Florida Flippers

Introduction

Our group decided to apply our machine learning and data analytics skills to predict election results. Given this year's uniquely long and thrilling election period, we decided to predict the 2020 Presidential Elections, not on a national level, but on a state level. We wanted to understand which factors drive election results, and if possible, predict the results of a state. But which state is worth predicting, both in terms of difficulty and importance to the US elections?

With this objective, we selected Florida because it is a critical state in the election process given its 29 electoral votes: the third most in the US. Furthermore, Florida's demographics (which constitute a large chunk of our data) is particularly intriguing given its uniquely diverse Latino presence (Mexicans, Puerto Ricans, Cubans etc), a rising Asian population, and a consistent Black population. Furthermore, it is the state with the largest 60+ age group population (and 2nd largest per capita, after Maine[1]). Given Florida's diversity, it is considered a bellwether state and has selected the winning candidate in all but four elections since 1928. All these reasons, plus perhaps the infamous Bush v Gore in 2000, made Florida a perfect state for our project goals.

Data Gathering and Cleaning

Relevant data includes election data as well as demographic, income, occupation and other miscellaneous data for all of Florida's 67 counties. Demographics included 13 age brackets (from 'less than 5 yo' to 'over 85 yo'), 12 race brackets (ranging from Native American to all Latino breakdowns and one race vs multi-race), 18 occupation brackets (ranging from manufacturing and construction to Science, Business and Information), 10 income brackets (from less than $10,000 to more than $200,000 annually per household) and other data ranging from poverty and health insurance coverage to unemployment rate and male % vs female % breakdown . We retrieved the demographics data from the US Census Bureau[2] where we downloaded DP03: 'Selected Economic Characteristics' and DP05: 'ACS Demographic and Housing Estimates', for 2019 using the '5-year Estimate Data Profiles'. We downloaded the data in .csv format where we first cleaned it using excel (basic stuff like transposing, changing column names with weird symbols etc) and normalized all values to represent 'X per 100 people'. This made it easier to compare within Florida counties. For the election data, we downloaded the 2012,2016,2020 Presidential Election Data from Florida's Department of State website (under Election Results Archive[3]) where we downloaded a .txt file with all the relevant data. We then converted the .txt to a .csv and continued the cleaning process. We broke down the total votes to 'percent Democrat', 'percent Republican', 'percent difference' and 'total Democrat votes', 'total Republican votes', 'votes difference' for all three elections. This gave us a total of 89 factors for our data. Finally, we merged the two datasets on 'combined_fips[4]', a column we created, which offers similar functionality to a primary key in SQL, and continued with our analysis.

Visualizations, Correlations, and Machine Learning

We started off by making visualizations for election results. This includes constructing a Choropleth map for the US Florida election results in 2012, 2016, 2020 as well as another Choropleth map for counties that flipped from one election to the other. These were not only useful for presentation

[1] https://www.prb.org/which-us-states-are-the-oldest/

[2] https://data.census.gov/cedsci/advanced

[3] https://dos.myflorida.com/elections/data-statistics/elections-data/election-results-archive/

[4] FIPS county codes are standards used in data and military. Each county has a unique FIPS code.

purposes, but also to visualize any potential election trends (e.g. Pinella County switched consecutive elections so it is a battleground county).

We focused on 'percent Democrat 2020' since we wanted to predict the 2020 election and used this to correlate it with our factors data. The results were expected and straightforward: high income earners/jobs that pay well/require high level education (like Information, Business,Science etc) as well as young people (20-35) tend to vote Democratic (i.e. have a positive and relatively large R value) and low income earners/low paying jobs and older age groups typically had negative (and in absolute value) larger R values. For racial data, Asians were the most likely to vote Democratic, followed by Other Latinos (i.e. other than the ones mentioned above), Puerto Ricans and Blacks (for all R>0.4). Whites were the most likely to vote Republican (-0.54). Finally, females were correlated highly with democratic voting (R=0.61) whereas males were most likely to vote Republican (R= -0.61). To visualize and summarize these results, we made Bokeh barcharts, but also created a correlation matrix using sns.heatmap and matplotlib.patches.Rectangle. This more analytical visualization allows for easy and direct correlation observations.

Moving on to the machine learning model. Before jumping deep in our supervised ML algorithm, let's note the following assumption we made: only two parties exist in our prediction (GOP and Democratic). This should not change the results by too much (historically independents/third parties only amass roughly 1% of total votes) and we would only be predicting what % of votes is democratic for each county. We used a Regressor (and not Classifier) since regressors are used for predicting continuous values, whereas classifiers for categorical ones. We considered various regression models like Linear Regression, SVR (Support Vector Regressor), decision trees like Random Forest and other more advanced boosting models like CatBoost, LightGBM, and XGBoost (all three of which also use decision trees). After considering all of them, we decided to use CatBoost for our model for the following reasons:

1. It yields state-of-the-art results without expensive data training (we only have 67 counties so being efficient with training data is important)
2. It requires less parameter tuning than its peers (the default parameter values bring you quite close to the hypertuned results (15% deviation in our case). This allows us to get a good impression of our model's potential very quickly)
3. Catboost is better at regularization[5] than the other models - (a) it uses a stronger L_2 regularization on the weights (3 vs 1 in XGBoost, 2 in LightGBM) and (b) it uses an additional regularization parameter (bayesian_matrix_reg) for leaf-value calculations
4. Uses gradient boosting to reduce bias by fitting consecutive trees at each step and solves the net error from prior tree (same as XGB,LGBM) unlike LR, SVR,RandomForest
5. It contains an overfitting detector: Using use_best_model=True with od_type='Iter', the model can stop training earlier than the training parameters dictate, if overfitting occurs

Thus, we first trained a non-tuned CatBoostRegressor and performed repeated cross validation with K=10 and returned MAE,MSE,RMSE and R^2. Our errors were small (RMSE=0.003, R^2=0.67) but we could do even better with parameter tuning and GridSearchCV. Hence, we did exactly that with different parameter values (check code) and even repeated this process a couple of times (CatBoost documentation offers guidance for typically optimal parameters. We used these, plus some orders of magnitude more and less) until we could find the optimal parameters. After this step, we used the overfitting detector (mentioned above) to get 999 overfitting-free iterations. At this stage we were confident that we had taken overfitting (and bias-variance tradeoff) into account and that our generalization error was small, so our predictions were realistic. We ran our model with the optimal # of

[5] Regularization is a form of regression that constrains (or shrinks) the coefficient estimates to zero. Thus, it discourages a more complex model so as to avoid the risk of overfitting, which is critical in any ML discussion.

iterations and hypertuned parameters for different typical K choices (K=5, K=10, K=15), as well as different n_repeat values (n_repeat=3,5,10) for the Repeated Cross Validation and got our final model errors and score (MSE=0.0044, MAE=0.046, RMSE=0.0022 and R^2 = 0.76) with K=15, n_repeat=5.

Our next step was to find (and plot) the feature importances of our predictive model (check code or final presentation). Feature importances ranks our model features by relative importance (i.e. features that split nodes closer to the root of the tree will result in a larger relative importance and vice versa, but keep in mind that the actual way it is calculated is very complex). At this point we also analyzed the features by calculating the set of basic statistics and plotting for the 10 most important features and also for 1-2 unimportant ones for comparison purposes. This is a quick and reliable way to inspect, visualize (includes widgets) and find interesting results in our list of features upon our analysis (check code).

Results and Improvements

Our model's final prediction for Florida 2020 is J. Biden (DEM) 46.4% and D. Trump (REP) (1 - 46.4%) = 53.6% versus the actual J. Biden (DEM) 47.9% and D. Trump (REP) 51.6%. We also visualized the final results, again using Choropleth, showing our predictions, actual election results and MSE for each county. We were very close to the actual result (and did on average better than some of the country's' leading media and polling companies[6]), as exemplified by our prediction values and Florida map (check code). Although we are satisfied with what we got, we believe we could further improve our model by:

1) Removing less important features and replacing them with more useful data (for example campaign funding data, although we could not find a convincing dataset for that)
2) Accounting for COVID-19 having an impact on the elections (we believe that Democrats gained some votes from Trump's poor handling of the pandemic that is not shown in our model)
3) Including all US counties would improve the model given the training (and by extension test) set would increase substantially and trends would be clearer and more evened-out throughout the country
4) Including independents and third parties would decrease the Republican's lead, giving a more realistic solution (it would not affect democrats since they are the target variable)
5) Combining some sort of 'polling metric' so that our model is not exclusively data-driven (e.g. pollsters prediction uses 98% polling data and 2% demographic data for prediction. We would probably implement the inverse). We could do this in two ways:
    a) Perform/Find an existing poll on all Florida counties with demographic info, party voted in previous election, and also people's views on some contentious topics
    b) Perform sentiment analysis (probably on Twitter) to gauge people's preferred candidate and their views on contentious topics

We would ask these contentious topics given that polling accuracy has decreased the last 5 years or so maybe because people might be embarrassed about their candidate of choice, or more democrats are eager to participate in these polls (plus many more...) leading to skewed results. Thus, people's views on these topics can indicate which side they are leaning on the most (regardless of what party they say they will vote for).

Overall we are very content with our project. We applied the material learnt in class and extended it further (especially in our Machine Learning model) to get good predicted results with small errors. Some improvements could be made, which is always welcome and which could further improve our model.

[6] Please consult table in final presentation (original source is: https://projects.fivethirtyeight.com/2020-election-forecast/florida/)