Brooke Ann Coco

Project 3 Summary

To Coup Or Not To Coup?

May 10, 2019

**Motivation:**

While many are fortunate enough to live in a nation with a stable government that (generally) works for its people, recent news headlines from places such as Egypt, Turkey, or Venezuela, make clear that not everyone is so lucky. Such circumstances can lead to extreme, often violent actions to overthrow the regime in power. However, imagine if we could identify nations most at risk of coup attempts, then intervene before such catastrophic events ever occur. To this end, the goal of this project is to utilize supervised learning techniques to classify whether a sovereign nation will experience a coup attempt within the next year.

**Background:**

The theory underlying this project is based off of a 2011 book, *The Dictators Handbook: Why Bad Behavior Is Almost Always Good Politics,* where authors Bruce Bueno de Mesquita and Alastair Smith argue that the ultimate motivation of any ruler is to remain in power. Thus, all rulers (democrats to autocrats alike) are bound by the same set of political rules. Leaders are beholden to those key supporters who enable a leader’s power. Therefore, leaders who rely on a large percentage of the population for power (like democracies) often appeal to the masses through social benefits and public investments. On the other hand, in small winning coalition governments, like autocracies, leaders need only please that narrow fraction of the population who enable their power, often via private investment. As a result, we tend to see much higher levels of oppression and corruption in nations with small winning coalition governments. Assuming that this theory of politics is true, we would expect a coup to occur when leaders fail to keep their key supporters happy.

**Data:**

The data for this project was obtained from several sources, such as OEF Research’s REIGN (Rulers, Elections, and Irregular Governance) database, which records monthly updates on the political conditions of over 200 sovereign nations. From this source, I obtained information such as when and where coup attempts occurred, whether a leader’s legitimacy stemmed from a career in the military, the number of years a leader has been in power, and whether an executive election is anticipated within the next few months.

Next, I gathered several economic and developmental metrics from the World Bank’s World Development Indicators database, the most comprehensive and accurate global development database compiled from credible international sources. From here, I obtained information such as life expectancy and the percentage of the population with access to basic resources such as food, water, sanitation, electricity, healthcare, and education. I also collected information about government finances, such as tax revenues, overall revenue, reserves, natural resources rents, military spending, and debt forgiveness from World Development Indicators database. Finally, I looked to the Penn World Table for metrics such as expenditure-side real GDP and human capital index, which is based on years of schooling and returns to education.

Due to the scarcity of historic global development data, I decided to limit my observations years to 1990 to 2016, leaving me with a total of 5,138 observations.

**Pre-Processing:**

Before I could begin modeling, several steps had to be taken to prepare my data. For one, because the World Development Indicators database was aggregated by month, I first needed to take steps to aggregate this data by year so as to make it compatible with the Penn World Table and the REIGN database. Next, despite limiting my observations to data spanning from 1990 to 2016, my data remained sparse. For instance, 8 of my 23 variables were missing over half of the data. As a result, I decided to impute the missing values with their global mean, allowing me to utilize even my most sparse variables. Finally, my target variable was high imbalanced, with coup attempts representing only about 2% of the data. Thus, I utilized several oversampling methods, including resampling, SMOTE, and ADSYN.

**Modeling:**

Once my data was ready to be processed, I utilized Python’s scikit-learn module to train a suite of supervised learning models including K-Nearest Neighbors, Logistic Regression, Decision Trees, Random Forests, Gradient Boosting, and Bagging. The Area Under the Curve (AUC) was used as the evaluation metric for each iteration of the above models. I employed Scikit-learn’s GridSearchCV to tune each model’s parameters with the goal of optimizing the AUC.

Throughout the process, I began to notice that my AUC metrics were jumping above 0.99. This came as highly suspicious, as it is highly improbable that my features were able to capture the complexity behind why people decide to coup. This prompted me to look more closely at my data, which led me to discover two crucial errors in my data. First, though I had scaled some half engineered features that were buried in an obsolete section of my notebook, I had failed to scale the data that was being fed into the models. Second, I had leaky variables. Though I had considered the fact that I should ensure that my model was not privy to future data that may be the result of the coup, I had not taken steps to account for this in my data.

After taking steps to account for the data leakage, I continued the process of parameter tuning. While techniques such as logistic regression and decision trees were relatively quick to compute, others like KNN and random forests were taking an absurdly long time to complete. Because I had since taken some extra steps to optimize the recording of each model result, I did not pay close attention to the incoming AUC metrics. As a result, it was not until it came down to evaluate which was the best model, that I realized that I was still receiving unreasonably high AUC scores, indicating that there was still an error in my data.

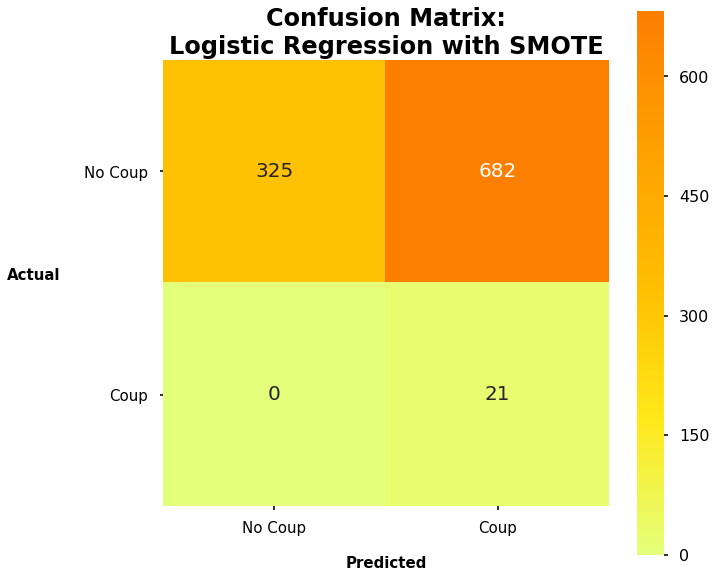
Upon closer inspection, it came to my attention that there was still data leakage to account for. Though I had had correctly split my data into train and test sets before oversampling the training data, I had not accounted for the fact that scikit learn’s GridSearchCV performs its own cross-validation. Thus, in feeding scikit learn’s GridSearchCV oversampled training data, my models were given some of the same data points to train on as they were to test. In short, the models were memorizing for the test.

**Results:**

After accounting for yet another instance of data leakage, I finally started to receive more reasonable AUC scores. In the end, a logistic regression model that was fed training data oversampled via SMOTE received the highest AUC metric on cross-validation, with a score of 0.83.

Ideally, this model would be used as a tool to flag nations that are at risk of experiencing a coup attempt, so that human actors can look more closely at the current state of affairs in that nation, then make an informed decision about if intervention is necessary. This model should NOT be used to replace human judgement. Given that the cost of incorrectly predicting a coup attempt is wasted time and resources, while the failure to flag a coming coup attempt results in human misery, not the mention the potential loss of innocent lives, I chose to utilize recall as my final scoring metric.

**FIGURE 1:**



While my aim was to maximize recall, it was not necessarily optimal to choose a threshold with the highest recall score. As shown in Figure 1, above, catching every coup would result in falsely flagging a coup nearly 66% of the time. This comes at the expense of time and resources on a scale that is not allocated to humanitarian causes. Thus, it seemed more sensible to try to lower the false positive rate a bit, while still catching as many coups as possible. In the end, I decided upon a threshold of 0.43, the results of which are shown in Figure 2, below. While we may miss 10% of actual coup attempts, the false positive rate drops down to 31%. This seems a much more realistic application in the real world.

**Figure 2:**

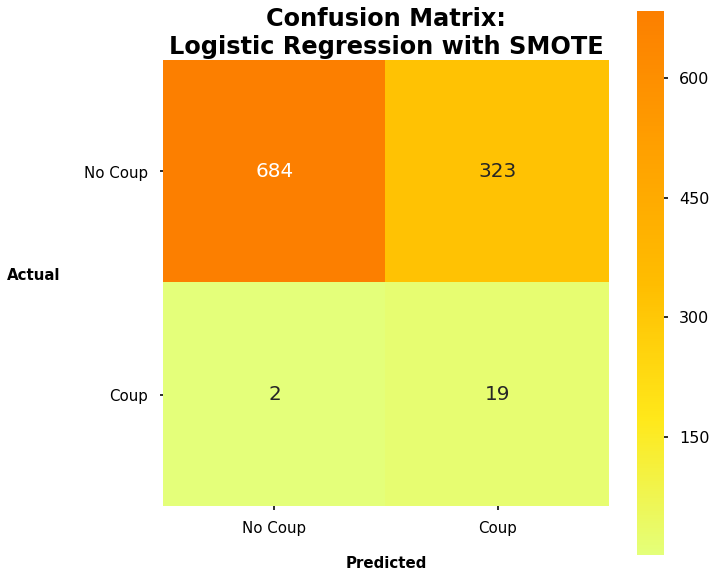


Figure 3, below, depicts the standardized coefficients of the model’s most predictive features. The length of a leader’s regime (coef = -0.315) had the most predictive power in this model, closely followed the percentage of the population that has access to electricity (coef = -0.307) and the anticipation of an irregular election (coef = 0.295).

**FIGURE 3:**

|  |  |
| --- | --- |
| **Feature** | **Coefficient** |
| Regime tenure | -0.315 |
| Access to electricity | -0.307 |
| Irregular election anticipated | 0.295 |
| Military career | 0.235 |
| Life expectancy | -0.207 |
| Human capacity | -0.202 |
| Military population | 0.165 |
| Election occurred recently | -0.144 |

**Future Model Improvements:**

Given the highly complex nature of the topic, I was pleased to have constructed a model that had any predictive power, let alone an AUC score of 0.83. This indicates that this study has great potential if future improvements were to be implemented.

First, I would have liked to have designed a more sophisticated solution to handling the abundance of null values in my data. For future work, I would like to try to impute the null values with the mean per country, to ensure that data on Sweden does not impact the data on Somalia, for example. Alternatively, I could try to predict null values via machine learning methods such as k-nearest neighbors or linear regression.

I could also supplement these null imputations by combining similar features into one variable. For instance, life expectancy and the percentage of the population with access to electricity, sanitation, and water correlated fairly well with one another. Good model design would dictate that we should remove variables with strong multicollinearity. However, because each of these features contained a lot of null values it did not make sense to keep any one of them as an independent variable. However, in combining these features into a variable to create a development index might have been a nice solution to both the multicollinearity as well as the slew of null values. If any nulls remain after the features have been combined, I could impute them with the mean or utilize machine learning methods to predict the missing values.

Next, given more time, I would like to play around with which variables to utilized within my modeling. Not having a background in economics, I struggled with the decision on which economic metrics to include in my model. Though I did some research before deciding which economic metrics to employ, perhaps I could do better by consulting an economist about which metrics might best capture different aspects of government revenue and expenditure.

Above, in the Background section, I discussed how different types of government please their key supporters through vastly different means. While I had intended to add government type as a feature, I did not have to time to make this happen. I believe that the addition of this feature to my model will significantly improve the performance of my model. Finally, as with all of our assignments, I would have liked more time to engineer my features and train other models such as SVM, Linear SVM, or Naïve Bayes.

**Personal Improvements:**

* Business case was better than the last, despite having a humanitarian purpose.
* This project felt much more organized than the last. I had a notebook dedicated for each step of the process, and spaghetti code was minimized. If a notebook got too jumbled, I made a copy so as to save the old code in case I wanted to return to it. Code that was not relevant for my current progress was then deleted from the new notebook. However, there is still improvement to be made in my organization. I plan to utilize Jupyter Notebook’s in future projects to avoid the need to copy notebooks.
* I made a Python script for the first time during the course of this project. Within it, I defined functions and variables that were common to several notebooks. However, this is where I struggled with organization. It quickly became rife with spaghetti code as I progressed through the project. It became hard to remember what elements of the script were still relevant, so it is clear that there is still more improvement to be made there.
* I had a much better sense of workflow throughout this project. Though I needed clarification from time-to-time, it was much more clear as to what steps I needed to be taken. Very rarely was I unsure what to do next.

**Lessons Learned:**

* Task prioritization. Complete pipeline before going too in-depth with data cleaning. I was too engrossed in how I could make my sparse data more accurate, that I didn’t start modeling with scikit-learn’s GridSearchCV until late in the game, despite having it mostly set up.
* Data leakage is a bitch, and it burned me too many times throughout this project. However, early failures lead to future success. I appreciate the value of making mistakes, as I am sure to be much more cognizant of leaky variables in future projects.
* There is most certainly an easier way to accomplish what I want. For instance, I spent quite a bit of time writing my own grid search, until I was later introduced to scikit-learn’s version of a grid search. Though the process of trying to figure it out on your own has pedagogic value. Only through this struggle can we truly understand the processes and appreciate the value and elegance of these speedy functions.
* I still need to work on the balance between accuracy and finishing. I became too enchanted with scikit-learn’s gird search and tried to tune the parameters too finely, which led to unreasonably long computation times. It got to the point where I was not able to complete a full grid search of random forest models, despite leaving it run overnight. But I kept thinking that though it is taking a while, I will be set once it finishes. In this case, it never finished. It was for this reason that I did not run my test set until the night before the project was due. I should have stuck to more crude parameter estimates much earlier in the process, yielding quicker results and more insights. Leaving me in a compromising position to present my results. That evening, I performed much more crude parameter searching which I was able to accomplish in two hours. Meaning all that I got out of that gridsearch was anxiety, frustration, and a valuable learning experience.