Predicting Political Revolutions

Political upheavals have been ever present throughout humanity. Political leaders shape the context for everything we know. For that reason, we as a society need to generate a well-founded understanding of when a revolution or political change is imminent, as well as data-based indicators. This analysis seeks to forecast whether a given protest will lead to a revolution within one year using fundamental characteristics of the protest as well as metrics to understand the government in place at the time of protest.

The stakeholders for this analysis are wide reaching, but it is most directly relevant to political scientists. As a field focused on understanding the science of politics, including to inform decision making and strategy, a data science investigation into the topic of regime change remains highly relevant. Similarly, political organizers or government leadership could also gather insights with the potential to guide strategic decisions and determine where efforts are best focused and with the highest likelihood of creating an impact.

Notebook Structure

1. Data & Sources
2. Data Preparation
3. Modeling
4. Evaluation
5. Conclusion

Part I. Data & Sources

The analysis combines three core datasets from different sources to provide a distinctly unique understanding of the subject. Together, they cover the time period from 1990 to 2020 and include 17,000 protests in 167 different countries. There is a total of 131 features available, covering a wide range of topics from government checks and balances to protest location, protester violence, and protester demands. Each dataset is described in more detail below.

The Mass Mobilization Project

The first dataset is described in the source documentation as "an effort to understand citizen movements against governments, what citizens want when they demonstrate against governments, and how governments respond to citizens. The MM data cover 162 countries between 1990 and 2018. These data contain events where 50 or more protesters publicly demonstrate against government, resulting in more than 10,000 protest events. Each event records location, protest size, protester demands, and government responses." (1) The project is sponsored by the Political

Instability Task Force (PITF). The PITF is funded by the Central Intelligence Agency (CIA). (1) Throughout the analysis, this dataset is referred to as the "Protests" dataset.

Although the data source does specify that the dataset is not entirely comprehensive of all country across this entire time period, it does contain over 17,000 recorded protests, each composed of 31 features. The data span 167 countries from 1990 to 2020. Seemingly the only large country to be omitted is the United States, which is certainly not a coincidence and can undoubtedly be tied back to the source of the project funding.

Citation:

Clark, David; Regan, Patrick, 2016, "Mass Mobilization Protest Data", https://doi.org/10.7910/DVN/HTTWYL, Harvard Dataverse, V5, UNF:6:F/k8KUqKpCa5UssBbL/gzg== [fileUNF]

The Polity Project

The second dataset codes "authority characteristics of states in the world system for purposes of comparative, quantitative analysis." (2) "The Polity5 dataset covers all major, independent states in the global system over the period [1800-2020] (i.e., states with a total population of 500,000 or more in the most recent year; currently 167 countries. The Polity conceptual scheme is unique in that it examines concomitant qualities of democratic and autocratic authority in governing institutions, rather than discreet and mutually exclusive forms of governance. This perspective envisions a spectrum of governing authority that spans from fully institutionalized autocracies through mixed, or incoherent, authority regimes (termed "anocracies") to fully institutionalized democracies." (2). Most relevant to this analysis, "it also records changes in the institutionalized qualities of governing authority." (2). These changes in governing authority are the target feature of this analysis. The dataset contains 1,693 rows of data, each with 24 features. Throughout the analysis, this dataset is referred to as the "Regime Changes" or "Regimes" dataset.

Citation:

“The Polity Project.” PolityProject, Center for Systemic Peace, www.systemicpeace.org/polityproject.html.

The Database of Political Institutions

The third dataset is provided by the Inter-American Development Bank (IDB). "The Database of Political Institutions presents institutional and electoral results data such as measures of checks and balances, tenure and stability of the government, identification of party affiliation and ideology, and fragmentation of opposition and government parties in the legislature ... [it covers] about 180 countries [from] 1975-2020. It has become one of the most cited databases in comparative political economy and comparative political institutions, with more than 4,500 article citations on Google Scholar as of December 2020." (3) Within the timeframe of this analysis, the data source includes 8,200 rows of data, each with 77 features. Throughout the analysis, this dataset is referred to as the "Governments" dataset.

Citation:

Cruz, Cesi, Philip Keefer, and Carlos Scartascini. 2021. Database of Political Institutions 2020. Washington, DC: Inter-American Development Bank Research Department. https://publications.iadb.org/en/database-political-institutions-2020-dpi2020

Notes about the breakdown of the Jupyter notebooks used herein

Given the large-scale nature of the project, this notebook inevitably does not contain the entire analysis. It does not go into depth on each of the choices made for feature selection or data cleaning. Consider this the "top level" notebook, and those more detail-oriented parts of the analysis are handled in their own spaces. Here is where you should go to find more details:

1. **cleaning_protests_dataset.ipynb:** Refer to this notebook for a ground-up analysis of the "Protests" dataset. It includes important features such as the nature of each protest (size, objectives, location, etc.). Especially exhaustive and detailed data cleaning choices are made in this notebook.
2. **cleaning_regime_changes_dataset.ipynb:** Refer to this notebook for a full study of the "Regime Change" data. Primarily, this data provides the target feature for the entire analysis: regime change. It indicates when and where political change occurs. Interesting feature engineering takes place here.
3. **cleaning_governments_dataset.ipynb:** Refer to this notebook for an impressively comprehensive dataset surrounding descriptive attributes of governments around the world over many decades. Important information includes the political system of countries at specified times, the tenure of the country's primary leader, and the percentage of the popular vote the leader received (where relevant). There are descriptors as granular as the total number of seats in congress (where relevant). Extensive feature selection takes place in this notebook.

Part II. Data Preparation

This section provides a brief overview of some data preparation. Note that for the sake of readability and brevity, the majority of the actual data cleaning takes place in the above-mentioned notebooks. Please refer there for information on specific decisions made.

Import packages used throughout analysis

In [1]:

```
# Basic imporrts
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sys
from sqlalchemy import create_engine
import pickle

# Model preprocessing and processing
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import make_column_selector, make_column_transformer
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import GridSearchCV
from imblearn.pipeline import Pipeline
from sklearn.base import clone

# Models
from sklearn.dummy import DummyClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

# Performance evaluation
from sklearn.metrics import f1_score, precision_score, accuracy_score
from sklearn.metrics import plot_confusion_matrix, recall_score
import shap

# Display options
pd.options.display.max_columns = 200
%matplotlib inline

# Convenience for working with external .py files
%load_ext autoreload
%autoreload 2

# Convenience for working with external src code files
%load_ext autoreload
%autoreload 2
sys.path.insert(1, '../src')
from custom_plots import *
from create_target import *
from remove_missing_data import *
from evaluate_model_performance import *

# Global constants
RANDOM_STATE = 2021
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

Import "Regime Change" dataset

Note that the SQL database being imported has been pre-cleaned with consequential feature engineering in the *cleaning_regime_changes_dataset.ipynb* notebook in this directory.

In [2]:

```
# Import cleaned database via SQL
engine = create_engine('sqlite:///../data/processed/regime_changes.db')
with engine.begin() as connection:
    df_regimes = pd.read_sql('SELECT * FROM regime_changes', con=connection)

# Type casting
df_regimes.startdate = pd.to_datetime(df_regimes.startdate)
df_regimes.enddate = pd.to_datetime(df_regimes.enddate)
```

Import "Protests" dataset

Similar to above, the SQL database has already cleaned with some feature engineering taking place. Refer to *cleaning_protests_dataset.ipynb* in this directory for specifics.

In [3]:
```
# Import cleaned database via SQL
engine = create_engine('sqlite:///../data/processed/protests.db')
with engine.begin() as connection:
    df_protests = pd.read_sql('SELECT * FROM protests', con=connection)

# Type casting
df_protests.startdate = pd.to_datetime(df_protests.startdate)
```

Import "Governments" dataset

This SQL database has already undergone very notable, detail-oriented data cleaning in the *cleaning_governments_dataset.ipynb* notebook in this directory. Notably, it also serves the role of decreasing the number of features from 77 (in the original dataset) to 55 here. That number will again be substantially reduced in the data cleaning section below.

In [4]:
```
# Import cleaned database via SQL
engine = create_engine('sqlite:///../data/processed/governments.db')
with engine.begin() as connection:
    df_govts = pd.read_sql('SELECT * FROM governments', con=connection)

# Set index to be used in SQL join
df_govts.index = df_govts.year_scode

# Drop now-duplicated feature
df_govts.drop('year_scode', axis=1, inplace=True)
```

Join "Protests" and "Governments" datasets

By joining these two datasets, a new dataframe is created with each row corresponding to one protest, containing original features about that protest as well as the characteristics of the government in place in that country during the selected year. The two datasets are joined on the index of "*year_ofprotest*""*three-letter_country_code*", otherwise known as "*year_scode*".

In [5]:
```
# Join both dataframes
df = df_protests.join(df_govts, how='left', on='year_scode')

# Remove entries that don't have corresponding 'government' data
df.dropna(inplace=True)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15064 entries, 0 to 15207
Data columns (total 76 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   country                         15064 non-null  object
 1   scode                           15064 non-null  object
 2   region                          15064 non-null  object
 3   protestnumber                   15064 non-null  int64
 4   protesterviolence               15064 non-null  int64
 5   startdate                       15064 non-null  datetime64[ns]
 6   duration_days                   15064 non-null  int64
 7   participants                    15064 non-null  int64
 8   participants_category           15064 non-null  object
```

```
9   demand_labor-wage-dispute          15064 non-null  int64
10  demand_land-farm-issue             15064 non-null  int64
11  demand_police-brutality            15064 non-null  int64
12  demand_political-behavior/process  15064 non-null  int64
13  demand_price-increases/tax-policy  15064 non-null  int64
14  demand_removal-of-politician       15064 non-null  int64
15  demand_social-restrictions         15064 non-null  int64
16  year_scode                         15064 non-null  object
17  participants_log                   15064 non-null  float64
18  duration_days_log                  15064 non-null  float64
19  protestnumber_log                  15064 non-null  float64
20  system                             15064 non-null  object
21  yrsoffc                            15064 non-null  float64
22  finittrm                           15064 non-null  float64
23  yrcurnt                            15064 non-null  float64
24  termlimit                          15064 non-null  float64
25  reelect                            15064 non-null  float64
26  multpl                             15064 non-null  float64
27  military                           15064 non-null  float64
28  defmin                             15064 non-null  float64
29  prtyin                             15064 non-null  float64
30  execrlc                            15064 non-null  object
31  execnat                            15064 non-null  float64
32  execrel                            15064 non-null  object
33  execage                            15064 non-null  float64
34  allhouse                           15064 non-null  float64
35  totalseats                         15064 non-null  float64
36  oppmajh                            15064 non-null  float64
37  oppmajs                            15064 non-null  float64
38  legelec                            15064 non-null  float64
39  exelec                             15064 non-null  float64
40  liec                               15064 non-null  float64
41  eiec                               15064 non-null  float64
42  mdmh                               15064 non-null  float64
43  mdms                               15064 non-null  float64
44  ssh                                15064 non-null  float64
45  pluralty                           15064 non-null  float64
46  pr                                 15064 non-null  float64
47  housesys                           15064 non-null  object
48  sensys                             15064 non-null  object
49  thresh                             15064 non-null  float64
50  cl                                 15064 non-null  float64
51  gq                                 15064 non-null  float64
52  gqi                                15064 non-null  float64
53  fraud                              15064 non-null  object
54  auton                              15064 non-null  float64
55  muni                               15064 non-null  float64
56  state                              15064 non-null  float64
57  author                             15064 non-null  float64
58  numvote                            15064 non-null  float64
59  oppvote                            15064 non-null  float64
60  maj                                15064 non-null  float64
61  partyage                           15064 non-null  float64
62  herfgov                            15064 non-null  float64
63  herfopp                            15064 non-null  float64
64  frac                               15064 non-null  float64
65  oppfrac                            15064 non-null  float64
66  govfrac                            15064 non-null  float64
67  tensys_strict                      15064 non-null  float64
68  checks                             15064 non-null  float64
69  stabs_strict                       15064 non-null  float64
70  tenlong_strict                     15064 non-null  float64
71  tenshort_strict                    15064 non-null  float64
72  polariz                            15064 non-null  float64
73  country_govt                       15064 non-null  object
```

```
 74  scode_govt                          15064 non-null  object
 75  percent                             15064 non-null  float64
dtypes: datetime64[ns](1), float64(51), int64(11), object(13)
memory usage: 8.8+ MB
```

Identify countries that are missing from "Regime Changes" dataset

In [6]:

```
# All countries in union of Protests and Governments
all_countries = df.scode.unique()

# All countries in Regimes
regime_countries = df_regimes.scode.unique()


# Loop over all_countries
missing = []
for country in all_countries:
    # Make note of any countries not in Regimes

    if country not in regime_countries:
        missing.append(country)

print('Countries missing from "Regimes" dataset:', missing)

# Remove these countries from dataset
scodes_to_remove_ind = [x in missing for x in df.scode]
df.drop(df.loc[scodes_to_remove_ind].index, axis=0, inplace=True)
```

```
Countries missing from "Regimes" dataset: ['LUX']
```

Join with "Regime Changes" dataset

Add the third and final dataset, which includes the "target" feature of the analysis.

In [7]:

```
target = create_target(df, df_regimes)
df = pd.concat([df, target], axis=1)
```

Basic cleaning

Start by running custom function that removes all features that don't have a minimum threshold of non-null values

In [8]:

```
# Remove entries with limited "government" data
df = remove_missing_data(df, 1000)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11840 entries, 0 to 15060
Data columns (total 54 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   index                               11840 non-null  float64
 1   country                             11840 non-null  object
 2   scode                               11840 non-null  object
 3   region                              11840 non-null  object
 4   protestnumber                       11840 non-null  float64
 5   protesterviolence                   11840 non-null  float64
 6   startdate                           11840 non-null  datetime64[ns]
```

```
 7   duration_days                       11840 non-null  float64
 8   participants                        11840 non-null  float64
 9   participants_category               11840 non-null  object
 10  demand_labor-wage-dispute           11840 non-null  float64
 11  demand_land-farm-issue              11840 non-null  float64
 12  demand_police-brutality             11840 non-null  float64
 13  demand_political-behavior/process   11840 non-null  float64
 14  demand_price-increases/tax-policy   11840 non-null  float64
 15  demand_removal-of-politician        11840 non-null  float64
 16  demand_social-restrictions          11840 non-null  float64
 17  year_scode                          11840 non-null  object
 18  participants_log                    11840 non-null  float64
 19  duration_days_log                   11840 non-null  float64
 20  protestnumber_log                   11840 non-null  float64
 21  system                              11840 non-null  object
 22  yrsoffc                             11840 non-null  float64
 23  finittrm                            11840 non-null  float64
 24  termlimit                           11840 non-null  float64
 25  military                            11840 non-null  float64
 26  defmin                              11840 non-null  float64
 27  execnat                             11840 non-null  float64
 28  execrel                             11840 non-null  object
 29  totalseats                          11840 non-null  float64
 30  oppmajh                             11840 non-null  float64
 31  legelec                             11840 non-null  float64
 32  exelec                              11840 non-null  float64
 33  liec                                11840 non-null  float64
 34  eiec                                11840 non-null  float64
 35  gq                                  11840 non-null  float64
 36  gqi                                 11840 non-null  float64
 37  auton                               11840 non-null  float64
 38  numvote                             11840 non-null  float64
 39  oppvote                             11840 non-null  float64
 40  maj                                 11840 non-null  float64
 41  herfgov                             11840 non-null  float64
 42  govfrac                             11840 non-null  float64
 43  tensys_strict                       11840 non-null  float64
 44  checks                              11840 non-null  float64
 45  stabs_strict                        11840 non-null  float64
 46  tenlong_strict                      11840 non-null  float64
 47  tenshort_strict                     11840 non-null  float64
 48  country_govt                        11840 non-null  object
 49  scode_govt                          11840 non-null  object
 50  xconst                              11840 non-null  float64
 51  present                             11840 non-null  float64
 52  next_regime_chg_date                11840 non-null  datetime64[ns]
 53  days_until_next_regime_chg          11840 non-null  float64
dtypes: datetime64[ns](2), float64(43), object(9)
memory usage: 5.0+ MB
```

In [9]:

```
# Convert startdate to a float instead of datetime since datetime cannot be
# handled by models but fractional years can
df['startdate'] = df.startdate.dt.year + \
                  df.startdate.dt.month/12 + \
                  df.startdate.dt.day/365
```

In [10]:

```
# Type casting
df['region'] = df.region.astype('category')
df['system'] = df.system.astype('category')
df['country'] = df.country.astype('category')
df['xconst'] = df['xconst'].astype('int')
```

Fill "placeholder" values in *xconst* with continuous numbers. See Model.ipynb for further details.

In [11]:
```
df.xconst.replace(-66.0, -1, inplace=True) # 'Interruption periods'
df.xconst.replace(-77.0, -2, inplace=True) # 'Interregnum periods'
df.xconst.replace(-88.0, 0, inplace=True) # 'Transition periods'
```

Define target

This allows the user to define the target in terms of the amount of time before which a regime transition will occur. For this analysis, it uses one year, but other values have also been explored with similar results.

In [12]:
```
DAYS_UNTIL_CHG = 365
target = pd.DataFrame(df['days_until_next_regime_chg'] < DAYS_UNTIL_CHG)
target = target.astype('int')
target.columns = ['target']
```

Drop unused columns

Refer to "model.ipynb" notebook for decision-making details. Specific descisions will not be discussed here as they are covered in depth in the previously mentioned file.

In [13]:
```
drop_cols = ['year_scode', 'scode_govt', 'country_govt', 'startdate',
             'days_until_next_regime_chg', 'scode', 'participants_category', 'participa
             'next_regime_chg_date', 'index', 'duration_days', 'present', 'protestnumbe
df.drop(drop_cols, axis=1, inplace=True)


# Features with high collinearity
colinear_cols = ['liec', 'eiec', 'tenlong_strict', 'tenshort_strict', 'finittrm', 'govf
df.drop(colinear_cols, axis=1, inplace=True)
```

Feature descriptions

Given the large number of features with abbreviated names, those that are used within the model are defined below.

Protest Details - as provided by *Protests* dataset:

- **country:** location of protest
- **region:** global region
- **protesterviolence:** (y/n) indicator of protester violence
- **participants_log:** number of individuals in attendance, log-transformed
- **duration_days_log:** duration of the protest, log-transformed
- **protestnumber_log:** protest number in the selected country in a given year, log-transformed

---

Categorizations of the reason for a protest (multiple options possible) - as provided by *Protests* dataset:

- **labor wage dispute**

- **land farm issue**
- **police brutality**
- **political behavior/process**
- **price increases/tax policy**
- **removal of politician**
- **social restrictions**

Government descriptors in the given country at the time protest - as provided by *Government Characterization* dataset. Strict, quantifiable definitions are provided in Data Manual.

- **system:** governmental system (Parliamentary, Assembly-elected president, Presidential, Unelected)
- **yrsoffc:** number of years the chief executive has been in office
- **termlimit:** (y/n) is there a term limit in place?
- **military:** (y/n) is the chief executive a military officer?
- **defmin:** (y/n) is defense minister a military officer?
- **execnat:** (y/n) is executive a nationalist?
- **execrel:** executive religion
- **totalseats:** total seats in the legislature
- **oppmajh:** (y/n) does one opposition party have an absolute majority in House?
- **legelec:** (y/n) is there a legislative election in the protest year?
- **exelec:** (y/n) is there an executive election in the protest year?
- **gq:** are there gender quotas in government (None, Voluntary, Reserved, Required)
- **gqi:** (y/n) is gender quota actually implemented
- **auton:** are there autonomous regions? (federalism)
- **numvote:** total vote share of governmental leadership party
- **maj:** margin of majority
- **herfgov:** Herfindahl index government (sum of squared seat shares of all parties in the government)
- **tensys_strict:** length of time the country has been autocratic or democratic
- **checks:** checks and balances
- **stabs_strict:** stability, as measured by the percent of veto players who *drop* from the government in a given year

---

Political regime authority characteristics - as provided by *Regime Changes* dataset. Strict, quantifiable definitions are provided in Data Manual.

- **xconst:** executive constraints / decision rules (Unlimited Authority through Executive Parity)

In [14]:
```
display(df.info())
display(df.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11840 entries, 0 to 15060
Data columns (total 34 columns):
 #   Column                            Non-Null Count  Dtype
```

```
---  ------                              --------------  -----
 0   country                             11840 non-null  category
 1   region                              11840 non-null  category
 2   protesterviolence                   11840 non-null  float64
 3   demand_labor-wage-dispute           11840 non-null  float64
 4   demand_land-farm-issue              11840 non-null  float64
 5   demand_police-brutality             11840 non-null  float64
 6   demand_political-behavior/process   11840 non-null  float64
 7   demand_price-increases/tax-policy   11840 non-null  float64
 8   demand_removal-of-politician        11840 non-null  float64
 9   demand_social-restrictions          11840 non-null  float64
 10  participants_log                    11840 non-null  float64
 11  duration_days_log                   11840 non-null  float64
 12  protestnumber_log                   11840 non-null  float64
 13  system                              11840 non-null  category
 14  yrsoffc                             11840 non-null  float64
 15  termlimit                           11840 non-null  float64
 16  military                            11840 non-null  float64
 17  defmin                              11840 non-null  float64
 18  execnat                             11840 non-null  float64
 19  execrel                             11840 non-null  object
 20  totalseats                          11840 non-null  float64
 21  oppmajh                             11840 non-null  float64
 22  legelec                             11840 non-null  float64
 23  exelec                              11840 non-null  float64
 24  gq                                  11840 non-null  float64
 25  gqi                                 11840 non-null  float64
 26  auton                               11840 non-null  float64
 27  numvote                             11840 non-null  float64
 28  maj                                 11840 non-null  float64
 29  herfgov                             11840 non-null  float64
 30  tensys_strict                       11840 non-null  float64
 31  checks                              11840 non-null  float64
 32  stabs_strict                        11840 non-null  float64
 33  xconst                              11840 non-null  int32
dtypes: category(3), float64(29), int32(1), object(1)
memory usage: 2.9+ MB
None
```

	protesterviolence	demand_labor-wage-dispute	demand_land-farm-issue	demand_police-brutality	demand_political-behavior/process	demand_pri increases/t pol
count	11840.000000	11840.00000	11840.000000	11840.000000	11840.000000	11840.000(
mean	0.270861	0.14848	0.042821	0.072973	0.695861	0.095
std	0.444423	0.35559	0.202462	0.260103	0.460061	0.294
min	0.000000	0.00000	0.000000	0.000000	0.000000	0.000(
25%	0.000000	0.00000	0.000000	0.000000	0.000000	0.000(
50%	0.000000	0.00000	0.000000	0.000000	1.000000	0.000(
75%	1.000000	0.00000	0.000000	0.000000	1.000000	0.000(
max	1.000000	1.00000	1.000000	1.000000	1.000000	1.000(

Part III. Modeling

Given the cleaned and aggregated dataset above, the next section moves into the Modeling phase. It uses the following general structure:

1. Test, train, and validation splits
2. "Dummy" model for baseline performance metric
3. Logistic regression model
4. Random forest model
5. XG boost model

Note that in the *MODEL.ipynb* notebook, alternative models are explored, such as K-Nearest Neighbors (KNN), Bayesian classifiers, ADA boost, and Decision Trees.

Each type of model is constructed using elements of encoding, scaling, resampling and hyperparameter optimization.

- One hot encoding was essential given the categorical type of some features.
- Standard scaling was essential given the vast array of different numerical feature distributions and ranges. Min-max scaling was tested but proved less effective.
- SMOTE was determined to be essential given the imbalanced nature of the dataset. Only 11% of the target feature values were 1, leaving the other 89% as 0. This is a prime example of the need for resampling, and SMOTE proved highly effective.
- Hyperparameter grid searches are inherently valuable when optimizing a model. Appropriate hyperparameter searches were used for each model type.

The performance of each model is evaluated on four core statistical measures (f1 score, accuracy, precision, and recall), in addition to displaying a confusion matrix for the test data. F1 was selected before the modeling process as the most relevant metric given that it encompasses all possible outcomes, as opposed to the other three metrics which leave out at least one possible outcome from their evaluation.

Lastly, the grid search is not actually completed in this file. The code shows the way in which the pipelines were established as well as the data to be used. However, in order to save the time of a thorough grid search twice, the calculations are completed in the MODEL file and exported via a Pickle file. They are then imported here after the grid search is complete.

Create train-test split

In [15]:
```
# Create standard train-test splits
x_train, x_test, y_train, y_test = train_test_split(df, target,
                                                    random_state=RANDOM_STATE,
                                                    test_size=0.3)
```

Create baseline "dummy" model

In [16]:
```
# Create instance of dummy classifier and fit to training data
dummy_clf = DummyClassifier(strategy='stratified')
dummy_clf.fit(x_train, y_train)

# Create model predictions for test data
```

```
pred_dummy = dummy_clf.predict(x_test)
print_scores(pred_dummy, y_test)
```

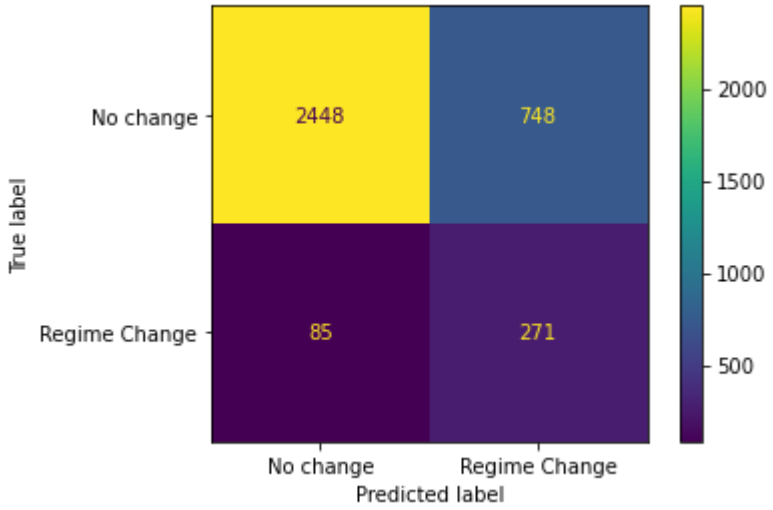
```
- f1: 0.10582010582010581
- accuracy: 0.8096846846846847
- precision: 0.1
- recall: 0.11235955056179775
```

Create Logistic Regression classifier model

In [17]:

```
# Instantiate model
model_log = LogisticRegression(max_iter=5000)

# Add one-hot encoding for categoricals
ohe = OneHotEncoder(handle_unknown='ignore')

# Add standard scaling
scaler = StandardScaler()

# Add resampling to address class imbalance
smote = SMOTE(random_state=RANDOM_STATE)


# Select object types for one-hot encoding and numeric types for scaling
selector_object = make_column_selector(dtype_exclude='number')
selector_numeric = make_column_selector(dtype_include='number')

# Create transformer for the selectors/encoders/scalers
transformer = make_column_transformer((ohe, selector_object),
                                      (scaler, selector_numeric))


# Create pipeline for transformer, resampling, and model
pipe = Pipeline([('transformer', transformer),
                 ('smote', smote),
                 ('model', model_log)])

# Run grid search to optimizer hyperparameters
grid_log = {'model__C': np.logspace(-1, 5, 20)}
grid_search = GridSearchCV(pipe, grid_log, scoring='f1', cv=10)

# Fit model to training data
# grid_search.fit(x_train, y_train)

# As described in this section's header, the fitted pipeline is
# simply imported via Pickle from the MODEL notebook, where it was
# fitted on the same data.
with open('../data/processed/model_logreg.pickle', 'rb') as f:
    model_logreg = pickle.load(f)


# Fit pipeline
model_logreg.fit(x_train, y_train)

# Print metrics
prediction = model_logreg.predict(x_test)
print_scores(prediction, y_test)

# Display confusion matrix
labels = ['No change', 'Regime Change']
plot_confusion_matrix(model_logreg, x_test, y_test, display_labels=labels);
```

```
A column-vector y was passed when a 1d array was expected. Please change the shape of y
to (n_samples, ), for example using ravel().
- f1: 0.3941818181818182
- accuracy: 0.7654842342342343
- precision: 0.26594700686947986
- recall: 0.7612359550561798
```



Overall, the Logistic Regression model performance is underwhelming, albeit substantially better than the baseline model: an F1 score performance of 0.32 instead of the baseline 0.07. Still, it seems that the model does not capture the complexity of the data. Below more complex models are explored.

Create Random Forest model

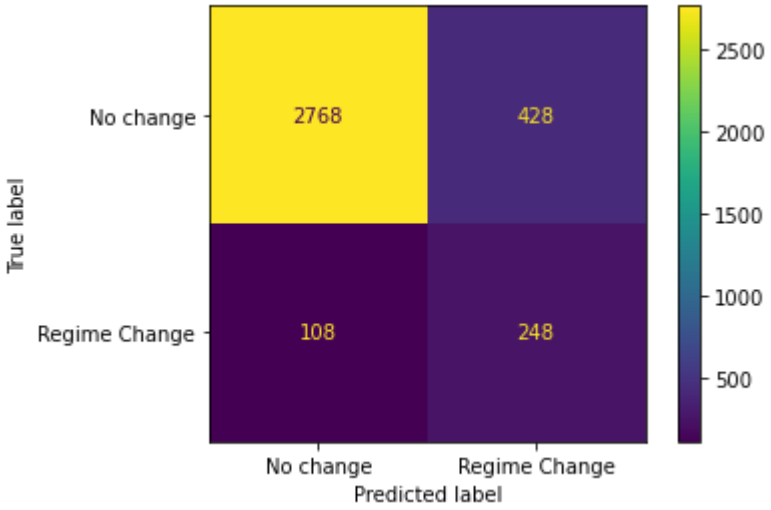
Reuse one-hot encoder, scaler, resampler, selectors, transformer from logistic regression.

In [18]:

```
# Instantiate model
model_rf = RandomForestClassifier()

# Create pipeline for transformer, resampling, and model
pipe = Pipeline([('transformer', transformer),
                 ('smote', smote),
                 ('model', model_rf)])

# Run grid search to optimizer hyperparameters
grid_rf = {
    'model__n_estimators': [25, 75, 150],
    'model__criterion': ['gini', 'entropy'],
    'model__max_depth': [3, 6, 10],
    'model__min_samples_split': [5, 10],
    'model__min_samples_leaf': [3, 6]}

# Create instance of grid search
grid_search = GridSearchCV(pipe, grid_rf, scoring='f1', cv=5)

# Fit model to training data
# grid_search.fit(x_train, y_train)

# As described in this section's header, the fitted pipeline is
# simply imported via Pickle from the MODEL notebook, where it was
# fitted on the same data.
```

```
with open('../data/processed/model_rf.pickle', 'rb') as f:
    model_rf = pickle.load(f)

# Fit pipeline
model_rf.fit(x_train, y_train)

# Print metrics
prediction = model_rf.predict(x_test)
print_scores(prediction, y_test)

# Display confusion matrix
plot_confusion_matrix(model_rf, x_test, y_test, display_labels=labels);
```

```
A column-vector y was passed when a 1d array was expected. Please change the shape of y
to (n_samples,), for example using ravel().
- f1: 0.4806201550387597
- accuracy: 0.8490990990990991
- precision: 0.3668639053254438
- recall: 0.6966292134831461
```



The random forest model shows substantial performance over the logistic regression model. Specifically, the F1 score increased by 0.12, and other metrics see similar improvements. Still, the performance isn't perfect, so the below model explores an increasingly complex model that is able to better capture the complexity of the data.

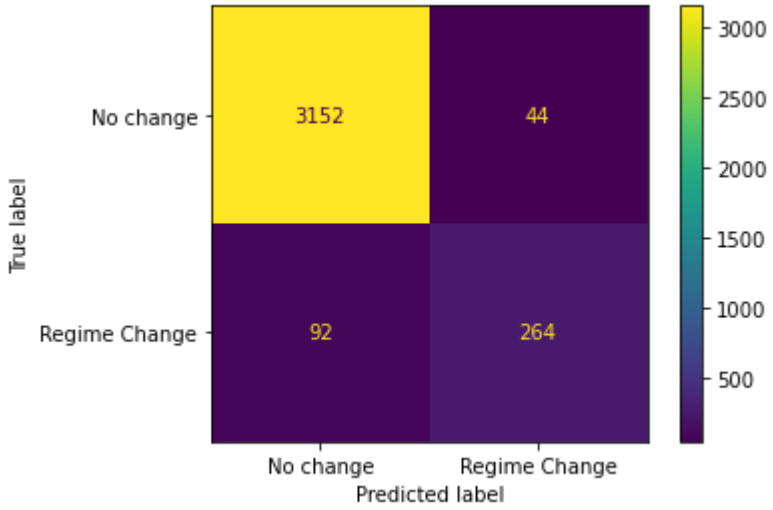
XG Boost model

In [19]:

```
# Instantiate model
model_xgb = XGBClassifier(eval_metric='logloss', use_label_encoder=False,
                          random_state=RANDOM_STATE)

# Create pipeline for transformer, resampling, and model
pipe = Pipeline([('transformer', transformer),
                 ('smote', smote),
                 ('model', model_xgb)])


# Run grid search to optimizer hyperparameters
grid_xgb = {
    'model__learning_rate': [0.01, 0.1, 0.25],
    'model__max_depth': [6, 8, 10, 12],
    'model__subsample': [0.4, 0.7, 1],
```

```
    'model__n_estimators': [100, 200, 300, 400]}

# Fit model to training data
# grid_search.fit(x_train, y_train)

# As described in this section's header, the fitted pipeline is
# simply imported via Pickle from the MODEL notebook, where it was
# fitted on the same data.
with open('../data/processed/model_xgb.pickle', 'rb') as f:
    model_xgb = pickle.load(f)

# Fit pipeline
model_xgb.fit(x_train, y_train)

# Print metrics
prediction = model_xgb.predict(x_test)
print_scores(prediction, y_test)

# Display confusion matrix
plot_confusion_matrix(model_xgb, x_test, y_test, display_labels=labels);
```

```
- f1: 0.7951807228915663
- accuracy: 0.9617117117117117
- precision: 0.8571428571428571
- recall: 0.7415730337078652
```



Show optimal model hyperparameters

In [20]:

```
print(model_xgb.steps[2])
```

```
('model', XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, eval_metric='logloss',
              gamma=0, gpu_id=-1, importance_type='gain',
              interaction_constraints='', learning_rate=0.5, max_delta_step=0,
              max_depth=8, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=300, n_jobs=8,
              num_parallel_tree=1, random_state=2021, reg_alpha=0, reg_lambda=1,
              scale_pos_weight=1, subsample=1, tree_method='exact',
              use_label_encoder=False, validate_parameters=1, verbosity=None))
```

The model performance does increase using the XG Boost model relative to all other previously tested models

All performance metrics improve significantly over the Random Forest model.

Part IV. Evaluation

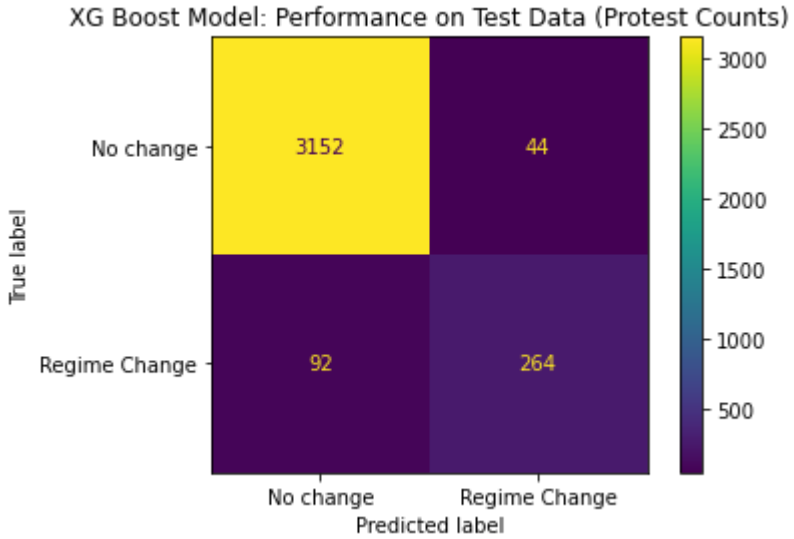
As can be seen based on the F1 scores – the metric chosen at the beginning of the analysis as most relevant – the XG boost model performs the best. This section will test the final model on the test data as well as visualize feature significance to the extent feasible.

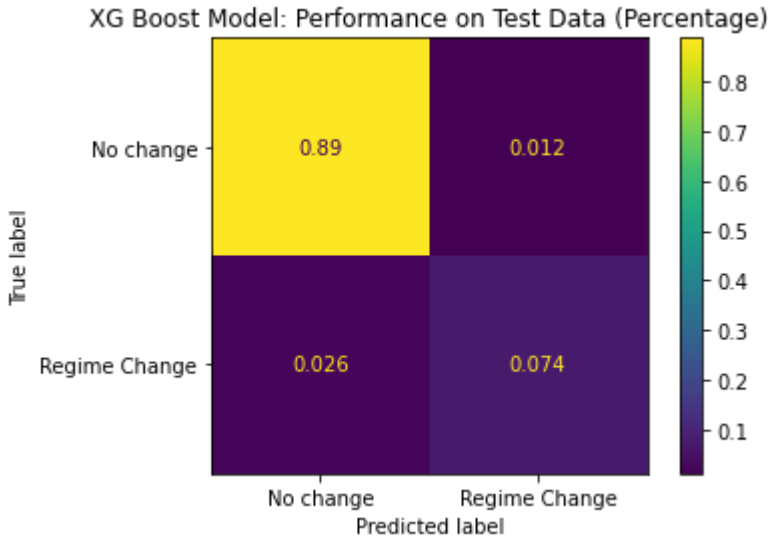
Evaluate performance on test data

The below metrics show strong model performance on the test data with nearly no noteworthy indication of overfitting

In [21]:
```
# Calculate predicted output of test data
prediction = model_xgb.predict(x_test)

# Print performance metrics
print_scores(prediction, y_test)

# Display confusion matrix (not normalized)
plot_confusion_matrix(model_xgb, x_test, y_test, display_labels=labels)
plt.title('XG Boost Model: Performance on Test Data (Protest Counts)')
plt.show()

# Display confusion matrix (normalized)
plot_confusion_matrix(model_xgb, x_test, y_test, normalize='all', display_labels=labels
plt.title('XG Boost Model: Performance on Test Data (Percentage)');
```

```
- f1: 0.7951807228915663
- accuracy: 0.9617117117117117
- precision: 0.8571428571428571
- recall: 0.7415730337078652
```



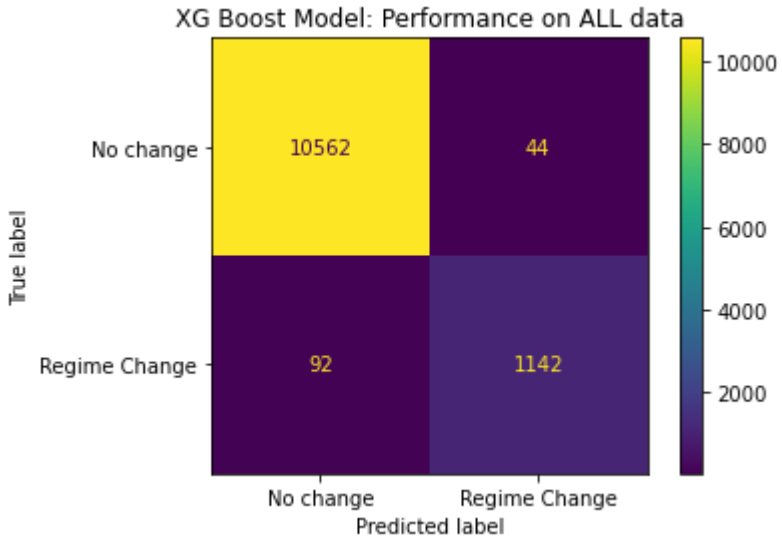


To identify overfitting, test the final model on the full dataset (train *and* test)

Although these metrics aren't used to state overall performance, they can be interesting and can help identify overfitting if the performance on *all* data is substantially higher than the performance on *test* data. The below printouts show that overfitting is significant, though the performance on the test data remains sufficiently strong for a high performing model.

In [22]:
```
# Calculate predicted output of *all* data
prediction = model_xgb.predict(df)

# Print performance metrics
print_scores(prediction, target)

# Display confusion matrix
plot_confusion_matrix(model_xgb, df, target, display_labels=labels)
plt.title('XG Boost Model: Performance on ALL data');
```

```
- f1: 0.943801652892562
- accuracy: 0.9885135135135135
- precision: 0.9629005059021922
- recall: 0.9254457050243112
```



Investigate Feature Importance

Here, we see that the five most significant features are:

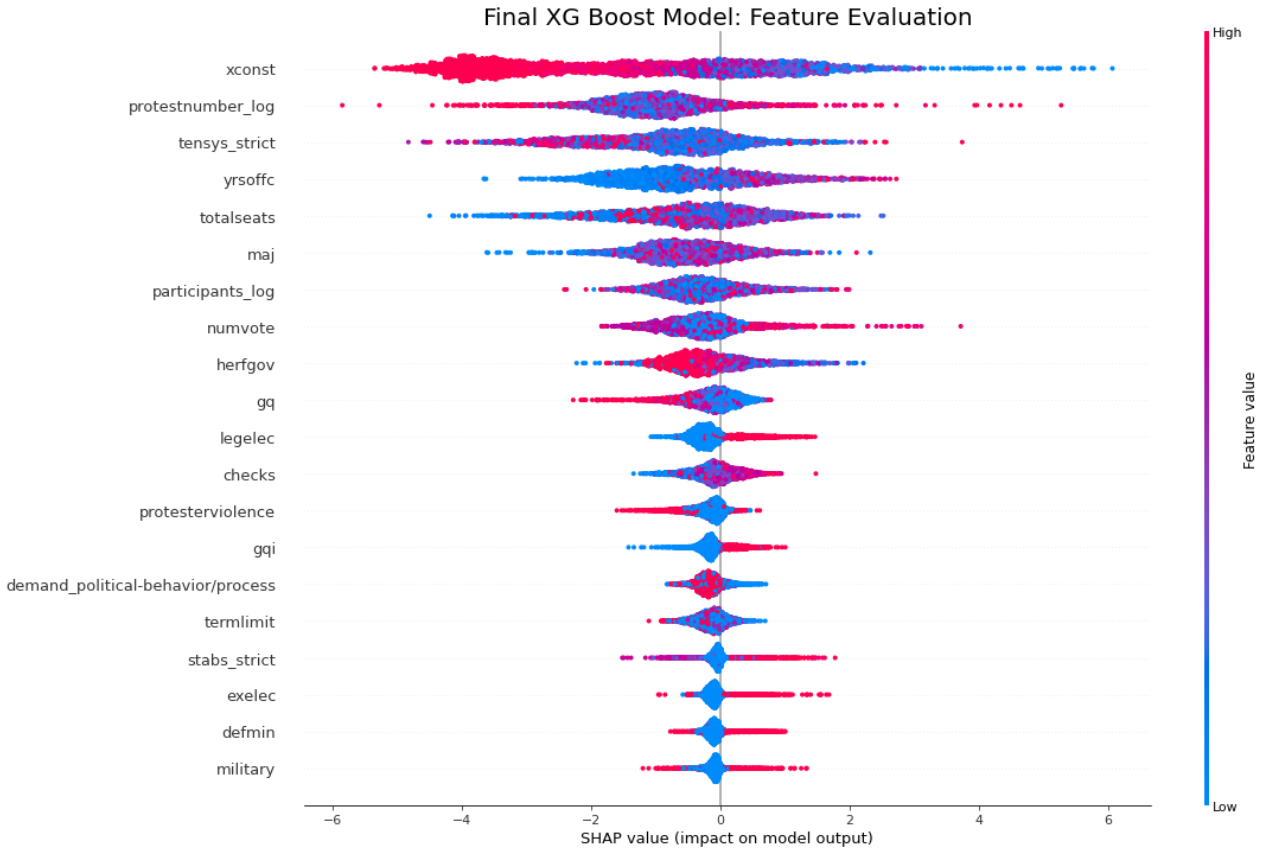
1. **xconst**: presence of executive constraints, ranging from "Unlimited Authority" through "Executive Parity"
2. **protestnumber_log**: number of protests that already occurred in the year of the protest, log-transformed
3. **tensys_strict**: length of time the country has been autocratic or democratic
4. **totalseats**: total seats in the legislature
5. **yrsoffc**: number of years the chief executive has been in office

To bring some meaning to the SHAP summary plot, it is worth noting that:

1. Countries with low executive constraint ("Unlimited Authority") are more strongly correlated with a protest overturning the regime.
2. Higher turnouts to protests are associated with regime transitions. However, the data is somewhat split: even very small protests can be associated with regime transition.
3. There is a less distinct divide in the relationship between the length of time a country has been autocratic vs. democratic than in other metrics. That said, the feature remains a strong predictor.
4. Regime transitions happen somewhat consistently across the size of governing bodies.
5. Protests in countries with new executive leadership are less likely to lead to regime change than countries with a long-ruling leader.

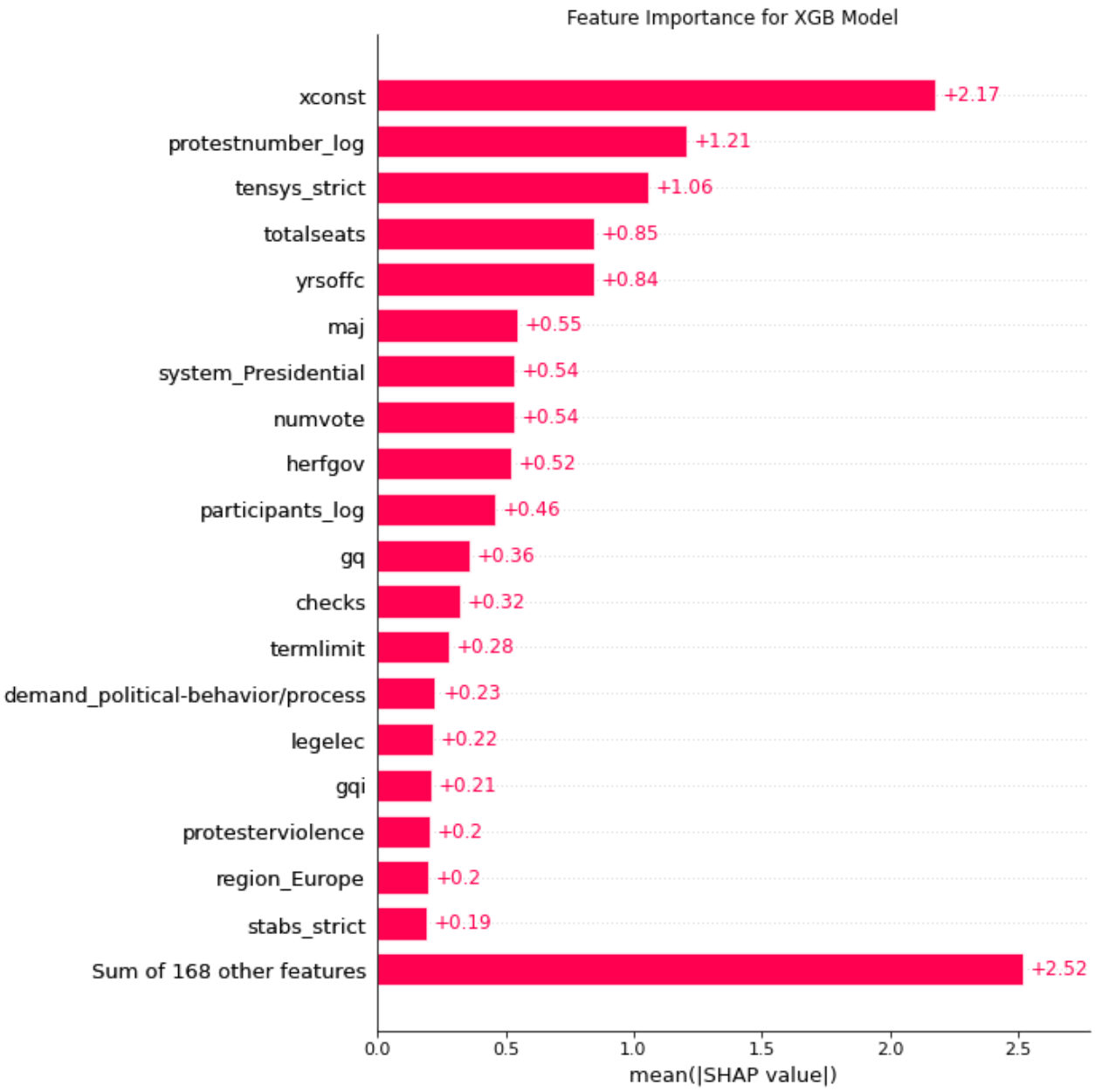
In [23]:

```
# SHAP summary plot for XGB
produce_shap_plot(x_train, y_train, x_test, y_test, clone(model_xgb),
                  title='Final XG Boost Model: Feature Evaluation');
```



In [24]:

```
# SHAP bar plot for XGB model
x_tr_manual, y_tr_manual, x_te_manual = get_shap_df(x_train, y_train, x_test)
model = model_xgb.steps[2][1]

# Calculate SHAP values
explainer = shap.Explainer(model)
shap_values = explainer(x_te_manual)

# SHAP bar plot for XGB model
plt.title('Feature Importance for XGB Model')
shap.plots.bar(shap_values, max_display=20)
```



Part V. Conclusion

Overall, this analysis successfully completes its objective. It creates and tunes a model that helps predict whether a given protest will lead to a regime transition within one year. This incredibly valuable tool can be used by political scientists to better understand regime transitions, including my investigating the model further than was conducted in this analysis. In addition, the model and its findings can be used for proactive or preventative measures by either side of government. With a very strong performing model, it can be trusted to give an accurate estimate of changes to come.

Going forward, this project allows for easy growth as more data is released. Each of the three primary datasets receive regular updates, and this new information can easily be incorporated in order to expand the temporal scope of the project and with more data comes to potential for stronger performance.

Next Steps

A powerful next step could be to investigate this data using time series analyses. Specifically, it would be valuable to understand how protest outcomes are affected by protests before it.