Fed Funds Rate Machine Learning Prediction Project Using NLP

Michelle Bonat
Last Updated August 19, 2019

Overview

- **Problem to be solved:**

  The problem we are looking to solve is to predict federal funds rate changes based on analyzing the transcripts of the FOMC (Federal Open Market Committee) meetings, using machine learning, starting with NLP. We may also potentially leverage related economic data sources and additional ML models to get to our target results of a better prediction than the futures market.

  Background:

  The fed funds rate is one of the most important interest rates in the U.S. economic system. Simply stated, it is the rate at which banks lend money to other banks for a single day. Yet it's impact is very broad. It directly affects everything from short term interest rates (the return on your checking account), to longer term interest rates (such as for credit cards, auto loans, and mortgages). Indirectly but as importantly, the fed funds rate influences the economy's employment, inflation and growth.

  The fed funds rate is set by a committee called the FOMC which meets eight times a year to set the rate. At their meeting they decide to keep it the same, move it up, or move it down. These meetings happen behind closed doors however the transcripts are published later for public consumption.

  The fed funds futures market is a financial representation of the opinion of where the fed funds rate will be in the future. These futures can be traded as far out as 36 months.

- Audience:

Investors and other players in the markets keep a very close watch on the fed funds rate, and there is much speculation over when any rate changes will happen and what they might be. Any changes in the rate have immediate and potentially broad effects on the financial markets, and by extension the U.S. economy and even the world. If these players could accurately predict these changes in advance they could make better decisions about investing, trading, business and more.

- Data:
    - FOMC meetings minutes transcripts accessed at federalreserve.gov on this page, spanning 1936-present. Currently The Federal Open Market Committee (FOMC) meets eight times a year to determine the federal funds target rate.
        - Minutes (1993-Present)
            - Summary of issues addressed at each FOMC meeting; 210 transcripts.
            - May 1999 is when the Committee began releasing statements consistently after each meeting. **Meetings before May 1999 that may not have published minutes.**
        - Record of Policy Actions & Minutes of Actions (1967-1992)
            - Precursor documents to the modern minutes
        - Historical Minutes (1936-1967)
            - Records of discussions and decisions at FOMC meetings
    - Actual Fed funds rates history back to 1954 (info here).
    - Fed funds rate changes are on a web page here it goes back to 1990 (not a dataset, have to scrape).
    - Potential Additional DataSets:
        - Mining text from fed reserve press releases that relate to monetary policy (info here).
        - Mining text from relevant fed reserve speeches (info here).
        - Other economic indicators as needed. Additional federal reserve economic data can be found here.

- Solution:

  Our plan is to initially craft this as an NLP-based exploration with a goal of getting the prediction above the recent prediction accuracy rate of the fed funds

futures market (76% accuracy over the period 1994-2000). If we are not able to beat the prediction record of the futures market, and if time allows, we will look to incorporate additional data into the model such as related economic data.

Deliverables:

- Code on GitHub in a public repo
- Slide deck
- Blog post (nice to have)

Other Comments:

It was suggested that we could initially focus our analysis on classifying the FOMC minutes as dovish or hawkish and building from there. This is something we plan to explore during the project. A monetary hawk, or hawk for short, is someone who advocates keeping inflation low as the top priority in monetary policy. In contrast, a monetary dove is someone who emphasizes other issues, especially low unemployment, over low inflation (Wikipedia).

More background:

- "More than two decades ago, in February 1994, Chairman Greenspan issued the first postmeeting statement following the FOMC's decision to tighten monetary policy--the first increase in the target for the federal funds rate since 1989.
- For the next five years, a statement was released only after FOMC meetings at which the Committee changed its target for the federal funds rate.
- In May 1999, the FOMC began releasing statements at the conclusion of every meeting regardless of whether it had decided to make a change to the policy interest rate or not.
- Later, in May 2002, the FOMC decided to provide voting information in the postmeeting statements, including a short explanation of any dissenting votes, thereby accelerating the provision of this information to the public by several weeks." --- from Fed report.

Data Wrangling

This section reviews the steps we took to wrangle the data for this FedFundsML project. It includes:

- Data acquisition
- Data cleaning
- Data wrangling specifics

Data Acquisition

Data sources:

This project uses 3 sources of data:

- FOMC meetings minutes transcripts accessed at federalreserve.gov on this page, spanning 1936-present. Currently The Federal Open Market Committee (FOMC) meets eight times a year to determine the federal funds target rate.
- Actual Fed funds rates history from 1954 - 2008 on a daily basis (info here).
- For Fed funds rates starting in Dec 2008 the rates are listed as a target range.
- Fed funds rate changes are on a web page here it goes back to 1990 (not a dataset). Starting in Dec 2008 the rates are listed as a target range (i.e. 0.5 - 0.75).
- There is no rate change data for the period in between December 16, 2008 and December 16, 2015. This is because the rate did not change during that time.

Data ingestion:

Once these data sources were identified, they were connected to the project.

- Meetings minutes were scraped from the FOMC website using a third party open source script that I adapted for Python 3. This can be found in the PullStatements.py file.
- Rate history was downloaded from the web and then augmented by hand with the most recent periods, as this was on a separate page and was

only a few periods. Once downloaded, this file was imported into the application. For the periods where rates were listed as a range, the average is calculated and listed as the single rate. This can be found in the fed_funds_target_rates_Dec2018_Sep1982.csv file.

Data Cleaning

Once imported into the project, steps to clean the data includes these things:

- Read in the statement and convert it to lower case
- Remove punctuation and newlines first, to keep space between words
- Keep only the characters that you want to keep
- Remove anything before (and including) 'for immediate release'
- Remove stop words

A note on stop words. I debated on how to treat stopwords: whether to remove stop words, to modify what stop words were removed, or not to remove any stopwords to start with. The reason is that some of the traditional stopwords may signal intent for rate changes in different ways. For example: won't or will. In v1 of this project I ended up removing stop words using the standard mcdonald_comb dictionary to gage baseline effectiveness, with an option of modifying this stop list later.

Data Wrangling

There are several steps that happen in the wrangling portion. This is to address that we are using two separate datasets that have inconsistent dates. This means that different events are recorded on different dates i.e. the meeting transcript is released following the meeting, and the rates are recorded on a daily basis earlier in the cycle and then just as a rate change later in the cycle. So, I needed to address the way dates are recorded since they vary between the two datasets.

These are the data wrangling steps (see the jupyter notebook data.ipynb):

Fed funds rate data:

- Add a column denoting rate changed or not from previous month
- Add a new column that denotes how much the rate changed from the previous month
- Add a new column to denote if a rate changed based on the previous month (1 is changed; 0 is did not change)
- Modify date string to a timestamp in format

Text data:

- Create a dataframe that includes the cleaned text doc content and adds a column for the date of the document taken from the file name
- Create a new column indicating if a rate changed within x days (7) after document published
    - To do this, for each row in df subtracts the dates, gets the minimum difference row for
- Rate_change df and gets the corresponding date and rate change and adds those as fields
- Indicate if the rate changed within a certain period from the document being published.
    - There were a couple different ways were investigated to accomplish this. Ultimately we ended up using Pandas for a merge_asof to merge the two dataframes (rates and documents). We used the tol (tolerance) parameter to indicate the number of days. At this writing we're using 7 days.

Statistical Analysis

This section discusses the most relevant statistical inference techniques for this project, details the approaches used, and explains the results.

The approach to statistical data analysis for NLP projects if often quite different from the analysis with non-NLP projects.

We found that to be true with this project. For example, other projects often look at ways to analyze continuous and discrete variables. In our project we have continuous data (the rates) and unstructured data (the document content).

RATES

We start with some Univariate analysis (explore variables 1 by 1) with Rates:

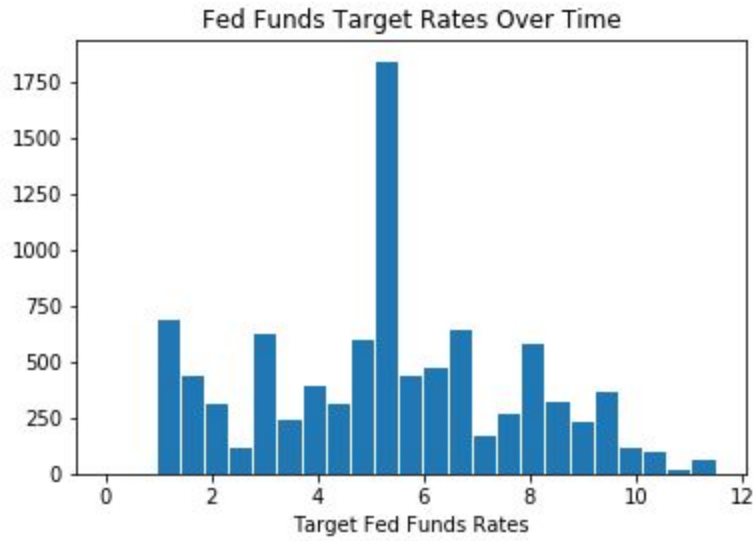
Central tendency and spread of the Rate variable:

Mean: 5.35
Median: 5.25
Min: 0.0125
Max: 11.5

Conclusion: The rates have quite a large range of values. The minimum rate is very close to zero. The mean is similar to the median.

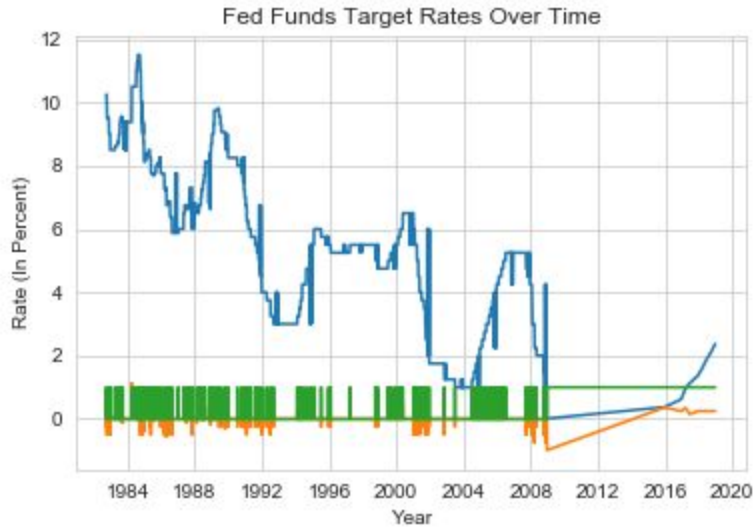
Next we did some visual EDA (exploratory data analysis) on Rates.

How the rates look as distributed by value (histogram):



Conclusion: We can easily determine from this visual that there are many months of rates that are between 5 and 6 percent. There are not many at all of the higher rates. The number of occurrences of the lower rates vary quite a bit.

Rate distribution over time as a Plot:



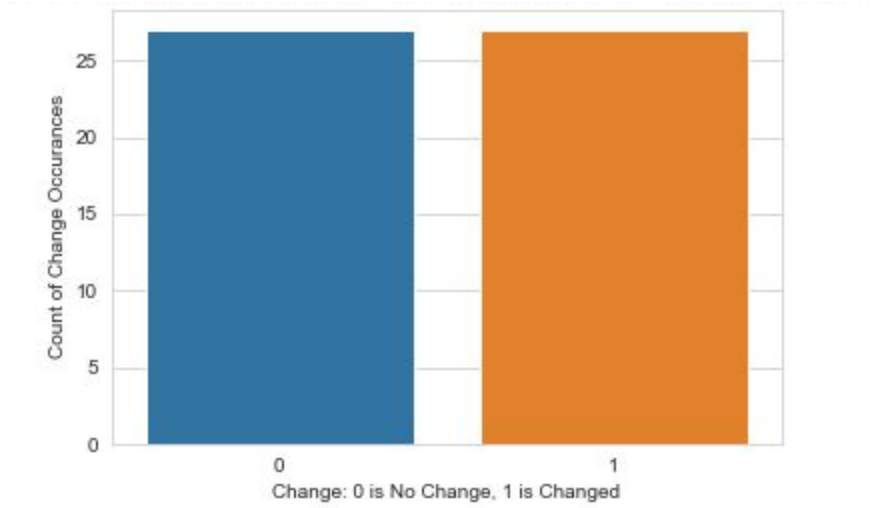
Conclusion: This is really interesting, it tells a lot that we did not see from the previous analysis. This makes it obvious that rates were generally higher in earlier years starting

around 1984, and have been generally decreasing since then, until 2016 when they started increasing again.

Occurance of Change Incidents

Next we looked at the periods where rates changed compared to where they did not change. This is visually depicted below using a bar graph. It visually highlights that the count of no-change periods compared to change periods is pretty similar.

Comparison of Times Fed Funds Target Changed (or Not) Within 7 Days of Document Date



DOCUMENTS

Documents Analysis

Next, we looked more into the documents data. We learned from the previous Word Cloud analysis that indeed a few words were most used, and we depicted that visually.

We wanted to dig a bit deeper and see what were the most frequently used words with a quantitative metric. Note that Stop words have already been removed from this corpus, so you can see it does reflect more useful words here. These are listed in alphabetical order.

Top 20 most common words in the documents corpus:

accommod

act

believ

commit

continu

decid

econom

fedfundsr

fomc

grow

infl

long

perc

policy

pricest

produc

remain

risk

target

today

Conclusion: It is interesting to see that the most frequently appearing words are both factual (fomc, target) and emotional/non-factual (believe, risk). This could be interesting to explore from a machine learning perspective as a predictor for action.

Machine Learning

This section discusses the most relevant machine learning techniques for this project, details the approaches used, and explains the results.

When people talk about the machine learning algorithms used for NLP projects, Naive Bayes (NB) is often on the short list.

Why is Naive Bayes so good for NLP?

- Great model to start with, sometimes works well, depends on use case
- Fastest model fast compute time
- Based on conditional probability
- Easy to interpret features, most importantly.

The hypothesis for this project was that NB would be useful, and that we would run that and compare against several other models.

To prepare for the machine learning models we took these steps:

- Split the dataset into a training and test set with a 70/30 training/test ratio.
- Then vectorized and transformed the data.
  1. CountVectorizer (BagOfWords type approach)
  2. TFIDFvectorizer (BagOfWords type approach)
- Note: The train/test split was done before the vectorization and transformation. This is in a different order than some projects perform these steps - many do it after. We did it this way because it results in a cleaner set of data on which the machine learning models can perform their analysis.
- Note: The bag of words treatment doesn't preserve information about the order of words, just their frequency.

Next we ran NB model on the data.

- We are only using the text data here for the X, and the 'did the rate change from previous period' for the predict.

For NB we got these results:

```
accuracy_score(y_test,predicted)
```

```
0.9411764705882353
```

```
precision_score(y_test,predicted)
```

```
0.8888888888888888
```

```
recall_score(y_test,predicted)
```

```
1.0
```

```
f1_score(y_test,predicted)
```

```
0.9411764705882353
```

This is a solid accuracy score of 94%. The good initial result was helped by our earlier feature engineering work and our domain knowledge in financial services.

For this project, we cared more about avoiding false positives (predicting a rate change when it does not change “false alarm”), than about false negatives (predicting a rate does not change and it does change). A higher Precision score here is better. Precision is when predict yes, how often correct?

Next, we added in three more machine learning models:

- Logistic regression
- SVM (Support Vector Machine)
- Decision Trees

To optimize these models we included Standard Scaler.

We also built a pipeline to handle these new models all together.

We did run Count Vectorizer and TF-IDF again with these models so the output from that was directly fed into each of these models.

We built a compare function that prints the test data accuracy of all the models and then outputs the model with the best (or tied) results.

The pipeline output of all these models running together looks like this:

The model results grouped by model:

Naive Bayes pipeline train accuracy: 0.973

Naive Bayes pipeline test accuracy: 0.941

Logistic Regression pipeline train accuracy: 1.000

Logistic Regression pipeline test accuracy: 0.882

Support Vector Machine pipeline train accuracy: 1.000

Support Vector Machine pipeline test accuracy: 0.882

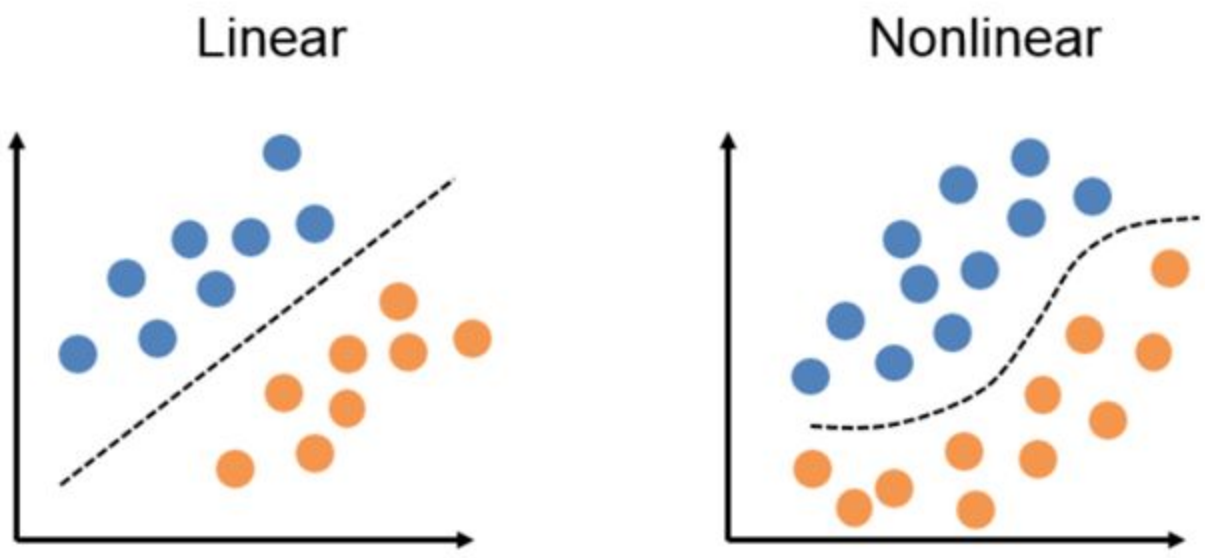
Decision Tree pipeline train accuracy: 1.000

Decision Tree pipeline test accuracy: 0.941

Classifier with best accuracy on test data (or tied): Naive Bayes

NB is the winner! We can see that the other models are overfitting since their train accuracy is very high (1.0) but their test accuracy is much lower (0.82-0.94).

Why is NB a better model here?



The non-linear models are the best fit for this data (Naive Bayes, Decision Trees).

In the future, we would also like to add an analysis of strongly predictive features. This approach uses the trained classifier to make predictions on this matrix, by sorting the rows by predicted probabilities, and picking the top and bottom rows.