Omar Villalba Vazquez

https://www.linkedin.com/in/omar-villalba-0550a612a/

September 18th 2017

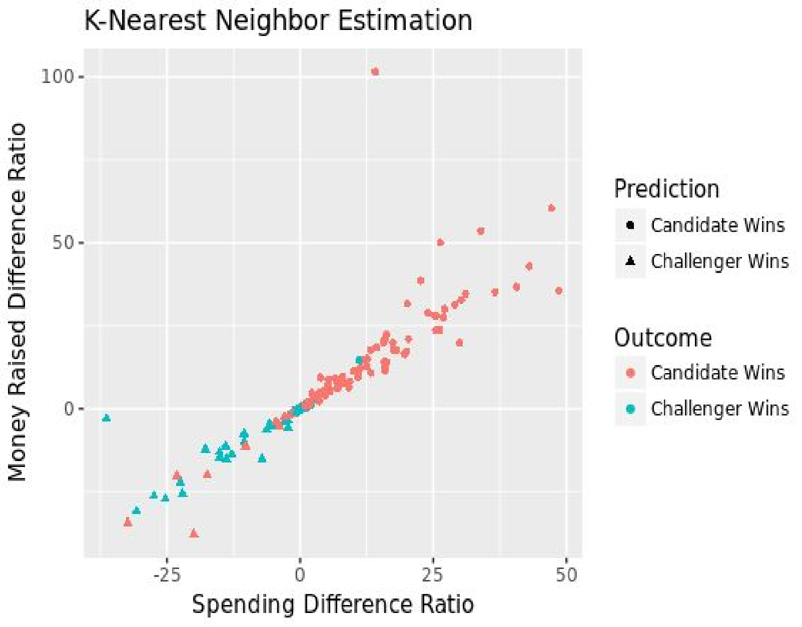
Small Essay on Congressional Elections

Congressional elections in the United States occur every 2 years at the federal level, with the a large number of senators and congressmen vying for reelection, usually against an opponent from the opposing party. Naturally, the understanding what affects the chances of getting elected is a topic with a great deal of interest and even more interesting is the capacity to predict these congressional elections. Using machine learning tools, analysis and classification of congressional elections is going to be the focus of the following essay.

The dataset obtained comes from the website OpenSecrets.org, which is part of the Center for Responsive Politics and contains 300 observations. Given the nature of elections in the United States, in which 2 parties tend to have solo monopoly on congressional seats, the dataset contains variable related with regards to the candidates of both parties. It includes the amount of money spent and raised by both candidates, as well as who is the incumbent in the election. Congressional races without an incumbent were not included in the dataset.

To start, the data was randomly split into a training set with 250 observations and a test set with 50 observations. The first model used analyze the data is a logistic regression, which includes all of the variables previously presented. For the purpose of regularization, Ridge Regression was utilized, with the value of lambda chosen based on k-fold cross validation, with k=10. The results of the regression were as one would expect with regards to the direction of the coefficients, negative for opponent raising and spending, while positive for the candidates itself. As mentioned in other analysis of American Elections, the incumbency has a positive effect on the electors chances of the candidate, although this is likely in good part due to incumbents having better access to financing. This is a hypothesis supported by the current dataset, in which ⅔ of the incumbents outspend or outraised their challenger at least 2 to 1.

While the logistic regression does provide insight into the relationship that the variables posses, its predictive power is not very good, with the test set having a classification error of 31%. In order to obtain a better predictive power, a k-nearest neighbor model model was prepared, with k = 8 and having being chosen based on



10 fold cross validation, similar to the logistic regression. The variables used are ratios of the difference between spending or raising money. Essentially, the values between the candidate and the challenger are subtracted and then divided by the smallest of the 2. To give a simple intuitive example, if the candidate spends $20, and the challenger $10, then the ratio would be (20-10)/10 = 1. If the ratio is positive, the candidate spent more and vice versa if its negative. Effectively, if the ratio is 1, then the candidate outspent the challenger by 100%, if it were -5, the challenger spent the candidate by 500%. Using these variables, the KNN model was far more successful on the test set than the logistic regression, with only 16% classification error rate. However, there is an issue with the both models, which can be seen to an extent in the KNN training set graph above. Both models do well when the difference ratios are large, but not so much when these ratios are between -2 and 2. The KNN model has only 60% error rate in those cases, while the logistic regression is 40%.

To conclude, the models presented here do reasonably well when there is cases of large outspending, but otherwise fails to predict consistently. Due to the nature of congressional elections, most races have heavy outspending, usually by the incumbent, which is why the model can do well in some cases for classification. However, for further research, besides obtaining more data, preferably large enough to obtain rare cases like an outspending incumbent losing, it is absolutely necessary to obtain more independent variables, particularly in order to prepare better models for classification.

Essay Code

## Necessary Packages, data split and variable creation ##

```
library(caTools)
library(sqldf)
library(glmnet)
library(kknn)

ElectionsData$Win <- ifelse(ElectionsData$Vote > ElectionsData$OpponentVote, 1, 0)

ElectionsData$SpendDiff <- ElectionsData$MoneySpent - ElectionsData$OpponentSpent

ElectionsData$SpendDiffRatio<- (ElectionsData$MoneySpent -
ElectionsData$OpponentSpent)/pmin(ElectionsData$MoneySpent,
ElectionsData$OpponentSpent)

ElectionsData$RaisedDiff <- ElectionsData$MoneyRaised - ElectionsData$OpponentRaised

ElectionsData$RaisedDiffRatio<- (ElectionsData$MoneyRaised -
ElectionsData$OpponentRaised)/pmin(ElectionsData$MoneyRaised,
ElectionsData$OpponentRaised)


set.seed(300)
sample = sample.split(ElectionsData, SplitRatio = .85)
OldData = subset(ElectionsData, sample == TRUE)
NewData = subset(ElectionsData, sample == FALSE)
```

## Logistic Regression ##

##Matrices of the independent variables in the regression, split into training and test set respectively ##

```
LogTrainingInput <- as.matrix(sqldf('SELECT MoneyRaised, MoneySpent, OpponentRaised,
OpponentSpent, Incumbency FROM OldData'))
```

```
LogTestInput <- as.matrix(sqldf('SELECT MoneyRaised, MoneySpent, OpponentRaised,
OpponentSpent, Incumbency FROM NewData'))


## Logistic regression ##

set.seed(100)
logreg <- cv.glmnet(LogTrainingInput, OldData$Win, family = c("binomial"), alpha = 0,
type.measure = 'class' , nfolds = 10)

bestlambda <- logreg$lambda.min

NewData$logpredict <- predict(logreg, type =
c('response'), newx = LogTestInput, s=bestlambda )

NewData$logPrediction <- ifelse(NewData$logpredict > 0.5, 1, 0)

table(NewData$logPrediction, NewData$Win)

##  Checking spending and raising differences for incumbents ##

sum(ElectionsData$Incumbency == 1 & ElectionsData$SpendDiffRatio > 1 |
ElectionsData$Incumbency == 0 & ElectionsData$SpendDiffRatio < -1)

sum(ElectionsData$Incumbency == 1 & ElectionsData$RaisedDiffRatio > 1 |
ElectionsData$Incumbency == 0 & ElectionsData$RaisedDiffRatio < -1)
```

## K-Nearest Neighbor model ##

```
## First, set response variable into factor ##

NewData$Win <- as.factor(NewData$Win)
OldData$Win <- as.factor(OldData$Win)

## Train the KNN model ##

TrainingKNN<- train.kknn(formula = Win ~ SpendDiffRatio, data = OldData, kcv = 10, kernel =
'rectangular', kmax = 10)
```

```
## Check the quality based on the Test Set ##

NewData$Prediction <- predict(TrainingKNN , newdata = NewData)

table(NewData$Prediction, NewData$Win)

## Graph of KNN Model ##

OldData$Pred <-as.vector(TrainingKNN$fitted.values[[8]])

OldData$Prediction <- ifelse(OldData$Pred == 1, "Candidate Wins", "Challenger Wins")

OldData$Outcome <- ifelse(OldData$Win == 1, "Candidate Wins", "Challenger Wins")

GraphingDataFrame <- subset(OldData, SpendDiffRatio< 50 & SpendDiffRatio > -40)

qplot(SpendDiffRatio, RaisedDiffRatio, colour = Outcome, shape = Prediction, data =
GraphingDataFrame, xlab = "Spending Difference Ratio", ylab = "Money Raised Difference
Ratio", main = "K-Nearest Neighbor Estimation")


## Check races with similar spending ##

NewData$PredDiff <- as.numeric(NewData$Win) - as.numeric(NewData$Prediction)

NewData$LogPredDiff <- as.numeric(NewData$Win) - as.numeric(NewData$logPrediction)

sum(NewData$SpendDiffRatio > -2 & NewData$SpendDiffRatio < 2 )

sum(NewData$SpendDiffRatio > -2 & NewData$SpendDiffRatio < 2 & NewData$PredDiff == 1)

sum(NewData$SpendDiffRatio > -2 & NewData$SpendDiffRatio < 2 & NewData$LogPredDiff
== 1)
```