Using SVM to build Democracy indicators

In [61]:
```
#Importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

In [62]:
```
#Importing the V-dem dataset
data_vdem = pd.read_csv("V-Dem-CY-Full+Others-v10.csv")
```

The variables that interest us from the v-dem dataset are: oppositions party autonomy (*v2psoppaut*), party ban (*v2psparban*), the percentage of adult population with legal right to vote (*v2asuffrage*), the percentage of enfranchised adults older than the minimal voting age (*v2elsuffrage*), and female suffrage (*v2fsuffrage*):

In [63]:
```
data=data_vdem[["country_name","year","v2psoppaut","v2psparban","v2asuffrage","v2elsuffrage", "v2fsuffrage"]]
```

We want to create a dataframe where we find the **atributes** ("v2psoppaut","v2psparban","v2asuffrage","v2elsuffrage", "v2fsuffrage") and **lables** (1, corresponding to democracies, and -1, corresponding to autocracies), so we can create our classifier (SVM). To do so, we need country-year observations that contain labels (1 or -1). Following Grundler and Krieger, we assume that a country-year observation is democratic (thus, its label=1) if its polity score is 10, and autocratic if its polity score is -10. Let us construct our dataframe then, following these steps:

1. We first import the Polity dataset, and then select the democratic/autocratic observations
2. We add the label 1 or -1 to the country-year observations that are democratic or autocratic, respectively
3. Take the country-year observations that were selected and add their attributes; putting all this information in the same table (to further apply the SVM)

Importing data from the polity database:

In [64]:
```
polity = pd.read_excel("politydata.xls")
```

Building the priming dataset

Selecting country-year observations that are democratic:

In [65]:
```
is_democracy= polity.polity2==10
democracy=polity[is_democracy]
onlydemo=democracy[["country","year"]]
```

Selecting country-year observations that are autocratic:

In [66]:
```
is_autocracy=polity.polity2<=-9
autocracies=polity[is_autocracy]
onlyauto=autocracies[["country","year"]]
onlyauto
```

Out[66]:

	country	year
145	Afghanistan	1945
146	Afghanistan	1946
147	Afghanistan	1947
148	Afghanistan	1948
149	Afghanistan	1949
...	...	...
17502	Zambia	1986
17503	Zambia	1987
17504	Zambia	1988
17505	Zambia	1989
17506	Zambia	1990

2498 rows × 2 columns

Adding the labels 1 (democratic) or -1 (autocratic) for each selected country-year observation:

Creating the table with attributes and labels:

In [67]:
```
#'Prussia', 'Yemen North' and 'Yugoslavia' do not appear in the v-dem dataset.
onlyauto.replace({'country': 'Congo Kinshasa'}, {'country': 'Democratic Republic of the Congo'}, inplace=True)
onlyauto.replace({'country': 'Myanmar (Burma)'}, {'country': 'Burma/Myanmar'}, inplace=True)
onlyauto.replace({'country': 'Cote D''Ivoire'}, {'country': 'Ivory Coast'}, inplace=True)
onlyauto.replace({'country': 'Cote D''Ivoire'}, {'country': 'United Arab Emirates'}, inplace=True)
onlyauto.replace({'country': 'Congo-Brazzaville'}, {'country': 'Republic of the Congo'}, inplace=True)
onlyauto.replace({'country': 'Korea North'}, {'country': 'North Korea'}, inplace=True)
onlyauto.replace({'country': 'Korea South'}, {'country': 'South Korea'}, inplace=True)
```

```
onlyauto.replace({'country': 'Sardinia'}, {'country': 'Piedmont-Sardinia'}, inplace=True)
onlyauto.replace({'country': 'Swaziland'}, {'country': 'Eswatini'}, inplace=True)
```

In [68]:
```
# After analysing the country names, we notice that in the polity dataset we have "Slovak Republic" and "United States" that ap
#... as "Slovakia" and "United States of America" in the v-dem dataset. Let us change the names then:
onlydemo.replace({'country': 'Slovak Republic'}, {'country': 'Slovakia'}, inplace=True)
onlydemo.replace({'country': 'United States                    '}, {'country': 'United States of America'}, inplace=True)
```

In [69]:
```
data.rename(columns={"country_name": "country"}, inplace = True) #we need to change the column name "country_name" from the v-d
                                                                  #to "country" (so it has the same column name as in the polity
                                                                  #otherwise the programe would not match country observations
democracies=pd.merge(onlydemo,data) #creates a table with democratic country-year observations (label = 1, and respective attri
autocracies=pd.merge(onlyauto,data) #creates a table with autocratic country-year observations (label = -1, and respective attr
```

In [70]:
```
a=onlyauto.country.unique()
```

In [71]:
```
b=autocracies.country.unique()
```

In [72]:
```
list(set(a) - set(b))
```

Out[72]:
```
['Prussia',
 'Germany East',
 'Korea South',
 "Cote D'Ivoire",
 'Yugoslavia',
 'Yemen North',
 'Vietnam North',
 'USSR']
```

In [73]:
```
autocracies
```

Out[73]:

	country	year	v2psoppaut	v2psparban	v2asuffrage	v2elsuffrage	v2fsuffrage
0	Afghanistan	1945	-1.541	-1.344	50.0	50.0	0.0
1	Afghanistan	1946	-1.541	-1.344	50.0	50.0	0.0
2	Afghanistan	1947	-1.541	-1.344	50.0	50.0	0.0
3	Afghanistan	1948	-1.541	-1.344	50.0	50.0	0.0
4	Afghanistan	1949	-1.541	-1.344	50.0	50.0	0.0
...	...	...	...	...	...	...	...
2312	Zambia	1986	-3.299	-1.938	100.0	100.0	100.0
2313	Zambia	1987	-3.299	-1.938	100.0	100.0	100.0
2314	Zambia	1988	-3.299	-1.938	100.0	100.0	100.0
2315	Zambia	1989	-3.299	-1.938	100.0	100.0	100.0
2316	Zambia	1990	-3.299	-1.938	100.0	100.0	100.0

2317 rows × 7 columns

In [74]:
```
democracies
```

Out[74]:

	country	year	v2psoppaut	v2psparban	v2asuffrage	v2elsuffrage	v2fsuffrage
0	Australia	1901	2.378	1.823	100.0	65.0	100.0
1	Australia	1902	2.378	1.823	100.0	90.0	100.0
2	Australia	1903	2.378	1.823	100.0	90.0	100.0
3	Australia	1904	2.378	1.823	100.0	90.0	100.0
4	Australia	1905	2.378	1.823	100.0	90.0	100.0
...	...	...	...	...	...	...	...
2276	United States of America	2011	2.901	2.153	100.0	100.0	100.0
2277	United States of America	2012	2.901	2.153	100.0	100.0	100.0
2278	United States of America	2013	2.859	2.038	100.0	100.0	100.0
2279	United States of America	2014	2.859	2.038	100.0	100.0	100.0
2280	United States of America	2015	2.859	2.038	100.0	100.0	100.0

2281 rows × 7 columns

In [75]:
```
onlydemo['Label']=1
onlyauto['Label']=-1
```

In [76]:
```
democracies=pd.merge(onlydemo,data) #creates a table with democratic country-year observations (label = 1, and respective attri
autocracies=pd.merge(onlyauto,data) #creates a table with autocratic country-year observations (label = -1, and respective attr
frames=[democracies,autocracies]
table0=pd.concat(frames)           #creates a table with both democratic and autocratic country-year observations
table0
```

Out[76]:

	country	year	Label	v2psoppaut	v2psparban	v2asuffrage	v2elsuffrage	v2fsuffrage
0	Australia	1901	1	2.378	1.823	100.0	65.0	100.0
1	Australia	1902	1	2.378	1.823	100.0	90.0	100.0
2	Australia	1903	1	2.378	1.823	100.0	90.0	100.0
3	Australia	1904	1	2.378	1.823	100.0	90.0	100.0
4	Australia	1905	1	2.378	1.823	100.0	90.0	100.0
...	...	...	...	...	...	...	...	...
2312	Zambia	1986	-1	-3.299	-1.938	100.0	100.0	100.0
2313	Zambia	1987	-1	-3.299	-1.938	100.0	100.0	100.0
2314	Zambia	1988	-1	-3.299	-1.938	100.0	100.0	100.0
2315	Zambia	1989	-1	-3.299	-1.938	100.0	100.0	100.0
2316	Zambia	1990	-1	-3.299	-1.938	100.0	100.0	100.0

4598 rows × 8 columns

In [77]:

```
table0.dropna(inplace=True)  #Drop NaN cells, otherwise the SVM does not work

#Moreover, we do not need the information about the country or the year for each observation, so let us work with a table that
#...contains the atributes and the labels
table1=table0[["v2psoppaut", "v2psparban", "v2asuffrage", "v2elsuffrage", "v2fsuffrage", "Label"]]
```

In [78]:

```
table1
```

Out[78]:

	v2psoppaut	v2psparban	v2asuffrage	v2elsuffrage	v2fsuffrage	Label
0	2.378	1.823	100.0	65.0	100.0	1
1	2.378	1.823	100.0	90.0	100.0	1
2	2.378	1.823	100.0	90.0	100.0	1
3	2.378	1.823	100.0	90.0	100.0	1
4	2.378	1.823	100.0	90.0	100.0	1
...	...	...	...	...	...	...
2312	-3.299	-1.938	100.0	100.0	100.0	-1
2313	-3.299	-1.938	100.0	100.0	100.0	-1
2314	-3.299	-1.938	100.0	100.0	100.0	-1
2315	-3.299	-1.938	100.0	100.0	100.0	-1
2316	-3.299	-1.938	100.0	100.0	100.0	-1

3294 rows × 6 columns

Applying the SVM

In [79]:

```
# Distinction between attributes and labels
X = table1.drop('Label', axis=1)
y = table1['Label']

# Dividing our data in training set and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)
```

Testing different kernels, for classification:

In [80]:

```
# Training the algorithm
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)

# Making predictions in the test set
y_pred = svclassifier.predict(X_test)

# Evaluating the algorithm
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[281   0]
 [  5 538]]
              precision    recall  f1-score   support

          -1       0.98      1.00      0.99       281
           1       1.00      0.99      1.00       543

    accuracy                           0.99       824
```

```
   macro avg       0.99      1.00      0.99       824
weighted avg       0.99      0.99      0.99       824
```

In [81]:

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='poly', degree=2)
svclassifier.fit(X_train, y_train)

y_pred = svclassifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[274   7]
 [ 33 510]]
              precision    recall  f1-score   support

          -1       0.89      0.98      0.93       281
           1       0.99      0.94      0.96       543

    accuracy                           0.95       824
   macro avg       0.94      0.96      0.95       824
weighted avg       0.95      0.95      0.95       824
```

In [82]:

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='rbf')
svclassifier.fit(X_train, y_train)

y_pred = svclassifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[271  10]
 [ 33 510]]
              precision    recall  f1-score   support

          -1       0.89      0.96      0.93       281
           1       0.98      0.94      0.96       543

    accuracy                           0.95       824
   macro avg       0.94      0.95      0.94       824
weighted avg       0.95      0.95      0.95       824
```

In [83]:

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='sigmoid')
svclassifier.fit(X_train, y_train)

y_pred = svclassifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[151 130]
 [ 49 494]]
              precision    recall  f1-score   support

          -1       0.76      0.54      0.63       281
           1       0.79      0.91      0.85       543

    accuracy                           0.78       824
   macro avg       0.77      0.72      0.74       824
weighted avg       0.78      0.78      0.77       824
```

The linear kernel presents the highest accuracy (99%).

Applying the linear kernel to classify the entire sample

In [84]:

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)

y_pred = svclassifier.predict(X_test)



## 2. Apply the classifier to the rest of the country-year observations from the v-dem dataset:
data.dropna(inplace=True)

Atrib=data[["v2psoppaut","v2psparban","v2asuffrage","v2elsuffrage", "v2fsuffrage"]]
Label_pred = svclassifier.predict(Atrib)     # Classifies our country-year observations in democracy/autocracy (label 1 or -1)
#data['PredictedLabel'] = Label_pred         # Creates a table with country-year observations, attributes, and labels (predicted

BinaryIndicator=data
BinaryIndicator['PredictedLabel'] = Label_pred
BinaryIndicator
```

Out[84]:

	country	year	v2psoppaut	v2psparban	v2asuffrage	v2elsuffrage	v2fsuffrage	PredictedLabel

	country	year	v2psoppaut	v2psparban	v2asuffrage	v2elsuffrage	v2fsuffrage	PredictedLabel
111	Mexico	1900	-0.570	-0.532	50.0	50.0	0.0	-1
112	Mexico	1901	-0.570	-0.532	50.0	50.0	0.0	-1
113	Mexico	1902	-0.570	-0.532	50.0	50.0	0.0	-1
114	Mexico	1903	-0.570	-0.532	50.0	50.0	0.0	-1
115	Mexico	1904	-0.570	-0.532	50.0	50.0	0.0	-1
...	...	...	...	...	...	...	...	...
25618	Zanzibar	2015	0.455	0.764	100.0	100.0	100.0	-1
25619	Zanzibar	2016	0.455	0.764	100.0	100.0	100.0	-1
25620	Zanzibar	2017	0.596	0.761	100.0	100.0	100.0	-1
25621	Zanzibar	2018	0.700	0.494	100.0	100.0	100.0	-1
25622	Zanzibar	2019	0.700	0.494	100.0	100.0	100.0	-1

17796 rows × 8 columns

Testing and applying the linear kernel in a regression (to create a continuous indicator)

In [85]:

```
# CONTINUOUS INDICATOR

from sklearn.svm import SVR
svregression = SVR(kernel='linear')
svregression.fit(X_train, y_train)

y_pred = svregression.predict(X_test)

accuracy = svregression.score(X_test, y_test)
print(accuracy)
```

```
0.909632014107809
```

In [86]:

```
data.dropna(inplace=True)
Atrib=data[["v2psoppaut","v2psparban","v2asuffrage","v2elsuffrage", "v2fsuffrage"]]
Score = svregression.predict(Atrib)
ContinuousIndicator=data
ContinuousIndicator['Score'] = Score
ContinuousIndicator
```

Out[86]:

	country	year	v2psoppaut	v2psparban	v2asuffrage	v2elsuffrage	v2fsuffrage	PredictedLabel	Score
111	Mexico	1900	-0.570	-0.532	50.0	50.0	0.0	-1	-0.463621
112	Mexico	1901	-0.570	-0.532	50.0	50.0	0.0	-1	-0.463621
113	Mexico	1902	-0.570	-0.532	50.0	50.0	0.0	-1	-0.463621
114	Mexico	1903	-0.570	-0.532	50.0	50.0	0.0	-1	-0.463621
115	Mexico	1904	-0.570	-0.532	50.0	50.0	0.0	-1	-0.463621
...	...	...	...	...	...	...	...	...	...
25618	Zanzibar	2015	0.455	0.764	100.0	100.0	100.0	-1	0.357174
25619	Zanzibar	2016	0.455	0.764	100.0	100.0	100.0	-1	0.357174
25620	Zanzibar	2017	0.596	0.761	100.0	100.0	100.0	-1	0.395672
25621	Zanzibar	2018	0.700	0.494	100.0	100.0	100.0	-1	0.383335
25622	Zanzibar	2019	0.700	0.494	100.0	100.0	100.0	-1	0.383335

17796 rows × 9 columns

Results

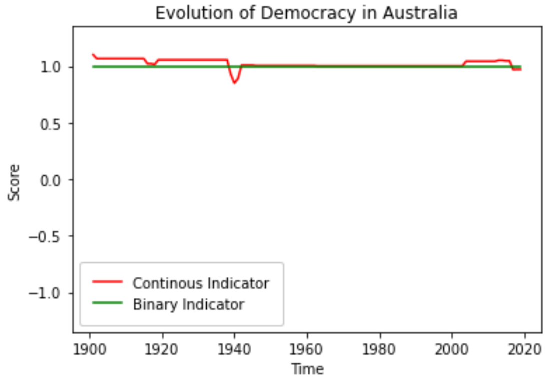
Let us see the evolution of the indicators (binary and continuous) for some countries:

Australia

In [132...

```
is_Australia= ContinuousIndicator.country=="Australia"
Australia=ContinuousIndicator[is_Australia]

plt.plot(Australia.year, Australia.Score, 'r', label="Continous Indicator")
plt.plot(Australia.year, Australia.PredictedLabel, 'g', label="Binary Indicator")
plt.legend(fancybox=True, framealpha=1, borderpad=1)
plt.title('Evolution of Democracy in Australia')
plt.xlabel('Time')
plt.ylabel('Score')
plt.ylim((-1.35,1.35))
```

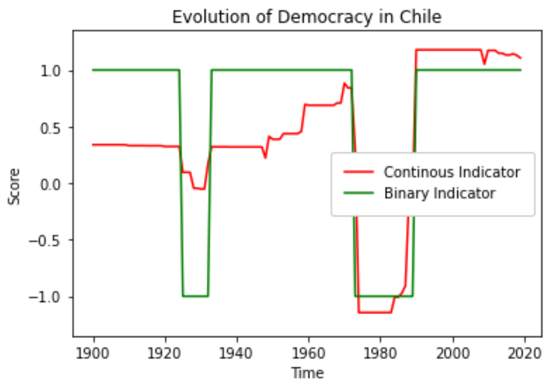


Chile

In [131…

```python
is_Chile= ContinuousIndicator.country=="Chile"
Chile=ContinuousIndicator[is_Chile]

plt.plot(Chile.year, Chile.Score, 'r', label="Continous Indicator")
plt.plot(Chile.year, Chile.PredictedLabel, 'g', label="Binary Indicator")
plt.legend(fancybox=True, framealpha=1, borderpad=1)
plt.title('Evolution of Democracy in Chile')
plt.xlabel('Time')
plt.ylabel('Score')
plt.ylim((-1.35,1.35))
```

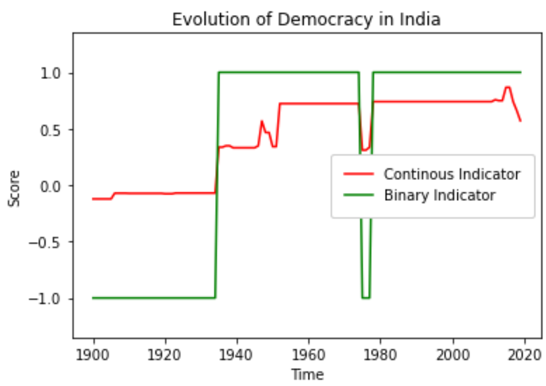


India

In [130…

```python
is_India= ContinuousIndicator.country=="India"
India=ContinuousIndicator[is_India]

plt.plot(India.year, India.Score, 'r', label="Continous Indicator")
plt.plot(India.year, India.PredictedLabel, 'g', label="Binary Indicator")
plt.legend(fancybox=True, framealpha=1, borderpad=1)
plt.title('Evolution of Democracy in India')
plt.xlabel('Time')
plt.ylabel('Score')
plt.ylim((-1.35,1.35))
```



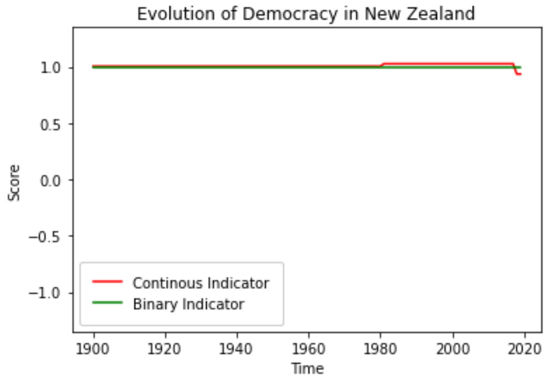
New Zealand

In [129…

```python
is_NZ= ContinuousIndicator.country=="New Zealand"
NZ=ContinuousIndicator[is_NZ]

plt.plot(NZ.year, NZ.Score, 'r', label="Continous Indicator")
plt.plot(NZ.year, NZ.PredictedLabel, 'g', label="Binary Indicator")
plt.legend(fancybox=True, framealpha=1, borderpad=1)
plt.title('Evolution of Democracy in New Zealand')
plt.xlabel('Time')
```

```
plt.ylabel('Score')
plt.ylim((-1.35,1.35))
```

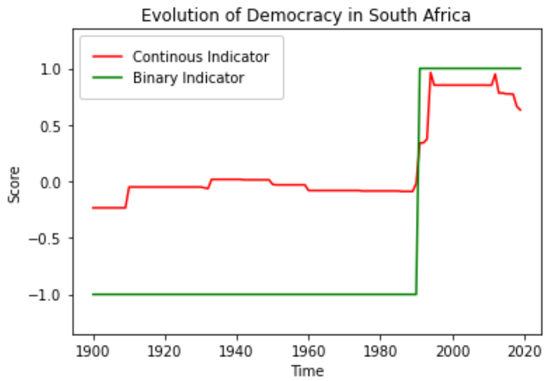


South Africa

In [128…

```
is_SA= ContinuousIndicator.country=="South Africa"
SA=ContinuousIndicator[is_SA]

plt.plot(SA.year, SA.Score, 'r', label="Continous Indicator")
plt.plot(SA.year, SA.PredictedLabel, 'g', label="Binary Indicator")
plt.legend(fancybox=True, framealpha=1, borderpad=1)
plt.title('Evolution of Democracy in South Africa')
plt.xlabel('Time')
plt.ylabel('Score')
plt.ylim((-1.35,1.35))
```

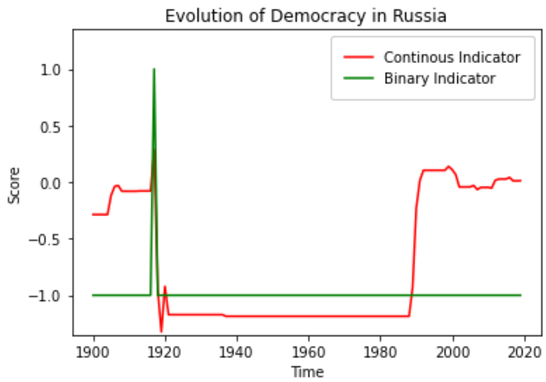


Russia

In [127…

```
is_Russia= ContinuousIndicator.country=="Russia"
Russia=ContinuousIndicator[is_Russia]

plt.plot(Russia.year, Russia.Score, 'r', label="Continous Indicator")
plt.plot(Russia.year, Russia.PredictedLabel, 'g', label="Binary Indicator")
plt.legend(fancybox=True, framealpha=1, borderpad=1)
plt.title('Evolution of Democracy in Russia')
plt.xlabel('Time')
plt.ylabel('Score')
plt.ylim((-1.35,1.35))
```



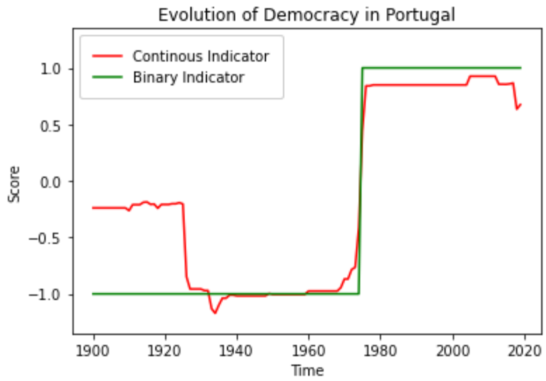
Portugal

In [133…

```
is_Portugal= ContinuousIndicator.country=="Portugal"
Portugal=ContinuousIndicator[is_Portugal]

plt.plot(Portugal.year, Portugal.Score, 'r', label="Continous Indicator")
plt.plot(Portugal.year, Portugal.PredictedLabel, 'g', label="Binary Indicator")
```

```
plt.legend(fancybox=True, framealpha=1, borderpad=1)
#plt.title='Democracy in Portugal'
plt.title('Evolution of Democracy in Portugal')
plt.xlabel('Time')
plt.ylabel('Score')
plt.ylim((-1.35,1.35))
```



United States of America

In [134…

```
is_USA= ContinuousIndicator.country=="United States of America"
USA=ContinuousIndicator[is_USA]

plt.plot(USA.year, USA.Score, 'r', label="Continous Indicator")
plt.plot(USA.year, USA.PredictedLabel, 'g', label="Binary Indicator")
plt.legend(fancybox=True, framealpha=1, borderpad=1)
plt.title('Evolution of Democracy in USA')
plt.xlabel('Time')
plt.ylabel('Score')
plt.ylim((-1.35,1.35))
```