variables = ['short_term_debt', 'external_debt', 'cpi', 'unemployment', 'gdp_percentage', 'total_reserves']
# Vẽ boxplot cho từng biến
for i, var in enumerate(variables):
    sns.boxplot(x=cleaned_data[var], color='skyblue', ax=axes[i])
    axes[i].set_title('Boxplot of {}'.format(var))
for j in range(len(variables), len(axes)):
    fig.delaxes(axes[j])
plt.tight_layout()
plt.show()
```



Biểu đồ 2: Biểu đồ boxplot của các biến(trước khi xử lí)

- Ta có thể thấy rằng ở biến **short_term_debt** mặc dù đơn vị phần trăm tuy nhiên có rất nhiều giá trị outlier lên tới hơn 100.000 đến 200.000 điều này có thể dẫn đến sai sót trong mô hình dự đoán, ta nên giữ lại tỷ lệ trong khoảng <500% vì những giá trị này có thể là những phát hiện khủng hoảng tài chính. Tương tự như vậy ở tỷ lệ nợ nước ngoài, ta cũng chỉ lấy giữ lại những giá trị <500%, ở **cpi** tỷ lệ lạm phát chỉ lấy những giá trị < 600% và tỷ lệ dự trữ **total_reserves** <600% và **external_debt** < 500%. Còn các như tỷ lệ thất nghiệp **unemployment, gdp_percentage** có thể thấy dữ liệu phân phối khá tốt các outlier có thể coi như phụ trợ trong phát hiện ra liệu rằng có khủng hoảng tài chính hay là không.



Biểu đồ 3: Boxplot của các biến (sau khi xử lí các outlier)

Sau khi xử lí các outlier ta có thể thấy phân phối dữ liệu trở nên hợp lý hơn, ta chỉ giữ lại những outlier có giá trị hợp lý và phù hợp cho mục đích dự đoán các cuộc khủng hoảng tài chính.

3.Explore Data Analysis (EDA)

3.1Thống kê mô tả

```
In [35]: cleaned_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3874 entries, 0 to 5741
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Country Code      3874 non-null   object 
 1   Country Name      3874 non-null   object 
 2   Year              3874 non-null   int32  
 3   short_term_debt   3874 non-null   float64
 4   external_debt     3874 non-null   float64
 5   total_reserves    3874 non-null   float64
 6   cpi               3874 non-null   float64
 7   unemployment      3874 non-null   float64
 8   gdp_percentage    3874 non-null   float64
 9   financial crisis  3874 non-null   float64
dtypes: float64(7), int32(1), object(2)
memory usage: 317.8+ KB
```

```
In [34]: #EDA
         #thống kê mô tả
         df = cleaned_data.drop(columns=['Country Code', 'Country Name'])
         stats_df = df.describe()
         stats_df
```

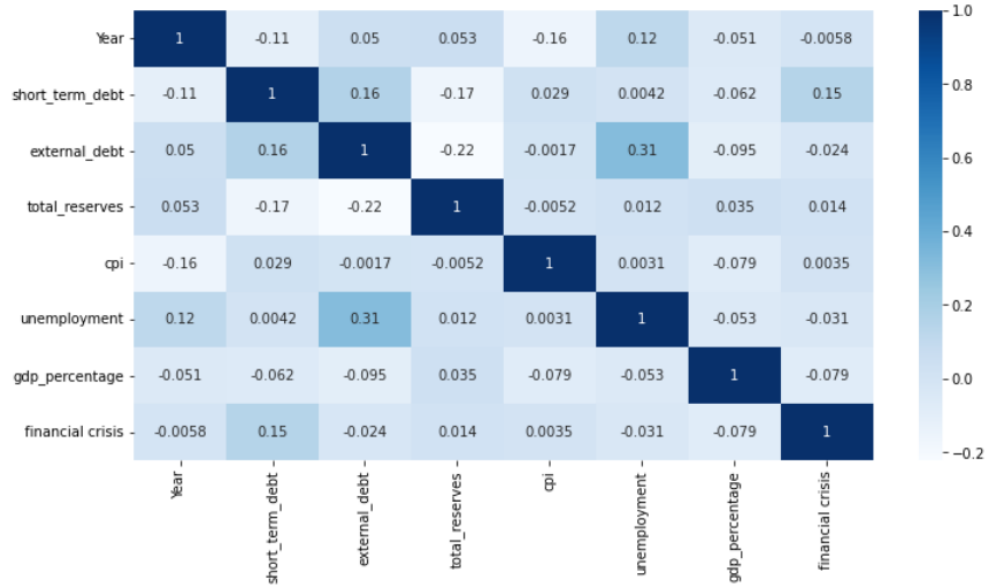
Out[34]:

	Year	short_term_debt	external_debt	total_reserves	cpi	unemployment	gdp_percentage	financial crisis
count	3874.000000	3874.000000	3874.000000	3874.000000	3874.000000	3874.000000	3874.000000	3874.000000
mean	2007.347186	52.738046	65.586122	46.251968	24.375162	8.870610	3.610833	0.038720
std	9.469208	71.206563	52.928702	59.309340	70.675332	7.375192	5.777616	0.192951
min	1991.000000	0.000000	0.463882	0.273598	-16.859691	0.039000	-64.047107	0.000000
25%	1999.000000	11.796626	32.881617	16.651150	2.474635	4.660650	1.646640	0.000000
50%	2007.000000	30.840840	51.432189	27.159095	5.382166	6.548000	3.777223	0.000000
75%	2015.000000	68.850640	77.038421	49.224182	11.372081	10.146750	6.015984	0.000000
max	2023.000000	578.639149	463.917774	565.568935	594.540387	57.000000	88.957666	1.000000

Hình 2: Thống kê mô tả của các biến.

- Sau khi tiền xử lý dữ liệu (các giá trị NA, giá trị trùng lặp và các outlier) bộ dữ liệu giảm còn 3874 quan sát từ năm 1991 đến năm 2023. Giá trị tối đa của **short_term_debt** là 578.63%, cho thấy một số quốc gia có mức nợ ngắn hạn rất cao. Tương tự như nợ ngắn hạn, nợ nước ngoài **external_debt** cũng có sự biến động lớn, với một số quốc gia có mức nợ nước ngoài rất cao. **cpi c**ó sự biến động rất lớn, từ giá trị âm (cho thấy giảm phát) đến giá trị rất cao (lạm phát cực cao). Điều này cho thấy tình hình lạm phát khác biệt lớn giữa các quốc gia và các thời kỳ khác nhau.Tỷ lệ khủng hoảng tài chính **financial crisis** là biến nhị phân (0 hoặc 1), với tỷ lệ trung bình khoảng 3.8%. Điều này cho thấy trong tập dữ liệu, có một tỷ lệ nhỏ các quan sát liên quan đến khủng hoảng tài chính.

```
In [22]: plt.figure(figsize=(12,6))
         sns.heatmap(df.corr(),annot=True,cmap='Blues')
         plt.show()
```



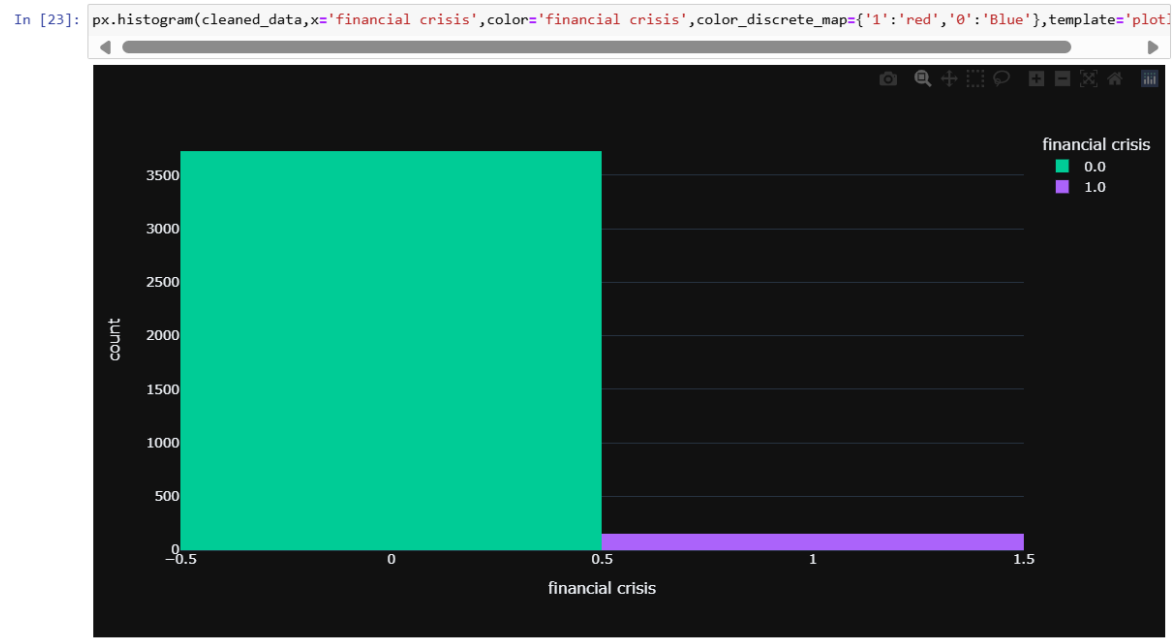
Hình 3: Ma trận tương quan giữa các biến

Biểu đồ ma trận tương quan cung cấp cái nhìn sâu sắc về mối quan hệ tuyến tính giữa các biến trong tập dữ liệu. Hệ số tương quan có giá trị từ -1 đến 1, trong đó:

- 1 cho thấy mối tương quan tuyến tính hoàn hảo dương.
- -1 cho thấy mối tương quan tuyến tính hoàn hảo âm.
- 0 cho thấy không có mối tương quan tuyến tính.

Qua ma trận tương quan ta có thể thấy các mối tương quan giữa các biến trong tập dữ liệu chủ yếu là yếu.

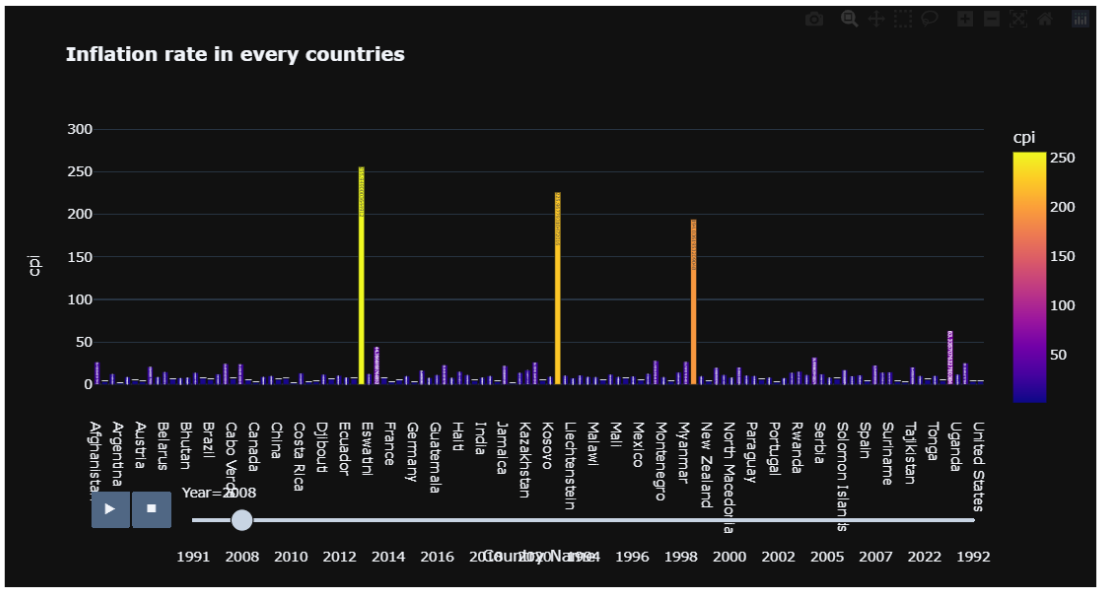
3.2 Trực quan hóa dữ liệu.



Biểu đồ 4: Phân phối dữ liệu của biến mục tiêu (Target variable)

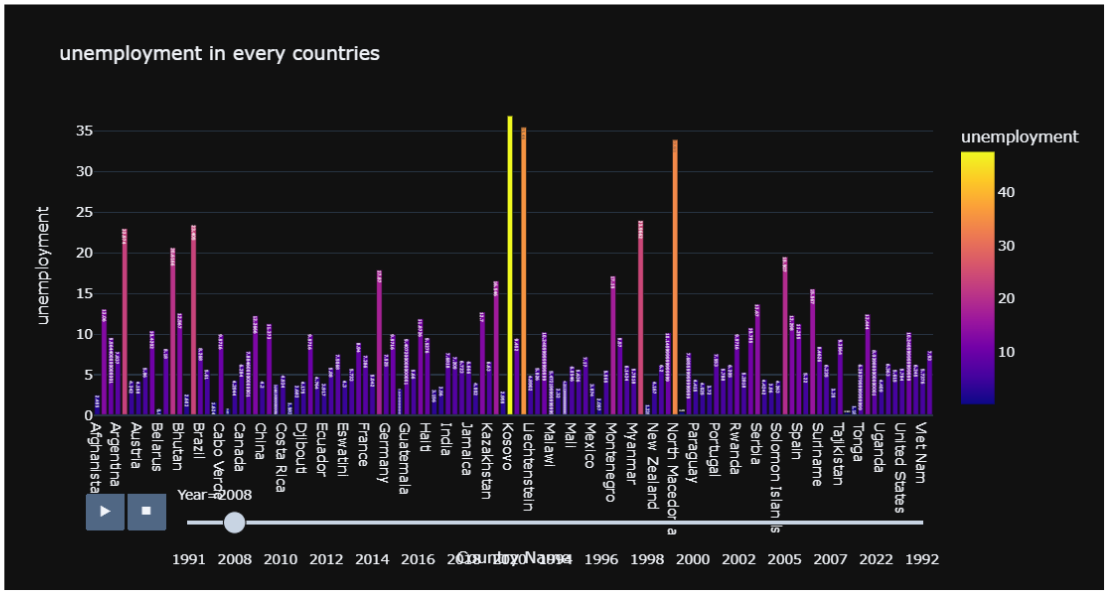
Ta có thể thấy, việc khủng hoảng tài chính xảy ra ở các quốc gia ở các thời kỳ chiếm rất ít trong bô dữ liệu. Điều này là vì từ năm 1991 tới năm 2023 xảy ra rất ít cuộc khủng hoảng tài chính và ở mỗi cuộc khủng hoảng chỉ có những quốc gia nhất định bị khủng hoảng. Điều này có thể gây ra mất cân bằng trong dữ liệu và thiên vị về lớp chiếm đa số Các mô hình có xu hướng dự đoán lớp chiếm đa số (label 0) nhiều hơn, vì chỉ cần dự đoán luôn là 0, mô hình đã đạt được độ chính xác cao (96.2%).Điều này dẫn đến việc mô hình có thể bỏ qua hoặc không học được các đặc điểm quan trọng của lớp chiếm thiểu số . Ta sẽ giải quyết ở phần sau.

```
In [20]: px.bar(cleaned_data,x='Country Name',y='cpi',color='cpi',animation_frame='Year',text='cpi',labels={'inflation_annual_cpi':'cpi ir
```



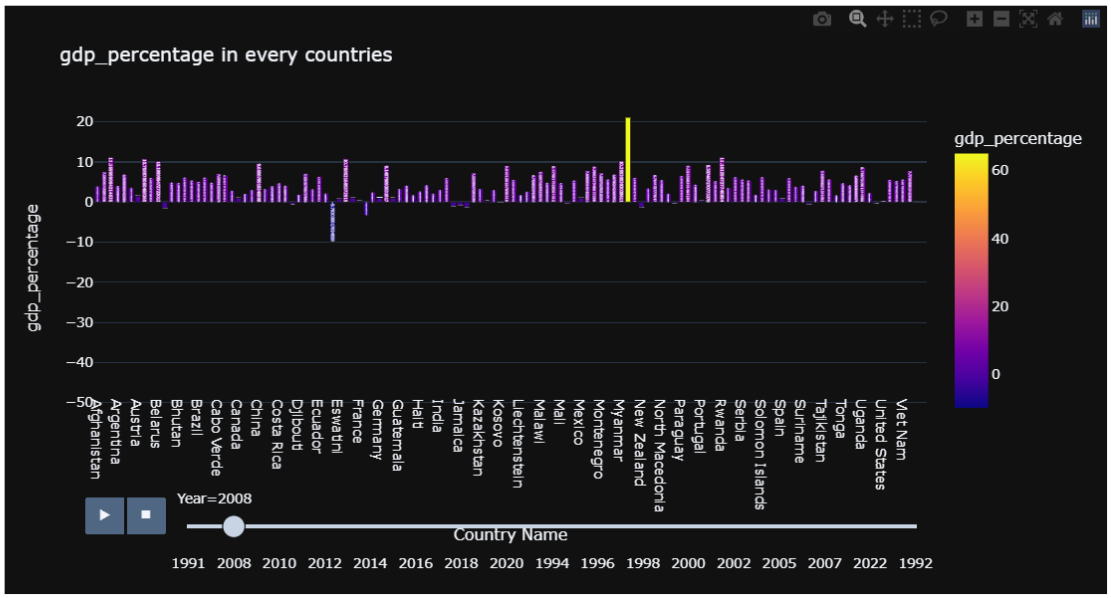
Biểu đồ 5: Tỷ lệ lạm phát ở các quốc gia trên thế giới năm 2008

```
In [21]: px.bar(cleaned_data,x='Country Name',y='unemployment',color='unemployment',animation_frame='Year',text='unemployment',labels={'ur
```



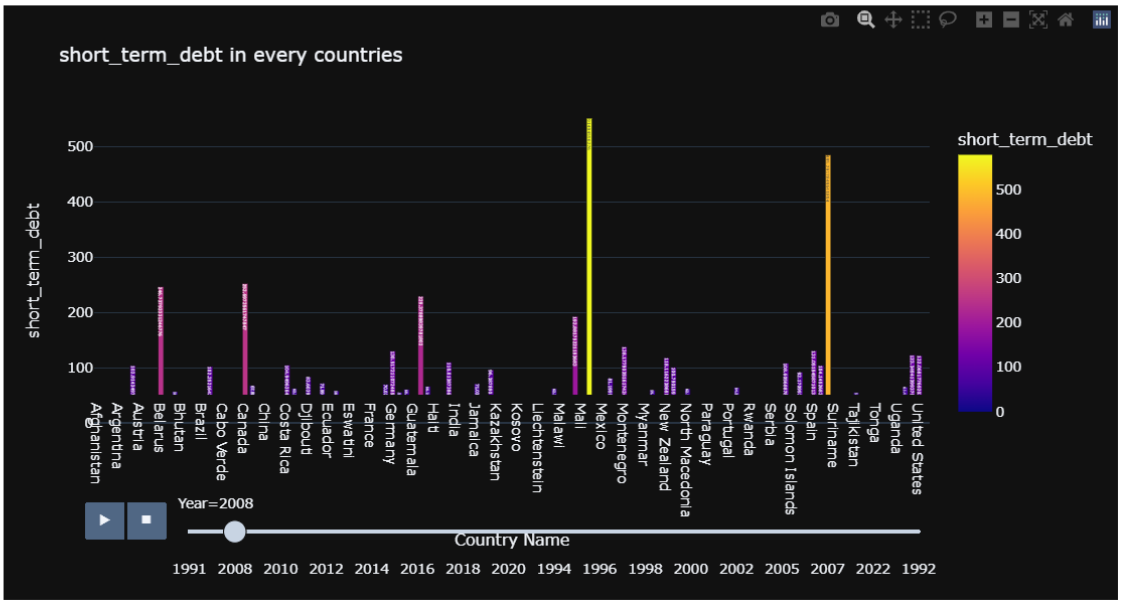
Biểu đồ 6: Tỷ lệ thất nghiệp ở các quốc gia trên thế giới năm 2008

```
In [22]: px.bar(cleaned_data,x='Country Name',y='gdp_percentage',color='gdp_percentage',animation_frame='Year',text='gdp_percentage',label
```



Biểu đồ 7 : phần trăm tăng trưởng GDP các quốc gia trên thế giới năm 2008

```
In [38]: px.bar(cleaned_data,x='Country Name',y='short_term_debt',color='short_term_debt',animation_frame='Year',text='short_term_debt',la
```



Biểu đồ 8: tỷ lệ nợ ngắn hạn của các quốc gia trên thế giới năm 2008

Vì bộ dữ liệu có nhiều năm và có tới 193 các quốc gia trên thế giới nên để dễ dàng trong việc trực quan dữ liệu ta sẽ chọn ra năm 2008- là 1 năm đặc biệt năm xảy ra cuộc khủng hoảng tài chính toàn cầu và thuận tiện hơn trong quá trình phân tích. Sau khi so sánh tỷ lệ thất nghiệp, tỷ lệ lạm phát và nợ ngắn hạn của các năm so với năm 2008, ta có thể thấy ở năm 2008, các chỉ số kinh tế này đều tăng cao đến mức đáng báo động, ở một số quốc gia có mức lạm phát rất cao, vượt trội so với các nước khác. Ví dụ, một số quốc gia có mức CPI lên đến hơn 250, trong khi nhiều quốc gia khác chỉ dao động ở mức thấp hơn (dưới 50). Đặc biệt, tỷ lệ thất nghiệp ở năm này tăng cao nhất trong các năm từ 1991-2023 có thể nói là cao nhất từ trước đến nay và phần trăm

tăng trưởng GDP cũng giảm đáng kể so giai đoạn trước và sau năm 2008. Điều này cho thấy bộ dữ liệu đang thể hiện đúng những gì nên diễn ra trong 1 cuộc khủng hoảng tài chính làm tăng thêm tính tin cậy cho bộ dữ liệu và tăng thêm độ chính xác cho mô hình máy học dự đoán trong việc ứng dụng nó vào trong thực tiễn.

4. Thử nghiệm các Model và chọn ra Model phù hợp.

- Để chọn ra mô hình phù hợp có độ chính xác cao ta sẽ thử nghiệm các mô hình máy học dự đoán.
- Để giải quyết vấn đề về việc mất cân bằng dữ liệu như đã nói ở phần phân phối của biến mục tiêu ta sẽ sử dụng kỹ thuật Random Over Samplingsẽ cải thiện hiệu suất của mô hình khi xử lý các lớp thiểu số, giúp mô hình không bị thiên vị về lớp đa số.

```
In [30]: from imblearn.over_sampling import RandomOverSampler
         from sklearn.preprocessing import LabelEncoder
         le=LabelEncoder()
         for col in df.columns:
             if df[col].dtype=='object':
                 df[col]=le.fit_transform(df[col])
         x=df.drop('financial crisis',axis=1)
         y=df['financial crisis']
         oversample=RandomOverSampler(sampling_strategy=0.15)
         x_over,y_over=oversample.fit_resample(x,y)
         print(x_over)
         print(y_over)
```

```
      Year  short_term_debt  external_debt  total_reserves        cpi  \
0     1991        70.799264     208.581427        7.238129   2.474635
1     2004         4.030371      54.900116       31.373527  13.982214
2     2008         0.603634      21.212133      141.900322  26.418664
3     2009         0.500545      20.036725      171.996793  -6.811161
4     2010         2.035235      15.333497      211.936351   2.178538
...    ...              ...            ...             ...        ...
4277  2008        22.676718      38.159777       25.121590   5.440782
4278  2007        31.276477      40.475277       44.120721   2.786797
4279  2008       131.051946      97.729500      331.688191   4.075343
4280  2010       101.521520      51.911864       54.068523   1.402573
4281  2008       122.069178      76.080885       36.173875   3.839100

      unemployment  gdp_percentage
0          34.9800        1.661940
1           5.4700        1.414118
2           2.4950        3.924984
3           8.9346       21.390528
4           6.7260       14.362441
...            ...             ...
4277        3.3200        4.831770
4278        8.2320        3.604738
4279       11.2550        0.887067
4280       10.7710        1.737625
4281        5.7840        0.122188

[4282 rows x 7 columns]
0       0.0
1       0.0
2       0.0
3       0.0
4       0.0
       ...
4277    1.0
4278    1.0
4279    1.0
4280    1.0
4281    1.0
Name: financial crisis, Length: 4282, dtype: float64
```

Tạo một đối tượng RandomOverSampler với chiến lược lấy mẫu lược lấy mẫu sampling_strategy = 0.15 sau đó sử dụng fit_resample(x, y) áp dụng kỹ thuật tăng cường mẫu lên dữ liệu, trả về dữ liệu cân bằng x_over và y_over.

In [31]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_over,y_over,test_size=0.30,random_state=42)
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[31]: ((2997, 7), (1285, 7), (2997,), (1285,))

Sau khi chia tập dữ liệu thì 30% dữ liệu sẽ được sử dụng để kiểm tra và 70% sẽ được sử dụng để huấn luyện.

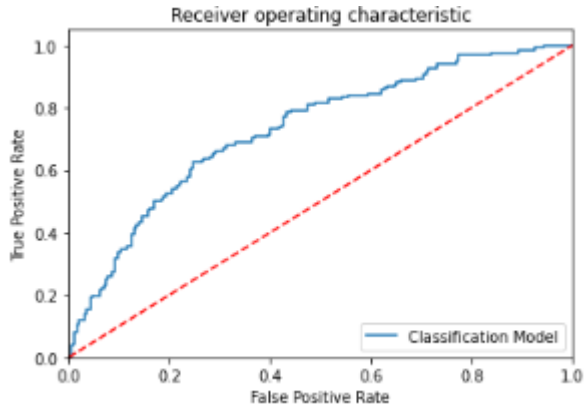
Hiệu suất các mô hình

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import GradientBoostingClassifier
import lightgbm as lgb
from sklearn.naive_bayes import GaussianNB
ModelLR = LogisticRegression()
ModelDC = DecisionTreeClassifier()
ModelRF = RandomForestClassifier()
ModelET = ExtraTreesClassifier()
ModelKNN = KNeighborsClassifier(n_neighbors=5)
ModelSVM = SVC(probability=True)
modelBAG = BaggingClassifier(base_estimator=None, n_estimators=100, max_samples=1.0, max_features=1.0,bootstrap=True, bootstrap_
ModelGB = GradientBoostingClassifier(loss='deviance', learning_rate=0.1,n_estimators=100, subsample=1.0,criterion='friedman_mse'
ModelLGB = lgb.LGBMClassifier()
ModelGNB = GaussianNB()
MM = [ModelLR, ModelDC, ModelRF, ModelET,ModelKNN, ModelSVM, modelBAG,ModelGB, ModelLGB, ModelGNB]
for models in MM:

    # Fit the model

    models.fit(x_train, y_train)

    # Prediction

    y_pred = models.predict(x_test)
    y_pred_prob = models.predict_proba(x_test)

    # Print the model name

    print('Model Name: ', models)

    # confusion matrix in sklearn

    from sklearn.metrics import confusion_matrix
    from sklearn.metrics import classification_report
    # actual values

    actual = y_test

    # predicted values

    predicted = y_pred

    # confusion matrix

    matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
    print('Confusion matrix : \n', matrix)
    tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
    print('Outcome values : \n', tp, fn, fp, tn)

    # classification report for precision, recall f1-score and accuracy

    C_Report = classification_report(actual,predicted,labels=[1,0])
    print('Classification report : \n', C_Report)

    # calculating the metrics
```

```
    sensitivity = round(tp/(tp+fn), 3);
    specificity = round(tn/(tn+fp), 3);
    accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
    balanced_accuracy = round((sensitivity+specificity)/2, 3);
    precision = round(tp/(tp+fp), 3);
    f1Score = round((2*tp/(2*tp + fp + fn)), 3);

    # Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
    # A model with a score of +1 is a perfect model and -1 is a poor model
    import math
    from math import sqrt

    mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
    MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
    print('Accuracy :', round(accuracy*100, 2),'%')
    print('Precision :', round(precision*100, 2),'%')
    print('Recall :', round(sensitivity*100,2), '%')
    print('F1 Score :', f1Score)
    print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
    print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
    print('MCC :', 'MCC')
     # Area under ROC curve

    from sklearn.metrics import roc_curve, roc_auc_score

    print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))

    # ROC Curve

    from sklearn.metrics import roc_auc_score
    from sklearn.metrics import roc_curve
    logit_roc_auc = roc_auc_score(actual, predicted)
    fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
    plt.figure()
     # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
    plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
    plt.plot([0, 1], [0, 1],'r--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic')
    plt.legend(loc="lower right")
    plt.savefig('Log_ROC')
    plt.show()
```

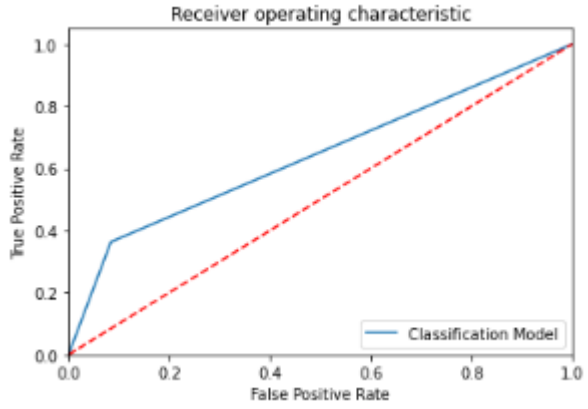
Mô hình LogisticRegression

```
Model Name:  LogisticRegression()
Confusion matrix :
 [[  14  154]
 [  16 1101]]
Outcome values :
 14 154 16 1101
Classification report :
              precision    recall  f1-score   support

           1       0.47      0.08      0.14       168
           0       0.88      0.99      0.93      1117

    accuracy                           0.87      1285
   macro avg       0.67      0.53      0.53      1285
weighted avg       0.82      0.87      0.83      1285

Accuracy : 86.8 %
Precision : 46.7 %
Recall : 8.3 %
F1 Score : 0.141
Specificity or True Negative Rate : 98.6 %
Balanced Accuracy : 53.4 %
MCC : MCC
roc_auc_score: 0.535
```



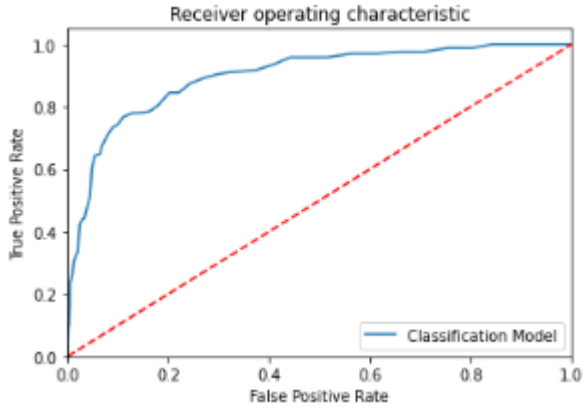
Mô hình DecisionTreeClassifier

```
Model Name:  DecisionTreeClassifier()
Confusion matrix :
 [[  61  107]
 [  94 1023]]
Outcome values :
 61 107 94 1023
Classification report :
              precision    recall  f1-score   support

           1       0.39      0.36      0.38       168
           0       0.91      0.92      0.91      1117

    accuracy                           0.84      1285
   macro avg       0.65      0.64      0.64      1285
weighted avg       0.84      0.84      0.84      1285

 Accuracy : 84.4 %
 Precision : 39.4 %
 Recall : 36.3 %
F1 Score : 0.378
Specificity or True Negative Rate : 91.6 %
Balanced Accuracy : 64.0 %
MCC : MCC
roc_auc_score: 0.639
```



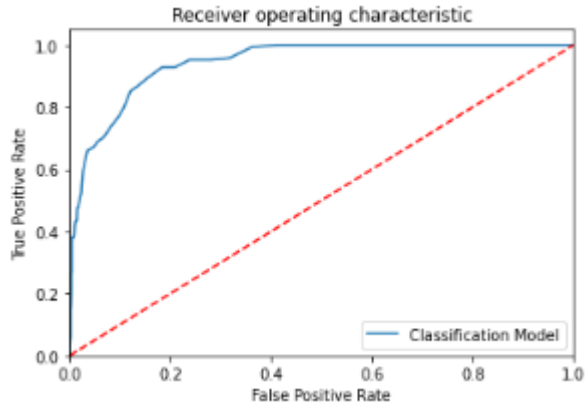
Mô hình RandomForestClassifier

```
Model Name:  RandomForestClassifier()
Confusion matrix :
 [[  41  127]
 [   7 1110]]
Outcome values :
 41 127 7 1110
Classification report :
              precision    recall  f1-score   support

           1       0.85      0.24      0.38       168
           0       0.90      0.99      0.94      1117

    accuracy                           0.90      1285
   macro avg       0.88      0.62      0.66      1285
weighted avg       0.89      0.90      0.87      1285

Accuracy : 89.6 %
Precision : 85.4 %
Recall : 24.4 %
F1 Score : 0.38
Specificity or True Negative Rate : 99.4 %
Balanced Accuracy : 61.9 %
MCC : MCC
roc_auc_score: 0.619
```



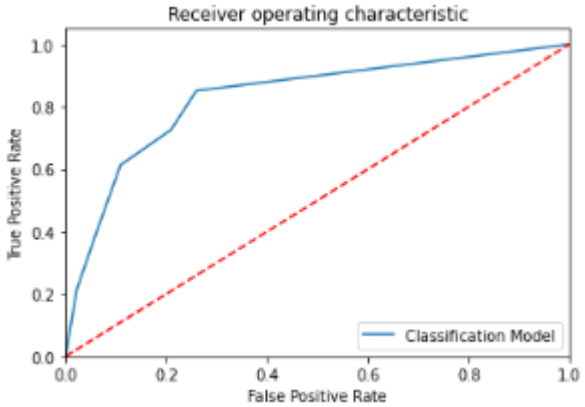
Mô hình ExtraTreesClassifier

```
Model Name:  ExtraTreesClassifier()
Confusion matrix :
 [[  34  134]
 [   4 1113]]
Outcome values :
 34 134 4 1113
Classification report :
              precision    recall  f1-score   support

           1       0.89      0.20      0.33       168
           0       0.89      1.00      0.94      1117

    accuracy                           0.89      1285
   macro avg       0.89      0.60      0.64      1285
weighted avg       0.89      0.89      0.86      1285

Accuracy : 89.3 %
Precision : 89.5 %
Recall : 20.2 %
F1 Score : 0.33
Specificity or True Negative Rate : 99.6 %
Balanced Accuracy : 59.9 %
MCC : MCC
roc_auc_score: 0.599
```



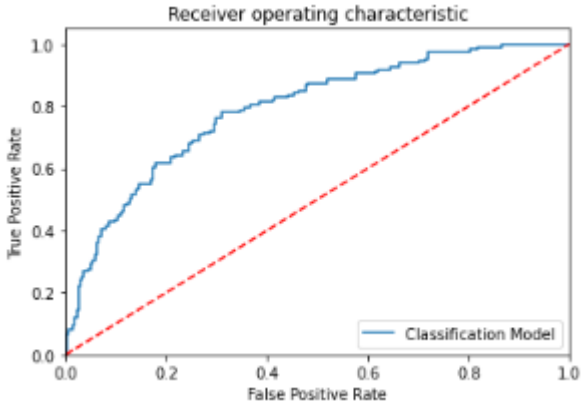
Mô hình KNeighborsClassifier

```
Confusion matrix :
 [[103  65]
 [122 995]]
Outcome values :
 103 65 122 995
Classification report :
              precision    recall  f1-score   support

           1       0.46      0.61      0.52       168
           0       0.94      0.89      0.91      1117

    accuracy                           0.85      1285
   macro avg       0.70      0.75      0.72      1285
weighted avg       0.88      0.85      0.86      1285

Accuracy : 85.4 %
Precision : 45.8 %
Recall : 61.3 %
F1 Score : 0.524
Specificity or True Negative Rate : 89.1 %
Balanced Accuracy : 75.2 %
MCC : MCC
roc_auc_score: 0.752
```



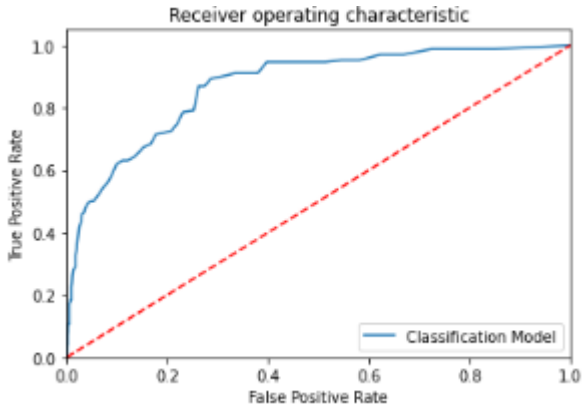
Mô hình SVC(probability=True)

```
Model Name:  SVC(probability=True)
Confusion matrix :
 [[  14  154]
 [   8 1109]]
Outcome values :
 14 154 8 1109
Classification report :
              precision    recall  f1-score   support

           1       0.64      0.08      0.15       168
           0       0.88      0.99      0.93      1117

    accuracy                           0.87      1285
   macro avg       0.76      0.54      0.54      1285
weighted avg       0.85      0.87      0.83      1285

Accuracy : 87.4 %
Precision : 63.6 %
Recall : 8.3 %
F1 Score : 0.147
Specificity or True Negative Rate : 99.3 %
Balanced Accuracy : 53.8 %
MCC : MCC
roc_auc_score: 0.538
```



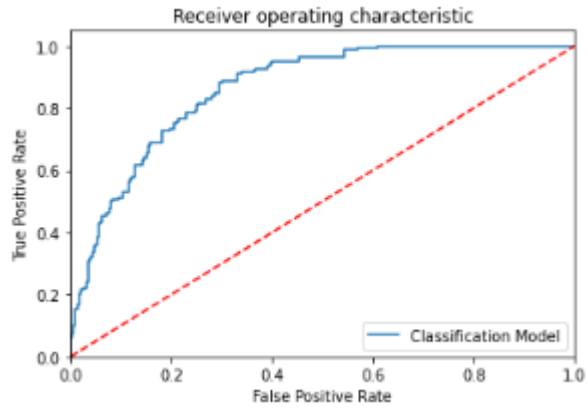
Mô hình BaggingClassifier

```
Model Name:  BaggingClassifier(n_estimators=100)
Confusion matrix :
 [[  55  113]
 [  20 1097]]
Outcome values :
 55 113 20 1097
Classification report :
              precision    recall  f1-score   support

           1       0.73      0.33      0.45       168
           0       0.91      0.98      0.94      1117

    accuracy                           0.90      1285
   macro avg       0.82      0.65      0.70      1285
weighted avg       0.88      0.90      0.88      1285

Accuracy : 89.6 %
Precision : 73.3 %
Recall : 32.7 %
F1 Score : 0.453
Specificity or True Negative Rate : 98.2 %
Balanced Accuracy : 65.4 %
MCC : MCC
roc_auc_score: 0.655
```



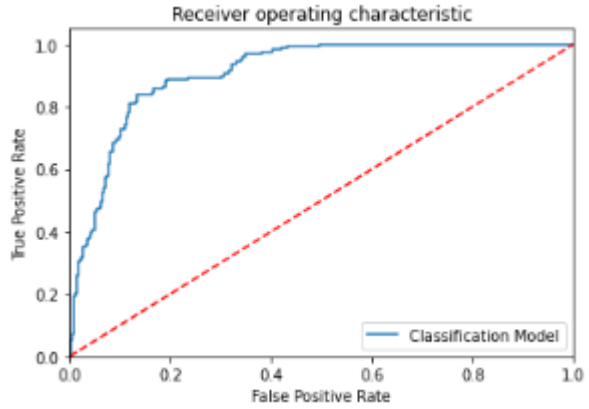
Mô hình GradientBoostingClassifier

```
Model Name:  GradientBoostingClassifier()
Confusion matrix :
 [[  51  117]
 [  40 1077]]
Outcome values :
 51 117 40 1077
Classification report :
              precision    recall  f1-score   support

           1       0.56      0.30      0.39       168
           0       0.90      0.96      0.93      1117

    accuracy                           0.88      1285
   macro avg       0.73      0.63      0.66      1285
weighted avg       0.86      0.88      0.86      1285

Accuracy : 87.8 %
Precision : 56.0 %
Recall : 30.4 %
F1 Score : 0.394
Specificity or True Negative Rate : 96.4 %
Balanced Accuracy : 63.4 %
MCC : MCC
roc_auc_score: 0.634
```



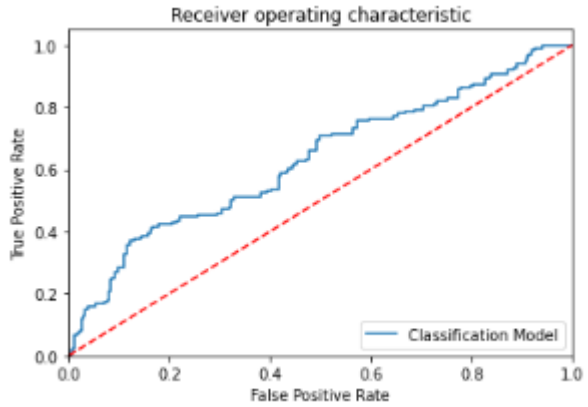
Mô hình LGBMClassifier

```
Model Name:  LGBMClassifier()
Confusion matrix :
 [[  40  128]
 [  15 1102]]
Outcome values :
 40 128 15 1102
Classification report :
              precision    recall  f1-score   support

           1       0.73      0.24      0.36       168
           0       0.90      0.99      0.94      1117

    accuracy                           0.89      1285
   macro avg       0.81      0.61      0.65      1285
weighted avg       0.87      0.89      0.86      1285

Accuracy : 88.9 %
Precision : 72.7 %
Recall : 23.8 %
F1 Score : 0.359
Specificity or True Negative Rate : 98.7 %
Balanced Accuracy : 61.2 %
MCC : MCC
roc_auc_score: 0.612
```



Mô hình GaussianNB

```
Model Name:  GaussianNB()
Confusion matrix :
 [[  29  139]
 [  81 1036]]
Outcome values :
 29 139 81 1036
Classification report :
              precision    recall  f1-score   support

           1       0.26      0.17      0.21       168
           0       0.88      0.93      0.90      1117

    accuracy                           0.83      1285
   macro avg       0.57      0.55      0.56      1285
weighted avg       0.80      0.83      0.81      1285

Accuracy : 82.9 %
Precision : 26.4 %
Recall : 17.3 %
F1 Score : 0.209
Specificity or True Negative Rate : 92.7 %
Balanced Accuracy : 55.0 %
MCC : MCC
roc_auc_score: 0.55
```



So sánh độ chính xác và hiệu suất giữa các mô hình

```
models={'models':['LogisticRegression','DecisionTreeClassifier','RandomForestClassifier','ExtraTreesClassifier','KNeighborsClas
out=pd.DataFrame(models)
out['Accuracy']=result['Accuracy']
out['Precision']=result['Precision']
out['Recall']=result['Recall']
out['F1 Score']=result['F1 Score']
out['True Positive']=result['True Positive']
out['False Negative']=result['False Negative']
out['False Positive']=result['False Positive']
out['True Negative']=result['True Negative']
out.sort_values(by=['Accuracy','F1 Score','Recall','Precision'],ascending=False,inplace=True)
out.reset_index(drop=True,inplace=True)
out
```

	models	Accuracy	Precision	Recall	F1 Score	True Positive	False Negative	False Positive	True Negative
0	BaggingClassifier	0.896	0.733	0.327	0.453	55.0	113.0	20.0	1097.0
1	RandomForestClassifier	0.896	0.854	0.244	0.380	41.0	127.0	7.0	1110.0
2	ExtraTreesClassifier	0.893	0.895	0.202	0.330	34.0	134.0	4.0	1113.0
3	LGBMClassifier	0.889	0.727	0.238	0.359	40.0	128.0	15.0	1102.0
4	GradientBoostingClassifier	0.878	0.560	0.304	0.394	51.0	117.0	40.0	1077.0
5	SVC	0.874	0.636	0.083	0.147	14.0	154.0	8.0	1109.0
6	LogisticRegression	0.868	0.467	0.083	0.141	14.0	154.0	16.0	1101.0
7	KNeighborsClassifier	0.854	0.458	0.613	0.524	103.0	65.0	122.0	995.0
8	DecisionTreeClassifier	0.844	0.394	0.363	0.378	61.0	107.0	94.0	1023.0
9	GaussianNB	0.829	0.264	0.173	0.209	29.0	139.0	81.0	1036.0

Từ kết quả trên ta có thể thấy **mô hình tốt nhất là RandomForestClassifier** với các chỉ số đánh giá như sau:

- **Độ chính xác (Accuracy):** 0.896 - cao nhất trong số các mô hình được so sánh.
- **Độ chính xác (Precision):** 0.854 - cao thứ hai trong số các mô hình được so sánh.
- **Độ thu hồi (Recall):** 0.244 - cao thứ ba trong số các mô hình được so sánh.
- **Điểm F1 (F1 Score):** 0.380 - cao thứ hai trong số các mô hình được so sánh

Mô hình Bagging Classifier tuy có độ chính xác cao tương đương với RandomForestClassifier, nhưng độ chính xác và độ thu hồi thấp hơn đáng kể.

Sử dụng mô hình (ModelRandomforest) để dự đoán trên tập dữ liệu kiểm tra (x_test). Các dự đoán này được lưu trữ trong y_pred.

```
y_pred=ModelRF.predict(x_test)
out=pd.DataFrame({'actual':y_test,'predicted':y_pred})
out=df_bk.merge(out,left_index=True,right_index=True)
out['actual'].replace(0,'no_crisis',inplace=True)
out['actual'].replace(1,'crisis',inplace=True)
out['predicted'].replace(0,'no_crisis',inplace=True)
out['predicted'].replace(1,'crisis',inplace=True)
out[['Country Name','Year','short_term_debt','external_debt','cpi','unemployment','gdp_percentage','total_reserves','predicted']]
```

	Country Name	Year	short_term_debt	external_debt	cpi	unemployment	gdp_percentage	total_reserves	predicted
1837	Georgia	2013	73.695443	80.621735	-0.512058	19.4160	3.615554	20.755179	no_crisis
227	Argentina	2020	108.128981	68.086370	160.302411	11.4610	-9.900485	15.418897	no_crisis
764	Bulgaria	1996	132.691563	87.137655	121.607542	12.5000	5.205393	8.335530	no_crisis
1775	France	2017	32.102501	51.088927	1.032283	9.4100	2.291420	19.162032	no_crisis
3774	Nicaragua	2003	117.519773	133.759919	5.302388	7.6000	2.520733	7.414964	no_crisis
1051	Chad	2019	8.551600	28.868254	-0.971939	11.6256	3.247182	9.639865	no_crisis
1038	Chad	2006	2.076726	28.267047	8.036289	10.2102	0.648262	35.973403	no_crisis
2584	Kazakhstan	2001	53.553959	72.845347	8.354138	10.4300	13.500000	16.335635	no_crisis
1967	Grenada	2011	13.110595	71.270795	3.033473	10.2406	0.764774	22.676825	no_crisis
3682	Nepal	2010	2.031488	23.488307	9.326504	9.3022	4.816415	79.280214	no_crisis

5. Dự báo khủng hoảng tài chính cho Việt Nam

In [149]:
```
# Giả sử df_bk là DataFrame chứa toàn bộ dữ liệu
vietnam_data = df_bk[df_bk['Country Name'] == 'Viet Nam']

# Lấy các cột đầu vào cho mô hình
input_columns = ['short_term_debt', 'external_debt', 'cpi', 'unemployment', 'gdp_percentage', 'total_reserves','financial crisis'
vietnam_input = vietnam_data[input_columns]
# Sử dụng mô hình để dự đoán cho dữ liệu Việt Nam
if not vietnam_input.empty:
    vietnam_pred = ModelRF.predict(vietnam_input)

    # Tạo DataFrame kết quả với các cột cần thiết
    vietnam_out = vietnam_data.copy()
    vietnam_out['predicted'] = vietnam_pred

    # Thay thế nhãn số bằng nhãn dạng chuỗi
    vietnam_out['predicted'].replace(0, 'no_crisis', inplace=True)
    vietnam_out['predicted'].replace(1, 'crisis', inplace=True)

    # Hiển thị kết quả
    print(vietnam_out[['Country Name', 'Year', 'short_term_debt', 'external_debt', 'cpi', 'unemployment', 'gdp_percentage', 'tota
else:
    print("No data available for Viet Nam.")
```

	Country Name	Year	short_term_debt	external_debt	cpi	\
5644	Viet Nam	1992	2.930855	255.919607	227.969045	
5672	Viet Nam	2020	28.052994	39.023491	3.220934	
5671	Viet Nam	2019	31.144804	38.573219	2.795824	
5657	Viet Nam	2005	22.544702	32.865160	8.284572	
5664	Viet Nam	2012	48.294180	32.873768	9.094703	
5674	Viet Nam	2022	44.088565	37.704187	3.156507	
5647	Viet Nam	1995	247.201617	123.008667	11.716473	
5650	Viet Nam	1998	109.527348	84.654986	7.266198	
5675	Viet Nam	2023	70.799264	208.581427	2.474635	
5646	Viet Nam	1994	7.630729	156.576321	201.986069	

	unemployment	gdp_percentage	total_reserves	predicted
5644	5.2768	8.646047	437.287267	no_crisis
5672	2.1030	2.865413	73.242303	no_crisis
5671	1.6810	7.359263	63.948907	no_crisis
5657	6.0144	7.547248	48.675831	no_crisis
5664	1.0270	5.504545	41.081372	no_crisis
5674	1.5240	8.019792	59.020185	no_crisis
5647	8.0824	9.540480	5.194307	no_crisis
5650	2.2900	5.764455	8.894078	no_crisis
5675	34.9800	1.661940	7.238129	no_crisis
5646	5.3544	8.838981	310.066413	no_crisis

Khi sử dụng mô hình để dự đoán ngẫu nhiên các năm thì ta có thể thấy mô hình dự đoán khá chính xác về tính hình khủng hoảng tài chính ở việt nam qua các thời kỳ. Điều này góp phần làm tăng độ uy tín và chính xác của mô hình dự đoán khủng hoảng tài chính mà ta đã xây dựng,

6. Kết luận.

Mô hình dự báo khủng hoảng tài chính là một công cụ quan trọng hỗ trợ cho việc phòng ngừa và giảm thiểu rủi ro trong hệ thống tài chính toàn cầu và có tính áp dụng trong thực tiễn cao. Trong nghiên cứu này, mô hình máy học dự đoán khủng hoảng tài chính đã được xây dựng thành công với khả năng dự báo khủng hoảng tài chính cho các quốc gia trên thế giới với độ chính xác tương đối cao.

Hứa hẹn trong tương lai nghiên cứu và phát triển mô hình với việc nâng cao độ chính xác và hiệu quả dự báo và có thể mở rộng phạm vi dự báo sang các loại khủng hoảng khác như khủng hoảng kinh tế, khủng hoảng môi trường, v.v. sẽ góp phần nâng cao hiệu quả hoạt động của hệ thống tài chính và thúc đẩy sự phát triển kinh tế bền vững. Ngoài ra, hỗ trợ các nhà hoạch định chính sách đưa ra quyết định kịp thời và hiệu quả để ngăn ngừa hoặc giảm thiểu tác động của khủng hoảng tài chính.