Machine Learning Engineer Nanodegree

Capstone Project

Predicting Speed of Bill Passage in Canadian Parliament

I. Definition

Project Overview

A common concern with the passage of a bill into law in the legislative body of a country is the likelihood of its success. There has been some recent interest in investigating this question for the United States Congress[1], where bill survival rates are very low[2]. In the Parliament of Canada, a bill passes into law by receiving Royal Assent. To date, 538 bills have received royal assent, whereas 295 were defeated or not proceeded with[3]. This indicates that the probability of success of a bill is much higher in Canada. A follow-up question one may then ask is this: how much time would it take for a newly introduced bill to pass into law, assuming that it receives royal assent? We believe that building a model to accurately answer this question would be fruitful, as there is often considerable political and economic interest in certain pieces of legislation[4].

Problem Statement

We aim to build a binary classifier that will predict the passage speed of a bill in the Parliament of Canada. The life of a bill will be defined as the number of days from the date of first reading until the date of royal assent.

$$bill\ life\ =\ royal\ assent\ -\ first\ reading$$

We shall create two categories; fast and slow. The fast category will represent bills that have a life of less than a particular value, and the slow category will represent bills that have a longer life. The value we choose for this purpose is 171 days, which divides the data into an approximately 59:41 split of fast and slow bills respectively. The reason 171 is chosen rather than the median is because it is the number of days from first reading of Bill C-45[5] until October 1st (which is the end date of a question posed by the Economist[6]).

The Parliament of Canada offers a website[3] which provides conveniently formatted XML files containing bill metadata as well as textual information, such as the title and main content of a bill. We shall create two feature sets - one comprised of the bill metadata and the other using vector representations of the text in bill titles.

Metrics

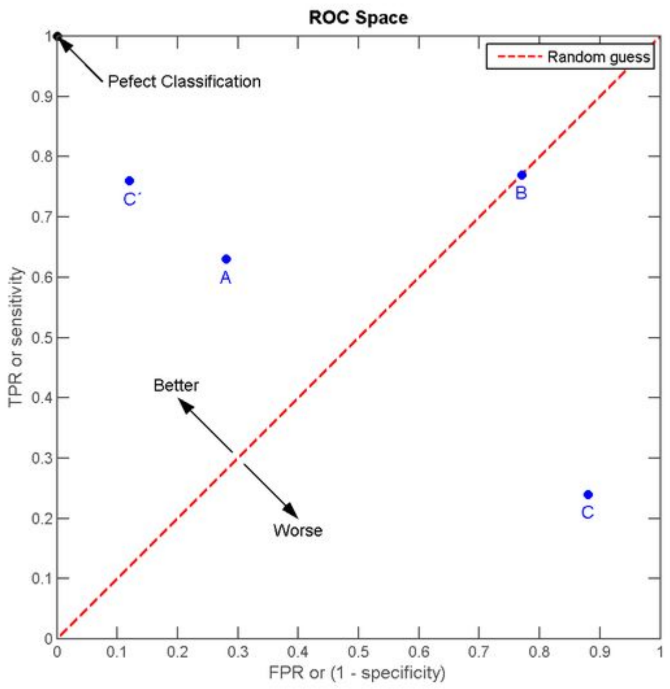
To assess the performance of our models, we shall consider two metrics. The accuracy score is a simple and easy to interpret metric,

$$accuracy\ =\ \frac{tp+tn}{tp+tn+fp+fn}\ ,$$

where tp = true positive, tn = true negative, fp = false positive, fn = false negative.

A more robust metric we shall also consider is the receiver operating characteristic (ROC) score, which is computed as the area under the ROC curve[8],

$$A = \int_{\infty}^{-\infty} \mathrm{TPR}(T)\left(-\mathrm{FPR}'(T)\right)\, dT = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(T' > T) f_1(T') f_0(T)\, dT'\, dT = P(X_1 > X_0)$$



This metric is derived from the ROC space, which is a plot of the true positive rate against the false positive rate as shown in the diagram.

We may wish to, for example, prefer a trade-off with a slightly lower false-positive rate, and the ROC curve gives us a way to visualize this in order to make adjustments.

II. Analysis

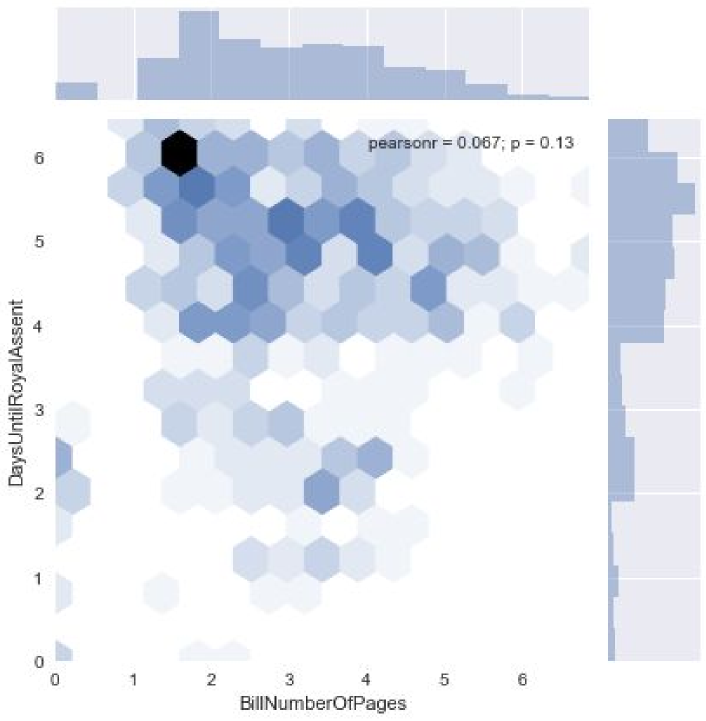
Data Exploration

One input to the machine learning model will be in the form of a pandas DataFrame with 12 features, however, some of them may be removed after some investigation. The feature of the number of pages of the bill has 8 null values, while the bill sponsor title has 2 null values. We can either impute the missing values in the bill number of pages or drop them entirely. Since the bill sponsor title has only two missing values, we simply drop these.

Another input will be the text titles of bills, which will be vectorized as word embeddings. This feature set will be dealt with in a separate pipeline. There do not appear to be any missing or malformed data in the titles.

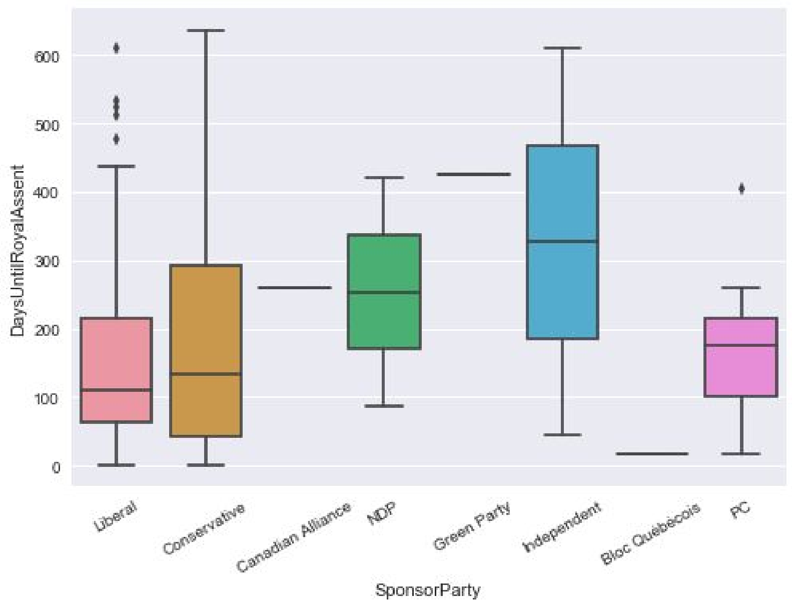
Exploratory Visualization

We primarily use boxplots to visualize the data because our dataset has only categorical features, with the exception of the number of pages of each bill.



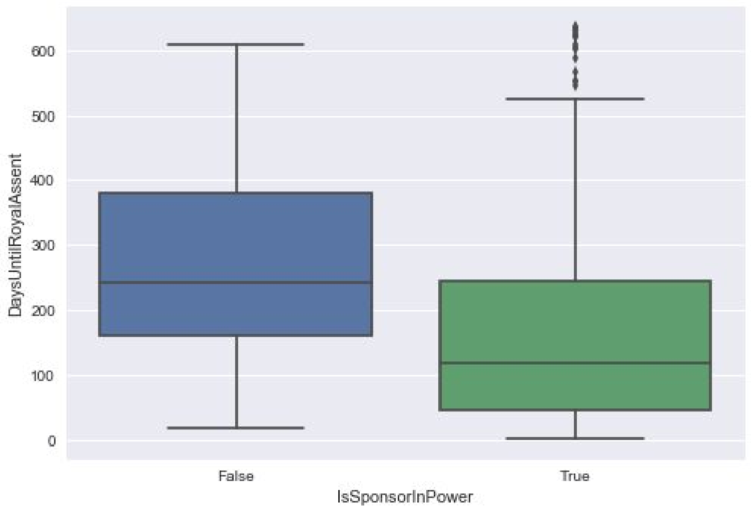
Surprisingly, there seems to be virtually no correlation between the number of pages of a bill and how long it takes before being passed into law. This likely means that the number of pages is not a good proxy for bill complexity, as presumably a more complex bill would require more extensive deliberations before being passed.

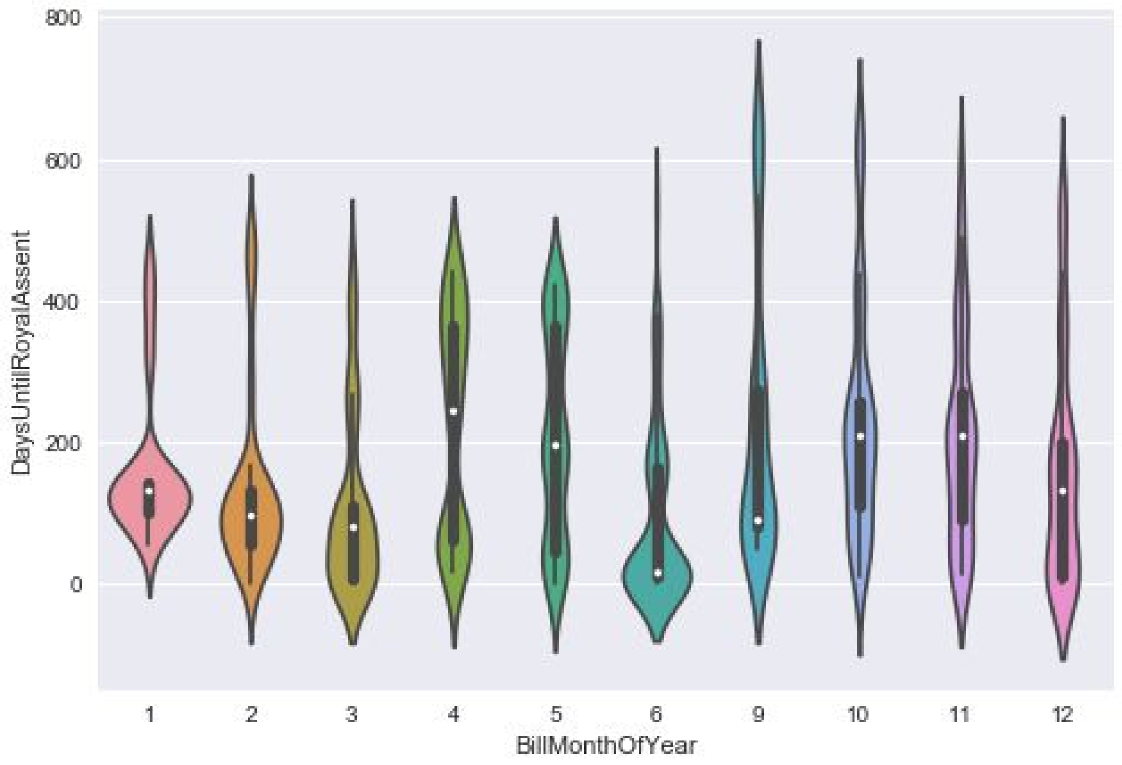
The data on both axes are log transformed because the distributions are highly skewed. We also note that the pearson correlation coefficient has a low value of 0.067.



We may wish to consider dropping the 'SponsorParty' feature, in order to avoid possible issues with multicollinearity with the 'IsSponsorInPower' feature. It certainly seems that the mainstream parties pass laws more quickly, but that is precisely because on average the party in power would more easily pass laws.

Examining this directly, we see that there is a significant difference in the distributions depending on whether the sponsor of the bill's party is the party currently in power, although there are some notable outliers.

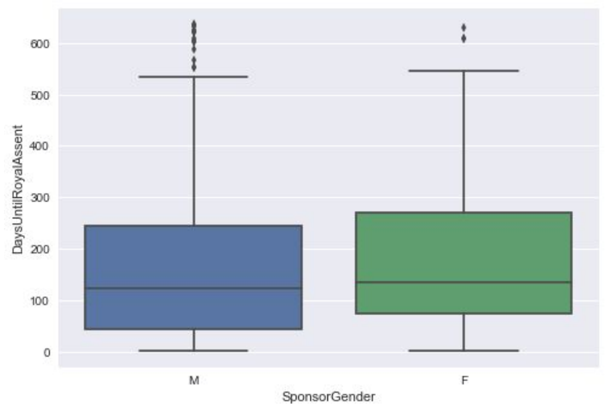


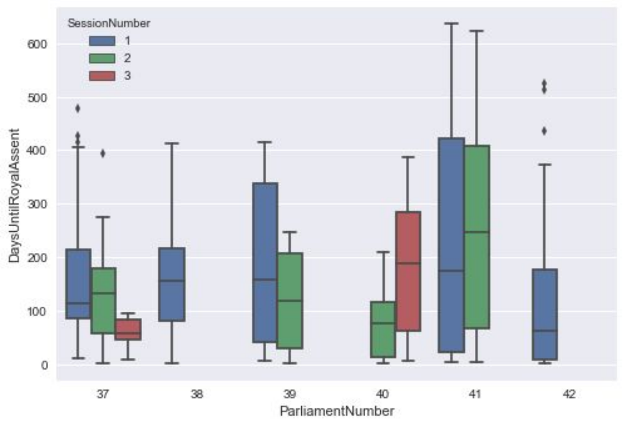


Using a kernel density estimate with a bandwidth of 0.33, we can see that in month 4, which is April, the distribution appears bimodal.

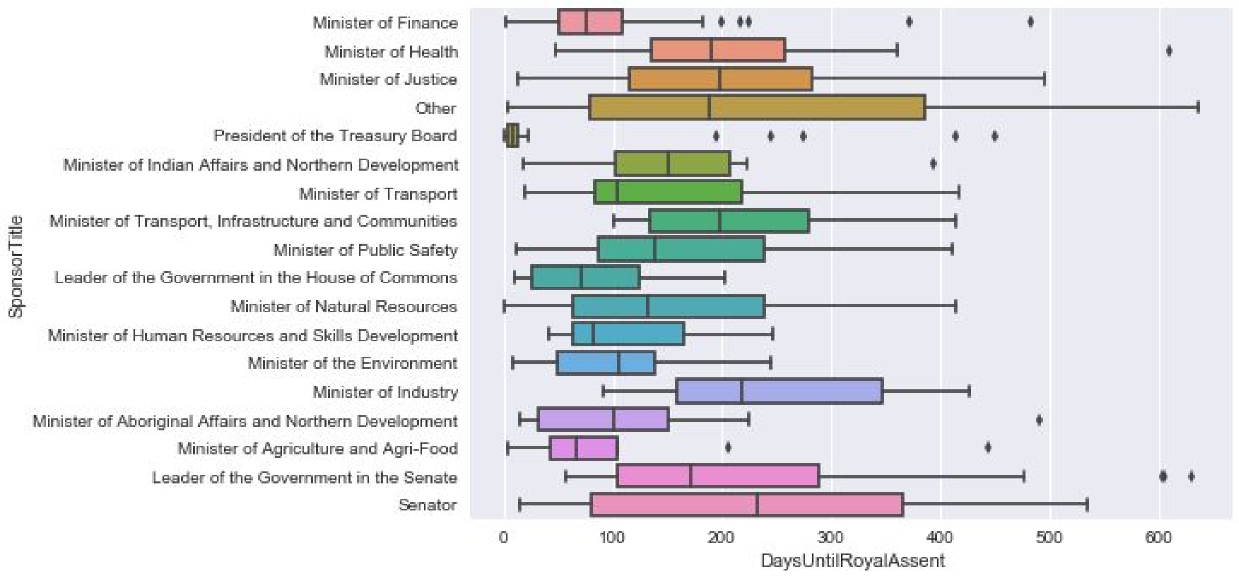
This makes sense because there is a two month long summer recess during the months of July and August. This is also why months 7 and 8 are absent from the plot.

Some candidate features to drop are the bill sponsor’s gender and the parliament and session numbers. The bill sponsor gender shows no real difference in the distribution, whereas the parliament and session numbers have no discernable pattern.





Finally, we note that bills sponsored by the President of the Treasury board are usually almost immediately passed, as shown in the diagram below:



The second feature set involving word embeddings can be visualized using word clouds. The image on the left represents the word cloud for titles of 'fast' bills, and the image on the left shows the word cloud of the titles of 'slow' bills.

Algorithms and Techniques

We favor classifiers that provide a well calibrated probability estimate[7], as confidence in our predictions is a crucial piece of information for answering certain types of questions[6]. Logistic regression is well suited to this task and so we use it for both feature sets. A logistic regression classifier also works well with a small number of features and is quite fast.

Logistic regression is a linear model that is in many respects quite similar to linear regression. It is also a regression, as the output is real - between the values 0 and 1, but it is bounded like classification. A regular linear regression would involve taking the dot product of weights and data

$$h(x) = w^T x \, ,$$

whereas logistic regression uses the logistic function to operate on the output. The logistic function is defined as

$$\theta(s) = \frac{e^s}{1+e^s} \, .$$

The crucial feature of this function is that it provides a soft threshold which allows a continuous range of values between 0 and 1, and these can be interpreted as probabilities for a particular binary outcome.

We now discuss the error measure for logistic regression. Assuming that the **x** and y data pairs are independently produced, the probability of getting everything in the data set correct would be the product of the probabilities of each data pair,

$$\prod_{n=1}^{N} P(y_n|x_n).$$

It is more convenient (and numerically prevents underflow) to take the negative log of this value and minimize instead. This works because log is a monotonically increasing function. Ultimately we get what is known as the cross-entropy loss.

Unlike with linear regression, an analytic solution to minimizing this quantity is not feasible, so we make use of the gradient descent algorithm. Gradient descent works particularly well with cross-entropy is a convex function, and has a unique global minimum[10].

The initial model will use the default values, which are the following:

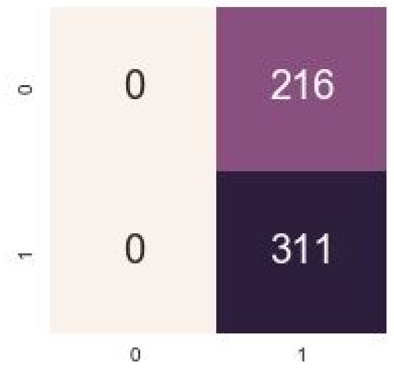
- penalty='l2'
- dual=False
- tol=0.0001
- C=1.0
- fit_intercept=True
- intercept_scaling=1
- class_weight=None
- random_state=None
- solver='liblinear'
- max_iter=100
- multi_class='ovr'
- verbose=0
- warm_start=False
- n_jobs=1

This model will then be fine-tuned by performing an exhaustive grid search for values of the ‘C’ parameter, which represents the inverse regularization strength.

In order to make use of bill title data, we first tokenize the title string into words, and then compute the word frequencies. Although we use the tf-idf algorithm, which stands for term frequency - inverse document frequency, we only make use of term frequencies - that is, we set the idf parameter to False. This creates a sparse feature matrix that can be fed into a machine learning classifier (in our case, logistic regression). We are then able to associate certain word frequencies with a positive outcome and others with a negative outcome. Thus, the following default parameters are used:

- norm=u'l2'
- use_idf=False
- smooth_idf=True
- sublinear_tf=False

Benchmark



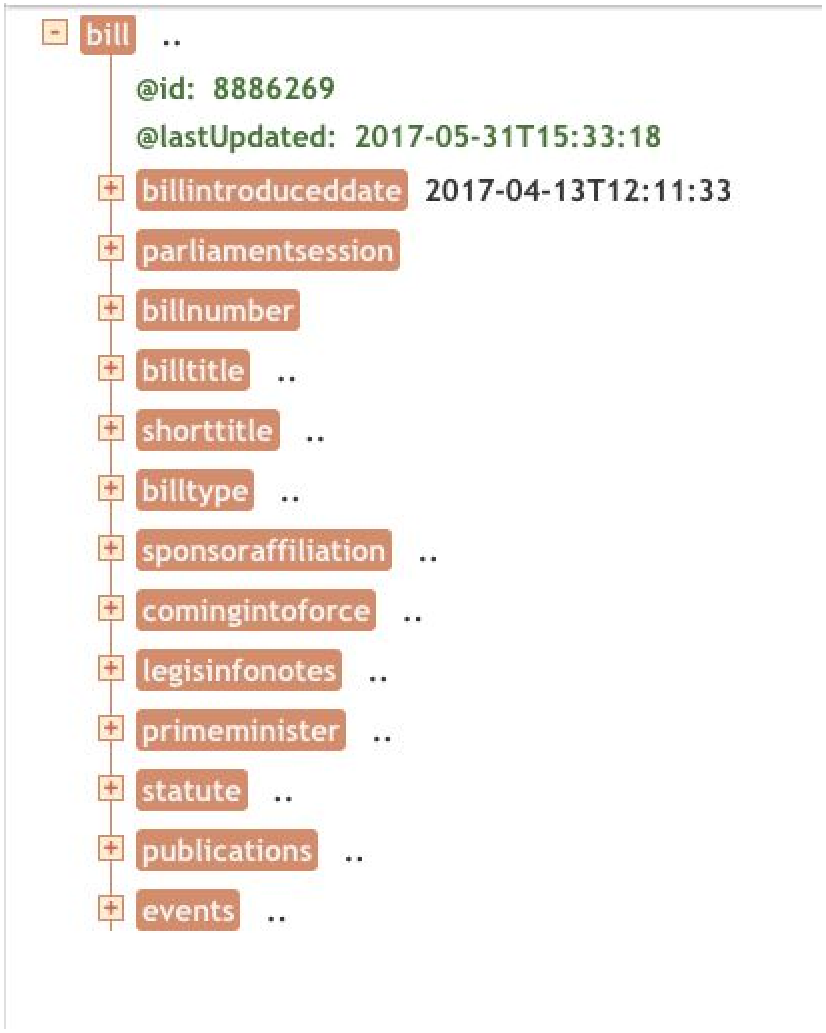
Although similar studies have been done on the probability of bill passage in US Congress, we are not aware of any studies on bill passage speed in Canadian Parliament. Therefore, we start with a naive predictor of assuming all bills will be 'fast' bills. This gives an accuracy of 0.59, and an roc_auc score of 0.5.

III. Methodology

Data Preprocessing

The raw data is obtained from the Parliament of Canada website as an XML file of all the bills that have received royal assent to date since the government started collecting this data. The root node has 537 child nodes, each of which represents a bill. These are the observations in our data. This number of observations is on the low side, which is something we may wish to keep in mind when creating our features; a good rule of thumb would be not to exceed 10-12 features in our first feature set (using bill metadata).

Extracting Bill Metadata



```
⊟ bill ..
    @id: 8886269
    @lastUpdated: 2017-05-31T15:33:18
  ⊞ billintroduceddate 2017-04-13T12:11:33
  ⊞ parliamentsession
  ⊞ billnumber
  ⊞ billtitle ..
  ⊞ shorttitle ..
  ⊞ billtype ..
  ⊞ sponsoraffiliation ..
  ⊞ comingintoforce ..
  ⊞ legisinfonotes ..
  ⊞ primeminister ..
  ⊞ statute ..
  ⊞ publications ..
  ⊞ events ..
```

Our initial goal is to preprocess the data into a number of features that can be structured into a pandas DataFrame. The top level tree structure of a typical bill is shown below.

We begin by creating the target variable. As noted earlier, this is the difference between the royal assent date and the first reading date in days, and then bucketed into two categories. We can find this information under the 'events' node in each XML document.

One feature which may be relevant is the size of a bill in terms of the number of pages in PDF format. Our starting point is thus a DataFrame with the following columns:

```
1. BillID
2. BillIntroducedDate
3. BillNumber
4. BillNumberOfPages
5. BillTitle
6. BillType
7. BillURL
8. DaysUntilRoyalAssent
9. FirstEventChamber
10.   FirstEventDate
11.   FirstEventId
12.   FirstEventMeetingnumber
13.   LastEventDate
14.   PDFURL
15.   ParliamentNumber
16.   PrimeMinisterName
17.   PrimeMinisterParty
18.   SessionNumber
19.   SponsorGender
20.   SponsorParty
21.   SponsorTitle
22.   StatuteChapter
23.   StatuteYear
```

Many of these features are definitely not relevant, such as StateYear, StateChapter, and PDFURL, so we shall refine the selection into the following final DataFrame which will serve as our feature set ready for exploratory data analysis and consumption of our machine learning model:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 537 entries, 0 to 536
Data columns (total 12 columns):
BillNumberOfPages        529 non-null float64
BillType                 537 non-null object
DaysUntilRoyalAssent     537 non-null int64
ParliamentNumber         537 non-null category
SessionNumber            537 non-null category
PrimeMinisterName        537 non-null object
PrimeMinisterParty       537 non-null object
SponsorGender            537 non-null object
SponsorParty             537 non-null object
SponsorTitle             535 non-null object
BillMonthOfYear          537 non-null category
IsSponsorInPower         537 non-null bool
dtypes: bool(1), category(3), float64(1), int64(1), object(6)
memory usage: 35.9+ KB
```

Word Embeddings from Bill Titles

This task is more straightforward than extracting features from metadata. The bill titles are readily available in the XML, and are saved in the 'parliament.csv' file. Once the data is loaded into a pandas dataframe, we use a count vectorizer on the title strings to get word frequency counts, which we then transform into a term frequency document. No stopwords are used.

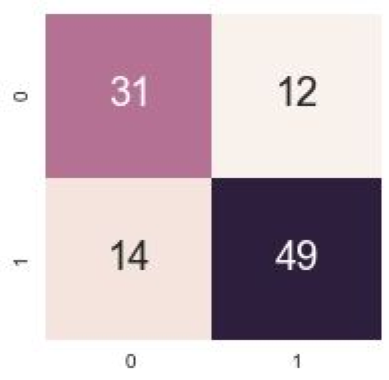
Implementation

Originally, the model was posed as a regression problem. This yielded results that were not very meaningful, as there was a lot of unexplained variation in the data. It wasn't clear, for example, how to interpret a result such as 40 days versus 140 days for a bill to pass into law. To address this, we instead converted the target variable into a categorical variable with two bins; 'fast' and 'slow'. WIth this we can aim to achieve results that are easier to interpret, while allowing us to answer questions such as the one posed by The Economist[6] due to the available confidence estimates for each prediction.

For each of our feature sets, we implement a logistic regression classification model. As most of the features for the first feature set are categorical, they are encoded using one-hot encoding using the pandas ‘get_dummies’ method.

The processed feature sets are split into a training and test set using the train_test_split module from sklearn with 20% being set aside as the test set and using a random_state of 0 so that we may reproduce results. Running the classifier on the first feature set gives an accuracy score of 0.75 and a roc_auc score of 0.75. This is a fairly good improvement over the benchmark model. The second feature set gives an accuracy and roc_auc score of 0.79 and 0.77 respectively.

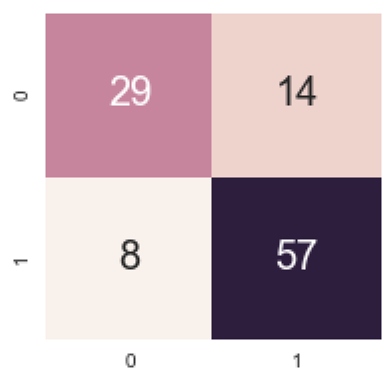
Refinements



We next attempt to tune the ‘C’ hyperparameter for logistic regression by performing a grid search and using the roc_auc as the scorer. This, however, does not yield any improvement in the model for the first feature set, as we observe exactly the same confusion matrix and scores as before we tuned hyperparameters.

As noted during the exploratory data analysis phase, there are a few features that we suspect are not helpful. We drop these features and observe the results.

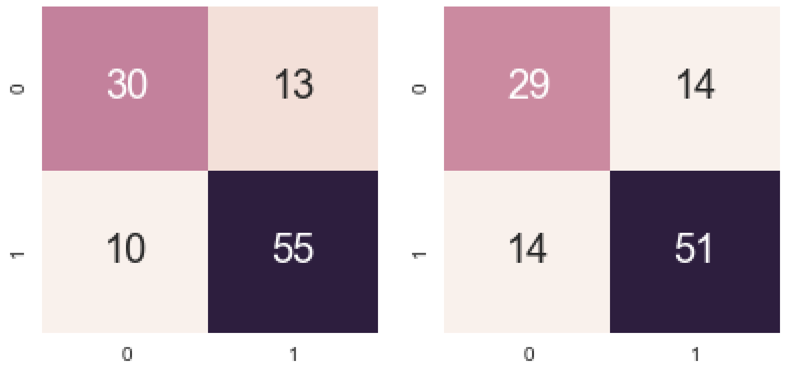
```
features_raw = features_raw.drop(['BillNumberOfPages','SponsorParty', 'ParliamentNumber',
'SponsorGender', 'SessionNumber'], axis=1)
```



This gives a modest improvement in accuracy and roc_auc scores at 0.796 and 0.776 respectively. The model with a reduced number of features is preferable because it reduces the model complexity while actually increasing performance.

Furthermore, this time when we perform the gridsearch exactly as before (searching over the values of the 'C' parameter) we do see an improvement and obtain a 0.80 accuracy score and 0.78 roc_auc score.

For the second feature set, the scores decrease after performing the grid search, which can be seen in the before and after confusion matrices shown below:



IV. Results

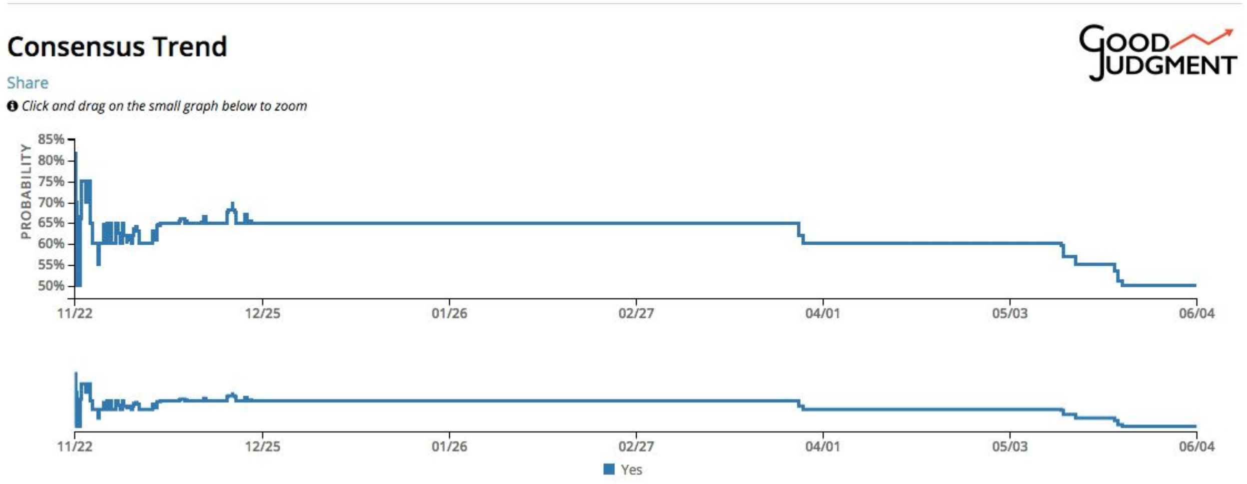
Model Evaluation and Validation

	Accuracy	ROC/AUC score
Benchmark	0.590	0.500
Model 1	0.755	0.749
Model 1 tuned (grid search)	0.755	0.749
Model 2 (reduced features)	0.796	0.776
Model 2 tuned (grid search)	0.806	0.783
Model 3 (bill titles)	0.787	0.772
Model 3 tuned (grid search)	0.741	0.730

As we can see, testing the model on the holdout set yields accuracies in the mid to high seventies. This demonstrates that we can have reasonable confidence that the model generalizes to new data. To be more thorough, however, it would be desirable to carefully compare scores on training and validation sets so that we can calibrate between overfitting and underfitting to the data.

One additional small data point to test the robustness of the model is to see what probability it predicts for a question posed by the Economist[6] on the Good Judgement Open website.

We can create a single observation from the data on the Parliament of Canada website[5]. The tuned second model which serves as our final model gives a probability that the bill is not a 'fast' bill of 0.832. To evaluate this result, we can compare it to the consensus trend provided by the Good Judgement website.



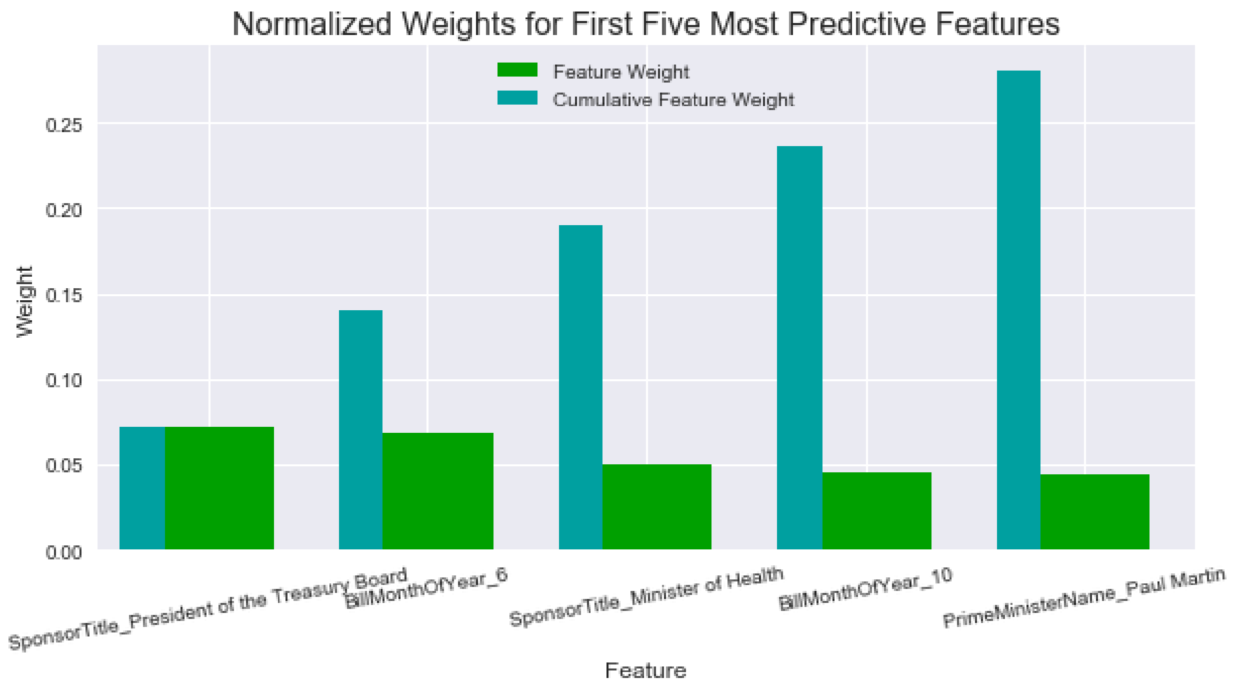
The consensus trend is an aggregate of hundreds of binary predictions to answer this particular question made by individual forecasters over time. The current probability that the bill will pass before the question period ends (that is, that the bill is a 'fast' bill according to our definition) is 50%. Furthermore, the trend seems to be rapidly decreasing. Considering the legal complexity of this piece of legislation, the crowd probability seems high, and in fact seems closer to the 60% probability one would obtain from the benchmark model around the time the bill was announced.

Justification

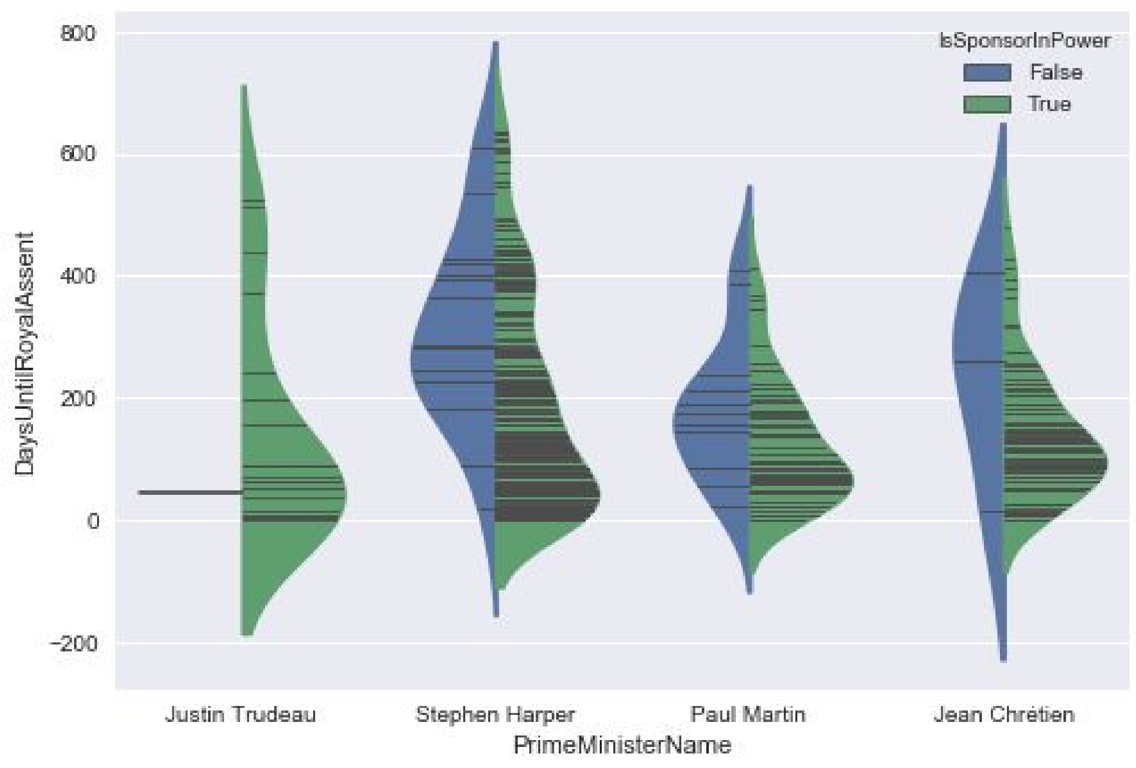
The model is highly dependent on perturbations of the train-test split of the data because the data set is small and can easily introduce unbalanced classes. Yet this is still significantly better than the benchmark model, and in the future more data points can be added, either by obtaining additional historical documents that are available in libraries, or by incorporating new data as new bills are passed in parliament. For such a simple out of the box solution, the model can be useful to provide a reasonable prior, with the caveat that relying too much on any single prediction for a significant piece of legislation with large economic and political consequences can be risky.

V. Conclusion

Free-form Visualization



The five most predictive features in the bill metadata are shown above. The sixth month, June, is the last month during which parliament is in session before summer recess, so it makes sense that it is an important feature. It was also observed during exploratory data analysis that the President of the Treasury Board as the bill sponsor strongly indicated that a bill would pass quickly. The feature that comes as a bit of a surprise is when the Prime Minister was Paul Martin, as he was only Prime Minister for a short while and had a minority government[9], yet he was able to pass bills fairly quickly. I'm a bit surprised that the party in power feature didn't appear in the five most important features, although by looking at the graph below we note that it may be because of the lack of data points for the False category.



Reflection

The workflow can be summarized as follows:

1. Obtain bill metadata in the form of XML files
2. Create preliminary pandas data-frame
3. Perform exploratory data analysis to investigate data and formulate hypotheses about which features may be safely discarded
4. Establish a benchmark model
5. Implement a classification model to improve on the benchmark

6. Use metrics to evaluate model performance and ability to generalize by testing on the holdout set

The most difficult part of the project was pre-processing the data; in particular, creating the feature for the number of pages of a bill. Many bills did not have the text content available as XML, so we had to create a script to download and read the PDF alternative. After that effort, it turned out that the this feature was not even useful, which I found counter-intuitive. But knowing this now allows me to focus my efforts on creating different features that may better capture what makes a bill complex for future development.

Improvement

The model could incorporate new information at various stages throughout the life of the bill. This would require training the model on all the available event data. As new event data becomes available for an unseen bill, it can be fed into the model to update a prediction.

Using the text content of a bill could also lead to more accurate results. It may be possible for a neural network to learn to encode features such as legal complexity using natural language processing techniques.

References

1. Yano, T., Smith, N. A., Wilkerson, J. D. (2012, June). Textual predictors of bill survival in congressional committees. https://projects.iq.harvard.edu/ptr/files/yanosmithwilkersonbillsurvival.pdf
2. Bill Prognosis Analysis. GovTrack.us. 2013
3. LEGISinfo, Parliament of Canada. http://www.parl.ca/LegisInfo/Home.aspx?Language=E&Page=1
4. Madelaine Drohan. Canada on a High, The Economist. http://www.theworldin.com/article/12607/canada-high
5. http://www.parl.ca/LegisInfo/BillDetails.aspx?Language=E&billId=8886269&View=0
6. https://www.gjopen.com/questions/354-will-canada-legalise-recreational-marijuana-nationally-before-1-october-2017
7. http://scikit-learn.org/stable/modules/calibration.html
8. Fawcett, Tom (2006); An introduction to ROC analysis, Pattern Recognition Letters, 27, 861–874.
9. https://en.wikipedia.org/wiki/Federal_minority_governments_in_Canada
10. Yaser S. Abu-Mostafa. Learning from Data, 88-94.