Logistic Regression using Bayesian Inference to Identify Stock Market Movements

STAT 5630

Dr. Brian Zaharatos CU Boulder

Adam Sanchez and Sandra Tredinnick

1 Overview and Description of Project

Logistic regressions are a common place in predictive analyses of categorical (or more commonly specifically binary) outcome data. For instance, given a set of stocks' daily returns, will the stock market as a whole move up or down? Similar to other standard regressions, the parameters of the model are often estimated using Maximum Likelihood Estimation or other numerical methods. In this paper, we take a Bayesian approach to estimating the parameters of a logistic regression involving a set of 5 stocks' daily returns and a binary variable representing the movement of the SPDR S&P 500 ETF Trust, from January 3rd 2000 to April 14th 2014. We found interest in this problem as the stock market effects everyone, thus making it beneficial to be able to explain and predict how the stock market moves on a given day.

First some terminology. A stock's "return" is defined as the difference between its closing price (the stock's price at the end of the day) and its opening price (the stock's price at the beginning of the day). An "ETF" or exchange traded fund is "a type of security that tracks an index, sector, commodity, or other asset, but which can be purchased or sold on a stock exchange that is the same as a regular stock" (Chen). As mentioned above, we use the SPDR S&P 500 ETF Trust (SPY) which tracks the S&P 500 index. Specifically, the SPY is "a market-capitalization-weighted index of the 500 largest publicly-traded companies in the U.S," and is often used as a benchmark for the whole stock market. We define a target variable $\mathbf{Y}$, which will be a vector of 1's and 0's, indicating the movement of SPY, i.e 1 if return is greater than zero, otherwise it is 0. Beyond the estimation of the parameters in a logistic regression, we hope to estimate and provide a distribution for the probability that SPY $= 1$ on any given day.

Our data set is comprised of the daily return indicator of SPY, $\mathbf{Y}$, and 5 stocks' (all members of the S&P 500 index) daily returns: Microsoft Corporation (MSFT); NVIDIA Corporation (NVDA); UnitedHealth Group (UNH); Johnson and Johnson (JNJ); and JPMorgan Chase and Co (JPM). The daily returns of each of the 5 stocks' are organized into a matrix $\mathbf{X}$. For clarification, each entry in $\mathbf{X}$ represents the daily returns for each stock on a single day. The data was sourced from Yahoo! Finance, clearly making it observational; we arrived at the set of 5 stocks through a combination of holistic and mathematical processes which we will discuss in section 2.

As a review, recall that in a logistic regression, if we wish to predict the value of $\mathbf{Y}$ at time i, we have:

$$\hat{y}_i = \frac{\exp\left\{\hat{\beta}_0 + \sum_j^n \hat{\beta}_j X_{i,j}\right\}}{1 + \exp\left\{\hat{\beta}_0 + \sum_j^n \hat{\beta}_j X_{i,j}\right\}}$$

where $\hat{\beta}_0$ is the estimate of log odds of success when all predictors are equal to zero and $\hat{\beta}_j$ is the estimate of the logistic regression parameter for stock j. Estimation of these $\hat{\beta}$'s will be the basis of inference. As previously mentioned, they are often calculated using frequentist methods, such as OLS or other MLE methods, and only provide point estimates for the parameters. In this project, we use Bayesian inference and place distributions over our parameters, β. In section 3, we formally introduce our modeling procedure; we utilize the Python package PyMC3 to implement a variation of the traditional Markov Chain Monte Carlo sampling algorithm, which provides estimates of the β coefficients (thus our logistic regression model) by sampling from their respective marginal posterior distributions.

The rest of this paper will proceed as follows: in section 2 we describe our data collection and analysis process, section 3 describes our modeling procedure, in section 4 we provide a discussion on key results, and finally in section 5, we provide a discussion on implications of our results and possible extensions.

2 Data Collection and Analysis

Mathematical analysis of financial data has been a staple since at least the 1980's. As such, financial data is widely available and there are many options for sourcing stock price data. One such source is Yahoo!

Finance, who report daily stock information including, but not limited to, closing price, opening price, volume of shares traded that day, and current price of the stock.

Given a list of all the companies in the S&P 500 (which can be found on Wikipedia), we scrap the Yahoo! Finance website using a Python script we developed to return the price of the stock for each day between January 3rd 2000 and April 14th 2014. Formally, a stocks' daily return is given by

$$r = \text{closing price of stock} - \text{opening price of stock}.$$

All the return data are stored in a large data set, composing of over 2,500,000 individual data points.

With such a large data set, it is necessary (for both ease of computation and ease of understanding) to subset this data. Fortunately, this task is made easier by the nature of construction of the S&P 500. As mentioned earlier, the index is a weighted collection of the 500 largest publicly-traded companies in the U.S., so clearly the companies with the highest weight will have the greatest effect on the index as a whole. This information is widely available and we only considered 5 companies with the largest weight. This reduction in the number of stocks we use results in a total of more than 27,500 individual data points, which correlates to more than 5,000 rows of daily returns.

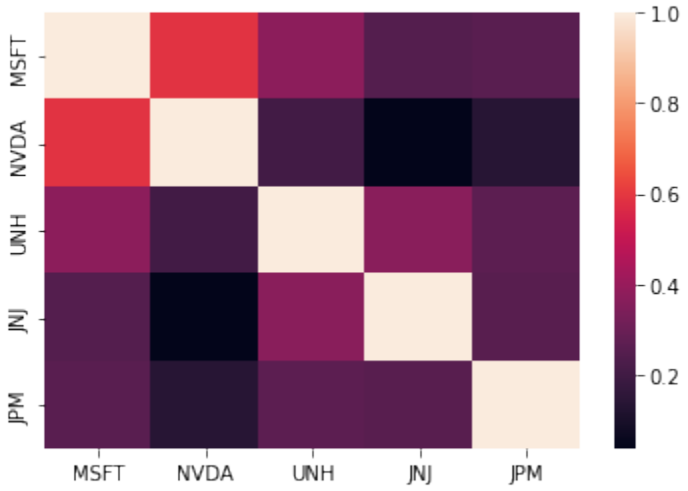

Figure 1: A correlation plot between MSFT, NVDA, UNH, JNJ, and JPM.

Similar to frequentist methods, we ideally want our explanatory data to have as little correlation as possible. Thus, for each of the 5 stocks with the highest weight, we calculate the correlation between their daily returns. Balancing the need for higher weight companies, we arrive at a list of companies: Microsoft Corporation (MSFT); NVIDIA Corporation (NVDA); UnitedHealth Group (UNH); Johnson and Johnson (JNJ); and JPMorgan Chase and Co (JPM). Fig. 1. depicts the correlations between each of the aforementioned stocks' daily returns, where the lighter the color, the higher the correlation. Distinctly, the largest correlation occurs between MSFT and NVDA, which is not surprising considering both are technology companies. That being said, we believe this slightly higher correlation will not be an issue as it is almost impossible to find companies with no correlation.

With these 5 stocks chosen, we arrive at our final data set:

$$\mathbf{X} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & r_{1,4} & r_{1,5} \\ r_{2,1} & r_{2,2} & r_{2,3} & r_{2,4} & r_{2,5} \\ r_{3,1} & r_{3,2} & r_{3,3} & r_{3,4} & r_{3,5} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ r_{n,1} & r_{n,2} & r_{n,3} & r_{n,4} & r_{n,5} \end{bmatrix} \end{array}$$

where $r_{i,j}$ is the daily return of stock j on day i.

We now turn our attention towards our indicator variable $\mathbf{Y}$. Formally, we define the value of $\mathbf{Y}$ on day i as

$$y_i = \begin{cases} 1 & \text{SPY Return on Day } i \geq 0 \\ 0 & \text{SPY Return on Day } i < 0. \end{cases}$$

As with the data collected for X, we collect SPY data via Yahoo! Finance and the python script we described above. Once the daily returns are calculated for SPY, we calculate $\mathbf{Y}$.

3 Modeling Methodology

Previously mentioned, the basis of our problem will be inferring the parameters of a logistic regression model in order to best predict our return indicator of SPY, $\mathbf{Y}$, based on the daily returns of the 5 stocks which make up, $\mathbf{X}$.

The parameters we are trying to infer can be denoted as:

$$\boldsymbol{\beta} = [\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5]$$

Where β_0 is the intercept, and the β_i's are the coefficients for $i = [1 = \text{MSFT}, 2 = \text{NVDA}, 3 = \text{UNH}, 4 = \text{JNJ}, 5 = \text{JPM}]$. Note that β_0 is interpreted as the log odds of success when all other daily stock returns are equal to zero. In addition, recall that a one unit increase in one stocks' daily return (with all other stocks' daily returns held constant) increases the log odds of success by β_j. Here, the log odds of success represents the probability that y_i=1. As is often the case for Bayesian inference problems, our goal is to generate a posterior distribution from which to sample these parameters; consider

$$\pi(\boldsymbol{\beta}|\mathbf{Y}, \mathbf{X}) \propto f(\mathbf{Y}|\mathbf{x}, \boldsymbol{\beta})\pi(\boldsymbol{\beta}), \tag{1}$$

where $\pi(\boldsymbol{\beta}|\mathbf{Y}, \mathbf{X})$ is our desired posterior distribution, $f(\mathbf{Y}|\mathbf{X}, \boldsymbol{\beta})$ is the likelihood function for our data, and $\pi(\boldsymbol{\beta})$ is the prior distribution over our parameters.

In order to estimate our posterior, we utilize the convenience of a standard python package called PyMC3, which focuses on advanced Markov Chain Monte Carlo (MCMC) and variational fitting algorithms. More specifically, we utilize PyMC3 to sample from our posterior via an MCMC type algorithm (although it is not exactly an MCMC algorithm) called NUTS. We will discuss NUTS in greater detail in a subsection below.

3.1 Likelihood and Prior

Given our data is composed of 5 observations with one binary target variable for each day, it is natural to consider each daily observation as an independent random experiment with only two outcomes; SPY goes up, or SPY does not go up. Thus, each day we have a Bernoulli trail, and so it follows that a logical likelihood function is the product of T such Bernoulli trials, assuming independence among the experiments. That is to say:

$$f(\mathbf{Y}|\mathbf{X},\boldsymbol{\beta}) = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i}$$

where p_i is the probability that $y_i = 1$ on day i and is given by $p_i = \frac{1}{1+e^{\eta_i}}$. Here $\eta_i = \beta_o + \beta_1 \text{MSFT} + \beta_2 \text{NVDA} + \beta_3 \text{UNH} + \beta_4 \text{JNJ} + \beta_5 \text{JPM}$. By default PyMC3 uses this likelihood.

We now consider the selection of an appropriate prior for our model. By default, PyMC3 uses a multivariate $\mathcal{N}\left(\mathbf{0}, \mathbf{\Sigma} = 10^{12}\mathbf{I}\right)$ distribution where $\mathbf{I}$ is the 5×5 identity matrix. Clearly this is a very vague prior and relative flat. While these non-informative parameters may be useful when we truly know very little about our data, a more informative choice may do better. We utilized a multivariate normal distribution, $\mathcal{N}\left(\mathbf{0}, \mathbf{\Sigma} = 0.16\mathbf{I}\right)$; our reasoning is that we believe the prior should somewhat match the data. On average, a stock has a daily return around 0 and very few days with large changes (stocks don't typically crash one day then flourish the next). Thus, we expect a small variance with the data being centered around 0. In addition, on average, we expect the change in the β's to have a minimal effect on the stock market moving up or down as a whole. Consequently, we do not think this prior distribution is an unreasonable choice, and as we will see, we achieve good results.

We now arrive at our posterior distribution which we hope to sample from,

$$\begin{aligned}
\pi(\boldsymbol{\beta}|\mathbf{Y},\mathbf{X}) &\propto \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i} \exp\left\{-\frac{1}{2}(\boldsymbol{\beta}^T\mathbf{\Sigma}^{-1}\boldsymbol{\beta})\right\} \\
&= \prod_{i=1}^{n} \left(\frac{1}{1+\exp\boldsymbol{\beta}\mathbf{X}_i}\right)^{y_i} \left(1-\frac{1}{1+\exp\boldsymbol{\beta}\mathbf{X}_i}\right)^{1-y_i} \exp\left\{-\frac{1}{2}(\boldsymbol{\beta}^T\mathbf{\Sigma}^{-1}\boldsymbol{\beta})\right\}.
\end{aligned}$$

3.2 NUTS

Previously stated, the PyMC3 package uses a version of a Markov Chain Monte Carlo sampler called the No U-Turn Sampler, or NUTS for short. More specifically, NUTS is a version of Hamiltonian Monte Carlo (HMC), also known as hybrid Monte Carlo, which is itself a version of the Metropolis-Hastings algorithm. The reason we used HMC over other traditional MCMC sampling algorithms is because HMC avoids the random walk behavior. HMC avoids the random walk by taking steps which are calculated as function of the gradient of the Hamiltonian for the target distribution (for us this is the posterior distribution); this is an operator corresponding to the total energy within a distribution, and has foundations in quantum mechanics. A brief overview of a single iteration for HMC is as follows:

- Start at value $\boldsymbol{\beta}_0$.
- Draw a random vector $\boldsymbol{\omega}_0 \sim p(\boldsymbol{\omega}) \propto e^{-\mathbf{V}(\boldsymbol{\omega})}$, where $\mathbf{V}(\boldsymbol{\omega})$ is referred to as the kinematic energy of $\boldsymbol{\omega}$. Often $p(\boldsymbol{\omega})$ is chosen to be a multivariate normal distribution.
- The Hamiltonian, $H(\boldsymbol{\beta}_0,\boldsymbol{\omega}_0) = -\log\pi(\boldsymbol{\beta}|\mathbf{Y},\mathbf{X}) + \mathbf{V}(\boldsymbol{\omega}a_0)$. We apply Hamiltonian dynamics which involves calculating the gradient of $H(\boldsymbol{\beta}_0,\boldsymbol{\omega}_0)$ and moving at a specific step size, for a specific number of steps, T, towards a new $H(\boldsymbol{\beta}_T,\boldsymbol{\omega}_T)$.
- We accept the new $\boldsymbol{\beta}_T$ with a probability of

$$\alpha = \max\left\{1, \frac{\exp(-H(\boldsymbol{\beta}_T,\boldsymbol{\omega}_T))}{\exp(-H(\boldsymbol{\beta}_0,\boldsymbol{\omega}_0))}\right\}$$

With HMC, the user specifies the size of the steps as well as the desired number of steps to take. In a nut shell (yes, pun intended), NUTS eliminates the need to set the number of steps to take. While we hope this provides a nice overview to both HMC and NUTS, we understand that the descriptions may be a little vague. We recommend reading Hoffman and Gelman (2014), who provide much better descriptions for both HMC and NUTS.

4 Results

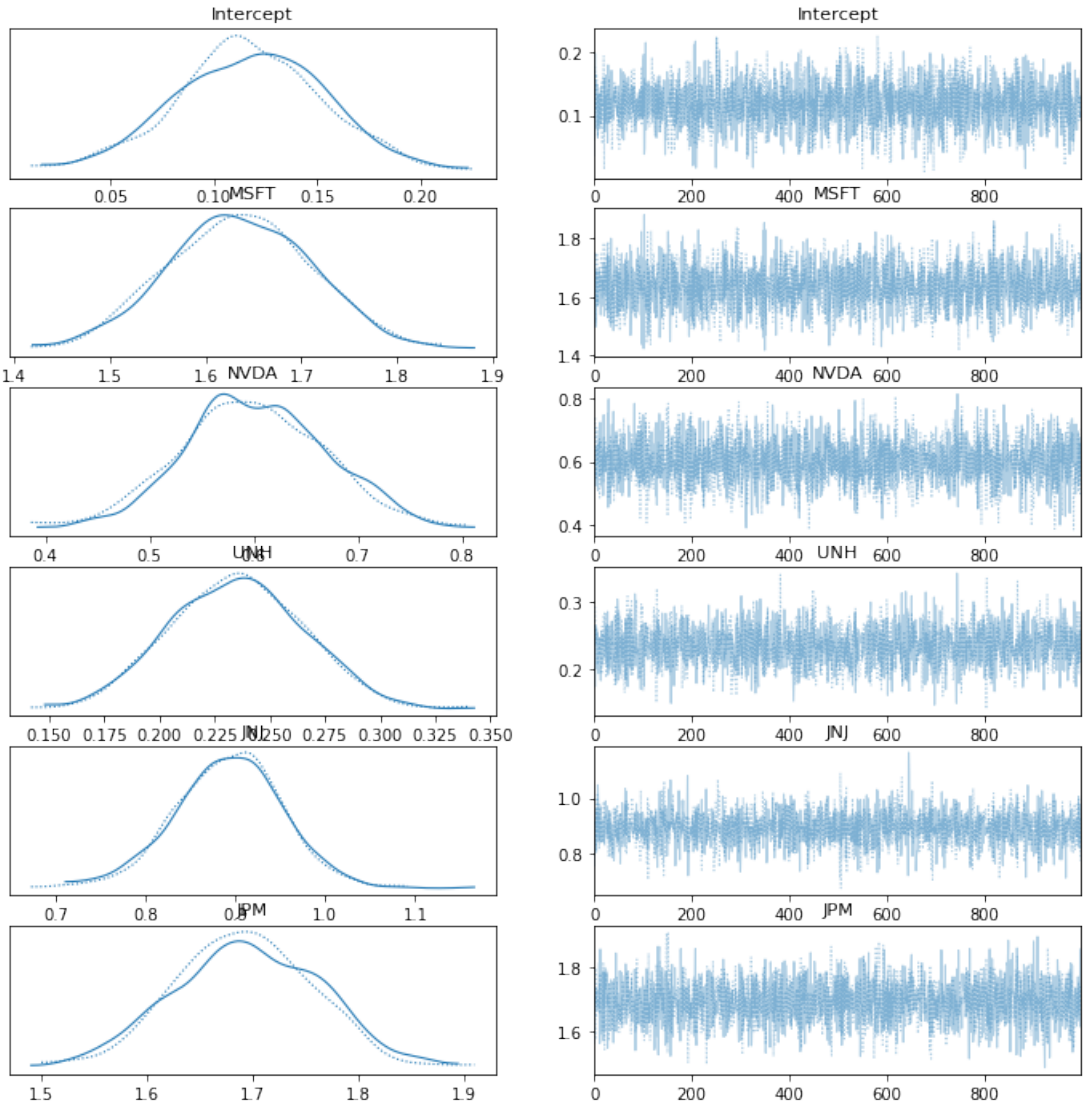

Figure 2: Respective Marginal Distributions for $\pi(\boldsymbol{\beta}|\mathbf{Y}, \mathbf{X})$ and the corresponding trace plots.

We ran the NUTS algorithm for 1,000 iterations to sample from $\pi(\boldsymbol{\beta}|\mathbf{Y}, \mathbf{X})$. Fig. 2 depicts the respective marginal distributions for each β_i from our posterior distribution as well as their corresponding trace plots. An important note about the PyMC3 package and the NUTS sampler is that it samples two chains and thus produces two posterior distributions on the same plot; hence, the solid blue line and the dashed blue line in each plot. For our purposes, we only consider one posterior (the solid blue lines on each plot) since they appear similar for both. Beyond this technicality, we can see that we seem to have very good convergence for all of our β_i's as each trace plot appear stable or stationary around a particular value.

Given our sample, we would like to evaluate its performance and compare it more traditional methods (i.e. frequentist methods). In the table below we present the Monte Carlo estimates of the marginal means for each marginal distribution of the β's ("Mean Bayes Coefficents"), as well as the parameter values from fitting a logistic regression model using frequentist methods (we used the Scikit-learn logistic regression model with 'liblinear' solver).

Stock	Mean Bayes Coefficients	Standard Regression Coefficients
Intercept, $\widehat{\beta^*}_0$	0.117544	0.118545
MSFT, $\widehat{\beta^*}_1$	1.635678	1.693530
NVDA, $\widehat{\beta^*}_2$	0.600274	0.615982
UNH, $\widehat{\beta^*}_3$	0.235202	0.238124
JNJ, $\widehat{\beta^*}_4$	0.892286	0.912624
JPM, $\widehat{\beta^*}_5$	1.693297	1.744529

We can see from the table that our Monte Carlo estimates of the marginal means (produced from the sampling method) align pretty closely with the coefficient values resulting from finding logistic regression coefficients in a traditional manner. This is a good indicator that our prior and likelihood were well suited for inferring proper logistic regression parameters.

We are not only concerned with how closely our Monte Carlo estimates of the marginal mean coefficient values match those of a frequentist estimate, but how the models preform from a predictive standpoint. Fig. 3 provides two confusion matrices for back tested predictions for models using the means Bayes coefficients (left) and standard logistic regression coefficients (right).

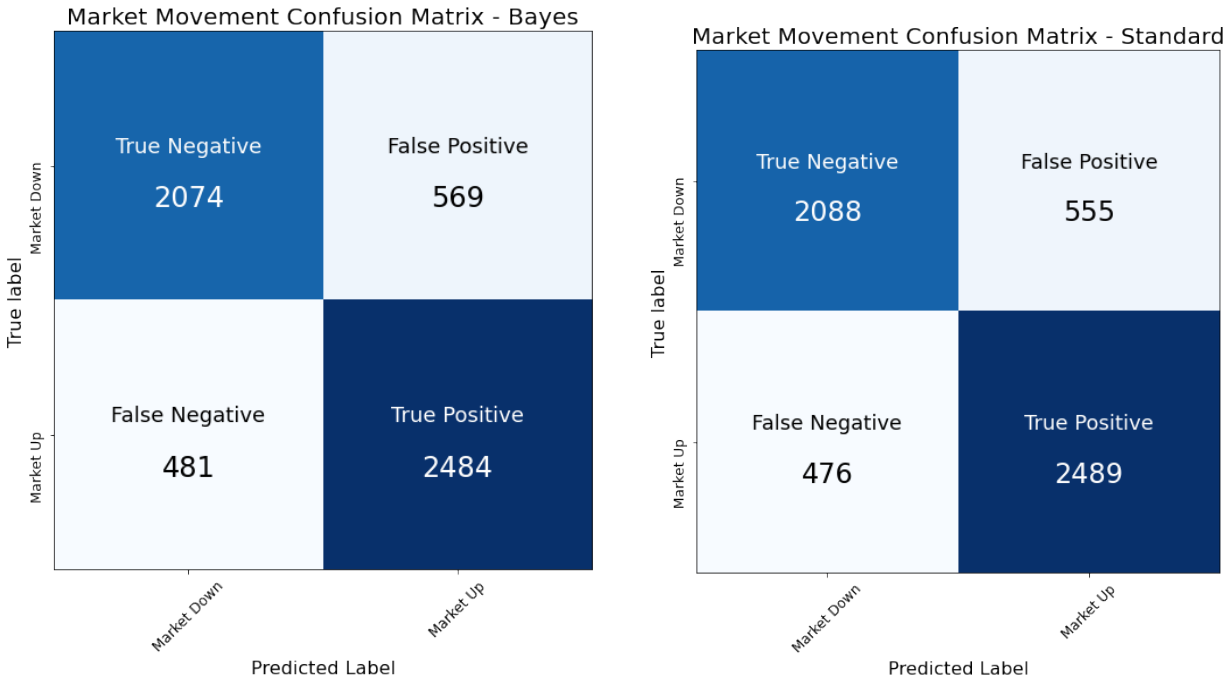

Figure 3: Left: Confusion matrix for back tested predictions using mean Bayes coefficients. Right: Confusion matrix for back tested predictions using standard frequentest estimated coefficients.

As we can see from the matrices, both the Bayesian logistic regression model and traditional logistic regression model performed almost identically. This is to be expected as the mean Bayes coefficients are very similar to the standard regression coefficients.

Moreover, we believe it is necessary to interpret the results of the confusion matrix, calculated using Bayesian inference. For instance, our Bayesian logistic model produced a true negative rate of 78.47%; this means that our Bayesian model correctly predicted the stock market as going down, when it truly went down, 78.47% of the time. Similarly, our Bayesian model correctly predicted the stock market as going up, when it empirically went up, 83.77% of the time. Conversely, the model incorrectly predicted the stock market

as moving up when it truly moved down 21.53% of the time. Finally, the model incorrectly predicted the stock market as moving down, when it truly moved up, 16.2% of the time. These are pretty good outcomes! While these results do provide more confidence that our sampling algorithm and choice of distributions were wise choices, the real power of Bayesian logistic is in the ability to predict the probability of SPY to go up.

4.1 Posterior Predictive Estimation

Mentioned before, we are mostly interested in predicting the probability of SPY having a positive daily return ($y_i = 1$) given new daily return values from each of our five stocks. Gratefully, this task is easily achieved. Recall that our likelihood is given by

$$f(\mathbf{Y}|\mathbf{X}, \boldsymbol{\beta}) = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i},$$

where p_i is the probability that $y_i = 1$ on day i and is given by

$$p_i = \frac{1}{1+e^{\eta_i}}.$$

Here $\eta_i = \beta_o + \beta_1 \text{MSFT} + \beta_2 \text{NVDA} + \beta_3 \text{UNH} + \beta_4 \text{JNJ} + \beta_5 \text{JPM}$. We are then able to generate a distribution around p_i, based on our posterior samples of $\boldsymbol{\beta}$. For each sample in our posterior sample set, we calculate η and a new value for p which we call p^*. Explicitly stated,

$$p^* = \frac{\exp\left\{\beta_0^{(t)} + \beta_1^{(t)}\text{MSFT}^* + \beta_2^{(t)}\text{NVDA}^* + \beta_3^{(t)}\text{UNH}^* + \beta_4^{(t)}\text{JNJ}^* + \beta^{(t)}\text{JPM}^*\right\}}{1 + \exp\left\{\beta_0^{(t)} + \beta_1^{(t)}\text{MSFT}^* + \beta_2^{(t)}\text{NVDA}^* + \beta_3^{(t)}\text{UNH}^* + \beta_4^{(t)}\text{JNJ}^* + \beta^{(t)}\text{JPM}^*\right\}},$$

where $\beta^{(t)i}$ is the t-th draw from the marginal posterior distribution for the i-th β and the STOCK* are the new values of the daily returns for each respective stock.

Once we have our distribution of p^*, we can sample $y^* \sim \pi(y^*|y) = \text{Bin}(1, p^*)$. Note that $\pi(y^*|y)$ is referred to as the posterior predictive distribution. While we are mainly concerned with the distribution of p^*, it is always good to know how this information relays to actual predictions.

For a first example we consider a sample date of April 22nd 2020, where MSFT = -7.65, NVDA = -7.88, UNH = -11.12, JNJ = -0.93, JPM = -3.75, and SPY = 0. Immediately these values jump out as unusually low, so we would naturally hope that we see a p^* distribution that reflects this.

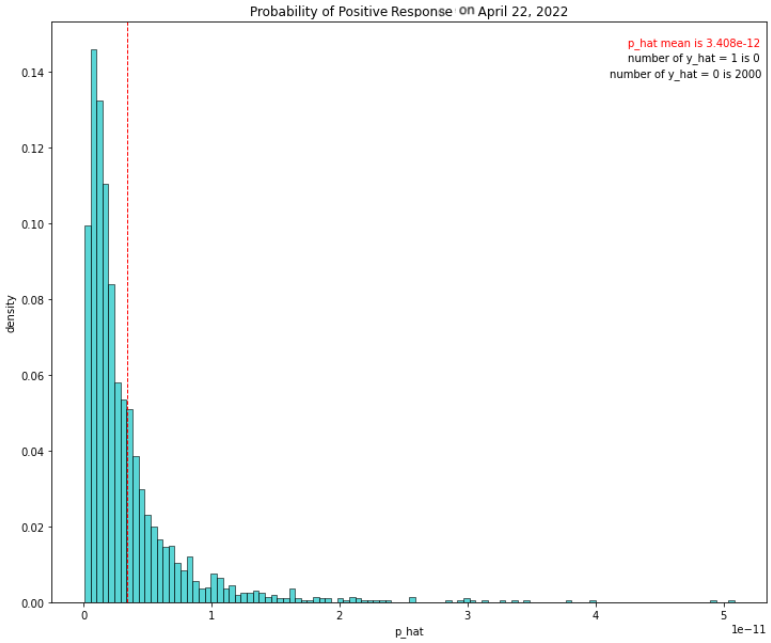

Figure 4: Distribution of p^* for April 22nd 2022

Thankfully as we can see in Fig. 4 that we achieve just that. The red line corresponds to the mean, and in the top right corner we list the the number of draws from $\pi(y^*|y)$ which are 1 (in this example 0) and the number of draws which are 0 (in this example 200).

Now we consider the distribution of p^* when the stocks take their average daily returns values, i.e: MSFT = 0.0106, NVDA = -0.0083, UNH = 0.0405, JNJ = 0.0168, and JPM = -0.0049. Fig 5 depicts the distribution of p^* under these values. As we expect, this appears to be relatively normal with a mean value of p^* being .536, indicating that under average daily returns, the probability of the market as a whole going up or down is approximately 50/50.

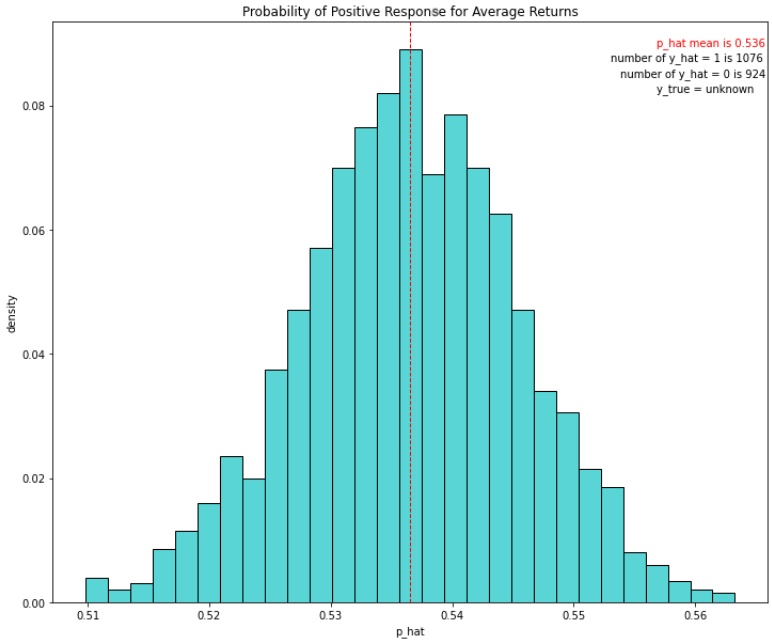

Figure 5: Distribution of p^* for April 22nd 2022

5 Discussion

The resulting marginal distributions indeed look promising. In fact, through the calculation of the confusion matrix for the Bayesian logistic regression model, we confirmed that the model did well at predicting the movement of the stock market on a particular day. Overall, we would feel comfortable implementing this Bayesian model in a work setting to predict and explain the general movement of the stock market.

Although our model produced good results, we believe there may have been some miss steps along the way. While we assumed that the prior was a multivariate distribution, other prior distributions may have been more appropriate. Perhaps if we considered the prior to be a product of normal distributions, instead of a joint normal, we could have seen different results. This is something to consider as an extension of the project! Another extension of the project would be to analyze how using more than 5 stocks' daily returns changes the marginal distributions of the β's and thus the confusion matrix and predictive distributions; will we get better or more sound results?

As we have seen, all of the estimated marginal parameter distributions appear to be approximately normal. So in theory we should be able to sample from our marginals and select our β_i without running a regression every time. But, even more practical is sampling from $\pi(y^*|y)$ and looking at the distribution of p^* which we could not otherwise do in a frequentist setting.

6 References

Chen, J. (2021, April 23) *Exchange Traded Fund – ETFs.*
Retrieved from:
`https://www.investopedia.com/terms/e/etf.asp`

Chen, Yen-Chi (2019)*Lecture 9: Hamiltonian Monte Carlo*
Retrieved from:
`http://faculty.washington.edu/yenchic/19A_stat535/Lec9_HMC.pdf`

Hoffman, M Gelman, A (April 2014) *The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo*
Journal of Machine Learning Research 15 (2014) 1351-1381
Retrieved from:
`http://www.stat.columbia.edu/~gelman/research/published/nuts.pdf`

Salvatier J., Wiecki T.V., Fonnesbeck C. (2016) *Probabilistic programming in Python using PyMC3.*
PeerJ Computer Science 2:e55 DOI: 10.7717/peerj-cs.55.

7 Appendix

Github

The link below is a URL to all the relevant code of this project:
`https://github.com/mra717/STAT5630Project`