Understanding Public Perceptions of the Affordable Care Act: A Sentiment Analysis Approach Using Open-Source Data

Final Report

GitHub Repository: Link

Team: Aïcha Camara, Matt Jackson, Rohit Kandala, Summer Long

Table of Contents

Project Abstract

Using the Yelp Open source data, we sought to create a sentiment analysis model trained on the latter and addressed its effectiveness on Twitter API data from 2019 to 2023 to summarize and understand public perceptions of the Affordable Care Act ("Obamacare"). See Checkpoint 1 and Checkpoint 2 for further details on our exploratory data analysis, model creation pipeline, and best models. The results of our three models, Support Vector Machine, XGBoost, and Random Forest, showed that training models on Yelp reviews did not generalize well to public opinion data from Twitter, resulting in poor AUC-ROC scores.

Project Structure

Our GitHub repository is structured as follows:

```
ml-affordable-care-act/
├── data_wrangling
├── data
├── initial_analysis
├── model_parameter_testing
├── models
└── reports
```

Most of our work is in Jupyter Notebook files, which can be downloaded and re-run for reproducibility. The `data_wrangling` folder contains code that created our manageable samples, which are in `data`. Materials used to put together Checkpoint in `initial_analysis`; those for Checkpoint 2 are in `model_parameter_testing`. The trained models are in `models`, stored as .pkl files for reasonable compression[1], and the `reports` folder contains the actual reports for Checkpoints 1, 2, and 3 (including this one).

Data + Task Setup

As discussed in Checkpoint 1 and Checkpoint 2, we conducted an exploratory analysis on key terms in the Twitter data and the Yelp review data, then made a 100,000-review sample with binary classification based on star ratings, with 1 or 2 stars classified as positive and 3 to 5 stars as negative.

[1] The neural_network.pkl file was too large to be stored on GitHub, so it is uploaded to Google Drive [here].

Using the `scikit-learn` Python library, we then trained five models—Random Forest, XGBoost, Support Vector Machines (SVM), Neural Network, and Logistic Regression—on the Yelp data, employing grid search to find optimal hyperparameters.

The task then shifted to applying the best-performing trained, tested models to the Twitter data.

Manual Sampling of 50 Random Tweets per Group Member

We immediately had to ask: since the Twitter data is unlabeled, how do we evaluate the accuracy of the labels our model generates? Given the large size of the data (n=101183), manually classifying every tweet was untenable. So, using seeds for reproducibility, we created a human-manageable sample of 200 tweets (50 per group member). Each group member rated the 50 tweets that they sampled using a binary classification of "positive" or "negative." (Much like the 3-star Yelp reviews, neutral tweets were labeled "positive.")

Application of three models to the manually-sampled tweets

Since several of our models had comparable metrics (see Checkpoint 2, table, p.10), and we had a way to efficiently save and reload our models without training them from scratch (export as .pkl files), we decided we didn't need to settle on a "best" model *ex ante*, and instead train all of them for comparison against one another. This, in effect, took the TF-IDF weights from each model as trained on the Yelp data and applied them directly to the Twitter data.

We then counted up the model-generated labels for tweets in our manual sample and compared them to our human-ascribed labels. This lets us calculate evaluation metrics.

As discussed below, two of the candidate models (Neural Net and Logistic Regression) ended up having errors, so we removed them and proceeded with the other three for our final comparison.

Evaluation Metrics

We settled on AUC-ROC score as our primary metric because the data was heavily imbalanced. Our random sample had 77% of the positive sentiment class and 23% of the negative sentiment class. We didn't continue to report accuracy because it could be misleading: a model could achieve 77% accuracy alone by predicting the majority class 100% of the time.

We also kept precision, recall, and F1-score. Since these metrics display how the models are performing on both the majority and minority classes, it provides a balanced perspective on the performance of the model.

Experiment Results

Best XGBoost from Yelp Data Applied on Twitter Data

	Precision	Recall	F1-Score
0	.83	.36	.50
1	.39	.85	.53

XGBoost AUC-ROC: 0.6008547008547009

Best Random Forest from Yelp Data Applied on Twitter Data

	Precision	Recall	F1-Score
0	1.00	.01	.01
1	.33	1.00	.49

Random Forest AUC-ROC: 0.5037037037037038

Best SVM from Yelp Data Applied on Twitter Data

	Precision	Recall	F1-Score
0	.82	.43	.56
1	.40	.80	.54

SVM AUC-ROC: 0.6148148148148148

Sidebar: Neural Net and Logistic Regression omitted

Notably, our Neural Network and Logistic Regression models ran into bugs that made it hard to compare them against the other models.

The Logistic Regression encountered a dimensionality error; since the Twitter data did not include every term that the Yelp data did, there was a number-of-features mismatch. With Neural Network, the error was hard to decipher: "`TypeError: __randomstate_ctor()` takes

from 0 to 1 positional arguments but 2 were given." After consulting with ChatGPT, Google, and the TAs, we were unable to resolve the error.

Which model(s) performed the best? Why might this be the case?

All of the models had similar precision on the majority class of the Twitter data as they did on the Yelp test data. However, they were all *much* worse at minority class Twitter data, and had hideous recall and F1 scores. Random Forest was especially atrocious, labeling virtually *every* tweet as positive. The "least bad" AUC score, for SVM, was about 61.5%—a massive 35 percentage point dropoff from our performance on the Yelp training data, in which it hit 96.8%.

SVM may have been the "least bad" because the "kernel trick" (see Checkpoint 2) makes it especially adept at creating a nonlinear separation that fits the contours of the data. But given the performances all-around, the most fruitful question is not: "Which model(s) performed the best?" but rather: "Why did all the models perform so badly?"

Model Challenges: Some Hypotheses

Yelp data and Twitter data had major class imbalances in opposing directions

The sentiments in the Yelp data sample were imbalanced heavily toward the positive (77% positive to 23% negative; see Checkpoint 2). By contrast, our manually labeled Twitter sample was 32.5% positive and 67.5% negative. (Also notable: topic-level opinions were heavily against "Obamacare," though our analysis does not concern this directly. See "Avenues for Further Investigation" below.)

These opposing imbalances help explain why the model performance was drastically worse on the Twitter data. When a model that was trained on highly positive-skewed data is used to evaluate highly negative-skewed data, we expect it to still be biased towards predicting the positive class, resulting in performance declines. That is indeed what occurred.

Sarcasm/snark

Naive sentiment analysis of the sort we performed is not able to handle *sarcasm*, a rhetorical technique that reverses the expected direction of sentiment. Many of the tweets in our sample used sarcasm that went undetected by our models (e.g. "@BillOReilly I thought Obamacare already fixed all that." [index 42171][2]; we labeled this negative, but all three models deemed it positive[3]). Other tweets were ambiguous; e.g. a user who responds to news about a politician's

[2] Bill O'Reilly is a right-wing talk show host who has publicly opposed the ACA. We interpret this tweet, which uses "@" to notify O'Reilly's account, to be coming from a person who agrees with O'Reilly that Obamacare did not actually fix anything. See, *e.g.,* https://www.foxnews.com/politics/transcript-bill-oreilly-interviews-president-obama

[3] Our Random Forest model may be trivial in this instance, since it classifies virtually every tweet as positive.

health by tweeting "...Thank goodness he had Obamacare!..." [index 57218] might be sincere if that politician supports the ACA, or sarcastic if that politician opposes it.

Merely "informative" tweets

Some tweets expressed little sentiment of any kind, instead being merely informative (e.g. "Enrollment for Obamacare opens today to 12/15/20 https://t.co/S7qdER87fJ" [index 67528]). We defaulted to human-labeling these tweets as "positive", since they didn't display outright negativity, but the model did not always do so; the example tweet above was classified as "negative" by both SVM and XGBoost.

Additionally, some tweets seemed to use ACA hashtags to boost engagement, even though the actual law was largely irrelevant to the topic of the tweet (e.g. "Colorado nurse transforms Covid vaccine vials into a work of art to show appreciation for health care workers - CNN https://t.co/NRrJ1m6A6a #HealthCare #HealthInsurance #ACA #ObamaCare" [index 80494], which all three finalist models classified as "positive"[4]).

Ambivalent tweets

Some of the observations in our data were conflicted or ambivalent[5] (i.e. appeared to exhibit both positive and negative sentiment). This isn't just a Twitter issue: in the Yelp data (especially for 3-star reviews), some reviews described experience as extremely good in some aspects and extremely bad in others. In one funny example of a tweet that takes a turn, one user wrote "@GovernorSununu On point! Bernie Sanders is doing a *great* job of showing what a *disaster* ObamaCare is[6]" [index 22817, emphasis added]; we labeled this tweet as negative, but all three models classified it as "positive."

To best understand our model's outputs, we have to understand how it handles these cases. Some sentiment models, such as the VADER model included in Python's `nltk` library, start by giving separate scores for positive, neutral, and negative "valence", which are then combined into a single sentiment metric.[7] Preserving those valence scores could help us determine when and how positive sentiments might outweigh negative, or vice versa, within a single tweet.

Potential for human error / disagreement among labelers

For all the above reasons, we sometimes struggled to decide if a tweet was positive or negative. Even when sentiment words seem largely aligned, there is some possibility of inter-rater

[4] See previous footnote re: Random Forest.
[5] In recent years, it has become common in American English to use "ambivalent" as a rough synonym for "indifferent," i.e. neutral or devoid of sentiment in either direction. We are strictly using the older meaning of the term.
[6] Governor Sununu, a Republican, leads the state of New Hampshire. He opposes the ACA.
[7] https://www.nltk.org/_modules/nltk/sentiment/vader.html

disagreement or human error. For one possible source of such error: some of us redid our ratings to better disentangle the notion of "sentiment of tweet" from "sentiment toward the ACA," and could have under-updated our lists when making that change.

If we wanted to make our human labels more robust, we might have had every team member classify all 200 tweets in the sample, then used the majority vote (with a tiebreaker, since there are 4 of us) for the classification. We could also have picked a larger sample size; if we wanted to do that, or increase the chance of accurate classification through "wisdom of crowds" effects on the small sample[8], we could have used MTurk, social media, or personal acquaintances to get a larger set of volunteers.

Did "food review" context distort feature importance? A data-driven investigation of model features

It is glaringly obvious that *Yelp is not Twitter*; the sites' purpose, userbase, and discursive norms are very different, as are the rules that posts must follow (e.g. regarding post length). Therefore, we should expect substantial classification and generalization error when models trained on Yelp data are applied to Twitter data.

In particular, we suspected that the most highly-weighted Yelp-trained features would include many irrelevant terms related to food and restaurant service. If such terms drove much of the Yelp classification, but were relatively infrequent in the Twitter data, the resulting classifications of tweets might be noisy. We decided to investigate the top features to test our suspicions.

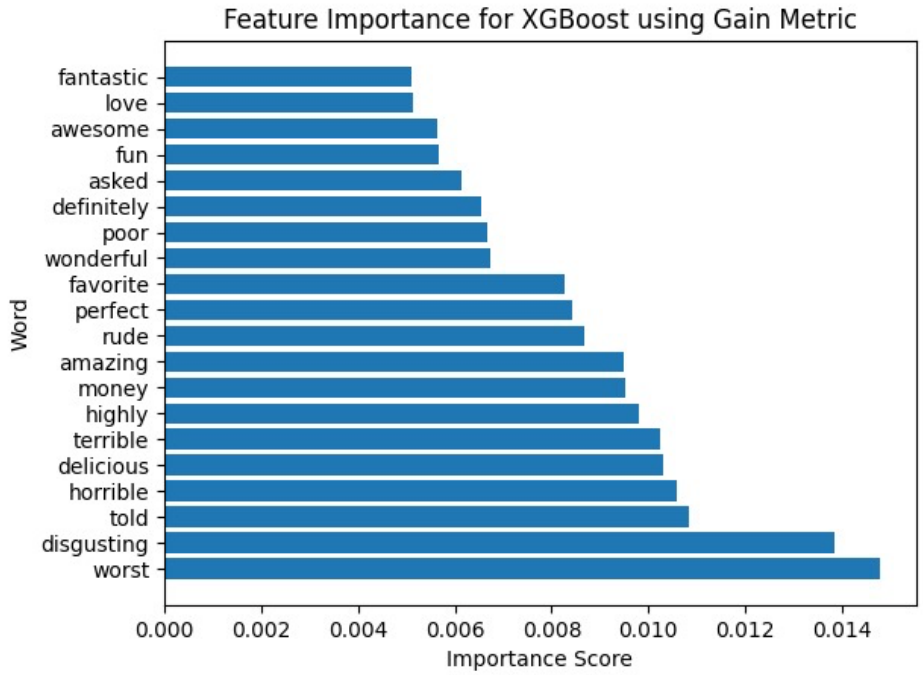
One issue we faced with our trained models is that in the process of creating our ideal hyperparameters, the TF-IDF method from `scikit-learn` had converted the corpus of text into numerical representations! We therefore needed to reconvert our most important features from numerical representations back into words. After doing so by using the tf-idf vectorizer's stored mapping of words, we used each model class's unique (albeit similar) metric[9] to extract its 20 most important feature words and the importance of each word. Here is a visualization for three of the models[10]:

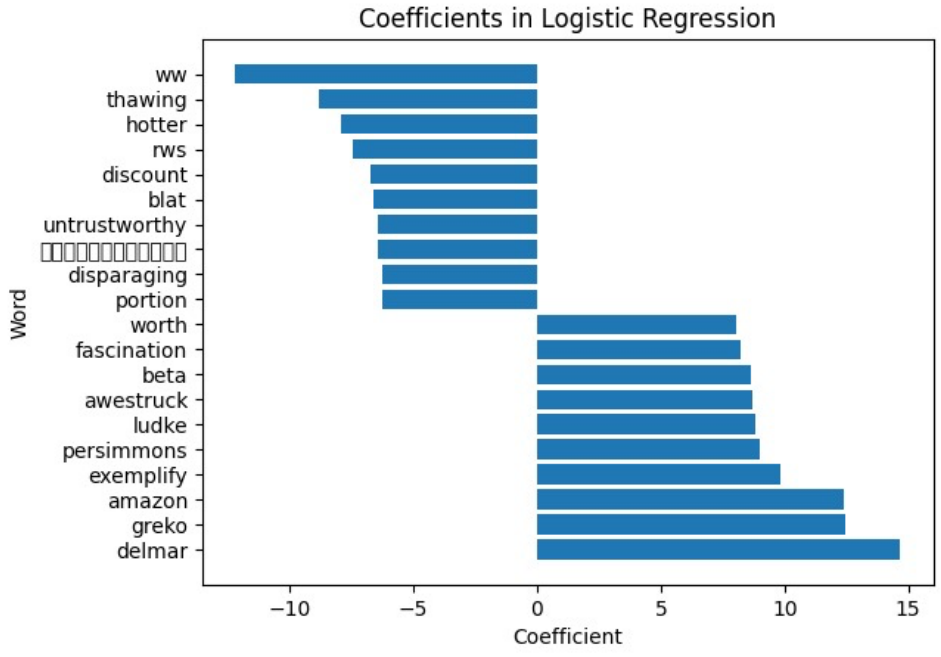
---

[8] The tendency for an average of statistically independent guesses from distinct individuals to be closer to the true value than the guess of a particular individual. See https://hbr.org/2018/12/the-right-way-to-use-the-wisdom-of-crowds

[9] To be brief: XGBoost's gain metric is the improvement of the loss metric when using that feature to split. The logistic regression coefficients represent the log-odds of the dependent variable being associated with the independent variable. The random forest Gini importance is calculated by the total Gini impurity reduction splitting on a particular feature across all trees in the random forest. Only logistic regression can indicate positive or negative skew of the feature.

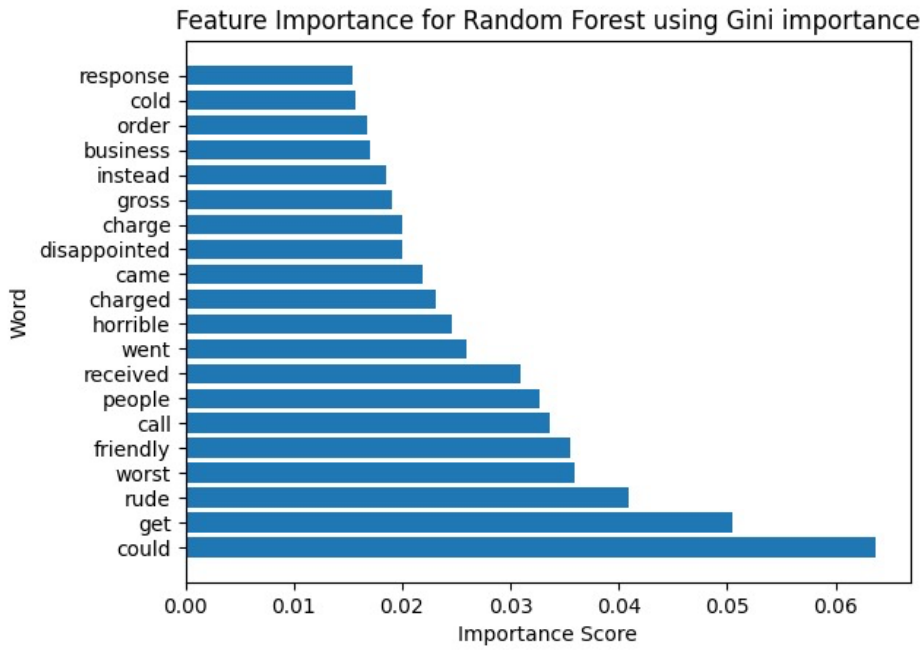
[10] It was impractical to determine feature importance for the SVM & Neural Net models due to the intense computation required (for instance, SVM was left to run overnight and had no indication of finishing soon at ~620 minutes). We believe that the general results would be similar for those models.





[11]

[11] The series of blocks in the Logistic Regression word axis, eighth from the top, correspond to the phrase "この店に向かうことにした", which is Japanese for "I decided to go to the store." `matplotlib` did not display the characters properly. The high weight of this feature is a good reminder that the Yelp data includes reviews in multiple languages, not just English.



Unfortunately, we were only able to break down features by positive or negative weight for one of the three models (Logistic Regression).

The most important features include food-specific reactions ("delicious"), adjectives for restaurant food ("thawing," "hotter") and waitstaff ("friendly," "rude"), and even a specific food noun ("persimmons"). Some generic words (such as "get", "could", and "told") also have high importance, which suggests that our initial list of stopwords could have been expanded further to include common verbs and verbal auxiliaries beyond the stopwords library in nltk.

These results confirm our initial hunch, suggesting that the Yelp-trained model had to classify tweets with "weaker" features, producing a classification that will shift drastically if even one "Yelp-y" high-importance term is present.

Avenues for Further Investigation

We leave off interested in many leads that we could have pursued in more depth with more time.

Topic-level opinion classification (pro- vs. anti-ACA)

The *direction of sentiment* in a Twitter user's rhetoric about the ACA (in terms of the mere positivity or negativity of the words used) does not correspond well with their *support or opposition to the legislation*. It takes semantic and contextual understanding to know that, e.g., a negative tweet attacking Republicans for attempting to repeal the ACA is supportive of the ACA proper. Models like ours, which do sentiment analysis at the word level, cannot gain that understanding. If we want to derive more complex insights about how political discourse varies

among supporters and opponents of the law, we could train a model that classifies tweets as "supports ACA" or "opposes ACA," but we'd need to take a completely different approach.

We also suspect—as we noted at earlier stages (see Checkpoint 1, *passim*; Checkpoint 2, pp. 2, 16)—that supporters and opponents of the Act use different terminology. Anecdotally, it seems that Democrats and supporters are more likely to refer to "the ACA" or "the Affordable Care Act", and Republicans and opponents are more likely to say "Obamacare."[12] **We were unable to test for these terminology differences** given the limitations discussed in the "Model Challenges" section above.

If we could train a model that classifies Tweets as pro- or anti-ACA, we could also train a model that does both kinds of classification at once, producing the following 2x2 matrix of subsets (each cell contains an example tweet that is, in our view, properly classed in that subset):

	Pro-ACA	Anti-ACA
Positive sentiment	"Yes! The #PatientProtection & #AffordableCareAct has already saved me more than $54,000 over 6yrs!! #ThanksObama, & Happy #ObamaDayUSA to @POTUS44, @BarackObama! https://t.co/NKCsKALY9f" [index 498 (*outside our 200-tweet sample)*]	"@RickSantorum praised .@POTUS Trump for leading on new #GOP #healthcare plan to replace #Obamacare: "We HAVE one. It's called the #HealthCareChoicesAct. We've been working on it for a year or so... There IS a bill, and it's a good bill...." https://t.co/gpImiacGP6 @KellyannePolls" [index 11692]
Negative sentiment	"@H3llfireSpecial @AngryJoeShow Obamacare: lived up to campaign promise. \r\nReplacing Obamacare with something better: Absolute bullshit. It was ready in "two weeks" so many times." [index 65475]	"ObamaCare is the greatest threat in our great nation to the people. Biden claims not good "oil crisis" is worse. DISGRACEFUL!" [index 80233]

If we went even further and addressed our issue with sarcastic tweets by building a working "sarcasm detector," we could add a third dimension (sincere vs. sarcastic) to get octants. We could then assess the relative size of each octant, and run data analysis within and across octants, deriving insights by, e.g. comparing term frequencies.

[12] In our 200-tweet sample, "Obamacare" is by far more prominent; it had 195 instances of "Obamacare" and just 26 distinct instances of "ACA".

Filter down to more representative training data

Assuming we are constrained to using only Yelp data, it might make sense to use our knowledge of Twitter to filter down to a smaller sample of Yelp reviews that are as "Twitter-like" as possible.

For example, Yelp reviews tend to be much longer than tweets[13]; we could limit our sample to those reviews that are approximately 280 characters, which was the length limit for tweets for most of the time spanned by our Twitter data[14]. We could also oversample (or exclusively sample) extreme content, e.g. by restricting ourselves to 1-star and 5-star reviews, to better sample the often hyperbolic tone of political tweets.

We could also "de-food-ify" the data, either by filtering out reviews with too many restaurant-related terms, or by adding food terms to our list of stopwords. This could be automated, e.g., by reducing the Yelp data to the set of its constituent words, using some kind of topic modeling to find the subset that are "food words," and adding those to the stopwords that the `nltk` library uses to clean data before training. We could also have made use of the optional `min_df` parameter in the `TfidfVectorizer` class, which would remove the least frequent words (such as rare foods) from our feature set.

Non-binary classification

We might also want our classification schema to have a "neutral" label, which would make the "positive" and "negative" models more substantively meaningful and reduce the difficulties we faced as humans labeling the Twitter data. Given the Yelp review setup, the easiest way to do this would probably be to classify all 3-star reviews as "neutral."

We could also subdivide further, with an "informative" neutral class for tweets that are effectively devoid of sentiment-laden terms, as distinct from "ambivalent" tweets with both positive and negative sentiment (as determined by a multi-valent underlying scorer such as the one in VADER).

Features beyond single words (n-grams, etc.)

Some words have different valence when used in combination. (For a toy example, the word "good" is typically assumed to have positive sentiment, but put it in the 2-gram "not good" or the

---

[13] An analysis as of 2015 suggests that the median Yelp review is 91 *words* long, and the mean value is 125 *words*, with a sizable long tail of longer reviews (https://rstudio-pubs-static.s3.amazonaws.com/127956_5202598cb2aa4736bfeceffdf613c932.html#/1). English text typically averages "4.79 letters per word" (http://norvig.com/mayzner.html), suggesting that an "average" Yelp review has somewhere in the range of 400 to 600 characters.

[14] Starting in February 2023, Twitter has allowed paying subscribers to post tweets of up to 4,000 characters in length. https://twitter.com/TwitterBlue/status/1623411400545632256?lang=en

3-gram “not very good” and the magnitude and direction of sentiment shift.) Our method of feature generation separated all text at the word level; a different approach to training that allowed for n-grams might produce more apt results.

Sentiment change over time, 2019-23

In the initial parameters for the project, and our exploratory analysis for Checkpoint 1, we hoped to assess how sentiment about the ACA changed over time from 2019 to 2023. We chose to deprioritize this question, and thus did not end up with significant results to share about such putative trends. One reason for our choice is that our model did not input the date of a Yelp review as a feature, or factor in the date of a Tweet when classifying it. Another is that our 200-tweet sample was not large enough to get a representative sample for each reasonably small unit of time (each month, perhaps, or each quarter).

In a quick manual assessment from browsing the Twitter csv, we don't notice much change in sentiment between 2019 and 2023. (It’s overwhelmingly negative throughout.) This makes some amount of sense, since the law didn't undergo any major changes (the last major adjustment was the GOP congress setting individual mandate to $0 in the Dec. 2017 Tax Cuts and Jobs Act,[15] and the most recent Supreme Court case attempting to invalidate the law, in 2021, resulted in no change[16]), and stayed a contentious, high-saliency issue for the duration of the data. Given the limited time we had and the large number of models we were already comparing on several other metrics, we let go of trying to test hypotheses about pretty subtle diachronic trends.

Conclusion

Machine learning techniques are quite powerful. One *could* say, as then-Vice President Joe Biden said about the Affordable Care Act on the day President Obama signed it into law, that they are “a big f---ing deal.”[17]

That said, we cannot let our enthusiasm for ML overwrite our common sense or our understanding of basic linguistic context. We should not be so convinced that “ML conquers all” that we think of all messages as interchangeable, when some are medium-to-long business reviews and others are short-form tweets about a public program. To put it in the discipline’s terms, we should expect substantial **generalization error** and present our results with humility when that error (expectedly) occurs. (We should also be highly skeptical of any future client, boss, or hype-monger who presents machine learning methods as a panacea, or expects near-perfect AUC-ROC on a corpus with completely different norms than the training data.)

[15] https://www.kff.org/health-costs/issue-brief/how-repeal-of-the-individual-mandate-and-expansion-of-loosely-regulated-plans-are-affecting-2019-premiums/

[16] https://www.reuters.com/business/healthcare-pharmaceuticals/us-supreme-court-rejects-republican-challenge-obamacare-law-2021-06-17/

[17] https://www.theguardian.com/world/richard-adams-blog/2010/mar/23/joe-biden-obama-big-fucking-deal-overheard

At minimum, it takes much more clever feature engineering to reduce generalization error to an acceptable level. For three of us, this was our first-ever attempt at engineering models of any kind. While we're disappointed that our models didn't work as well as we hoped, we gained a lot from the experience. We look forward to becoming more clever as we go on to study more advanced ML techniques, keeping in mind the varied contexts in which we plan to apply them.