# **A Study of Treasuries and Market Trends with Machine Learning**

Shane Rivera Sujay Jakka

Auburn University Auburn University

[srr0044@auburn.edu](mailto:srr0044@auburn.edu) svj0007@auburn.edu

**Abstract**

*Exploring the intricate dynamics of U.S. treasuries, GDP, unemployment rates and inflation, this research delves into their significant roles in shaping economic policies and market behavior. The study highlights the critical importance of understanding these factors at a simple level, given their profound impacts on almost all aspects of life revolving around money. We have begun uncovering trends and correlations within datasets, revealing insights unique to traditional analyses. Through this endeavor, the research seeks to contribute to a deeper understanding of economic dynamics and provide valuable insights that can be kept up to date. Our research demonstrates the effectiveness of employing polynomial basis functions in enhancing the predictive accuracy of Linear Regression models for forecasting Yield of Maturity. We have developed four distinct models as of this report, each tailored to specific feature combinations. Besides Linear Regression, we decided to implement models utilizing Recurrent Neural Networks (RNNs), Autoregressive Integrated Moving Averages (ARIMA), and Random Forest Regression.*

# **Introduction**

The investigation dives into how U.S. treasuries, inflation rates, GDP, and unemployment are connected, recognizing their big impact on the economy and financial markets. This problem highlights how these factors influence monetary policy, investment decisions, and overall economic well-being. Understanding this relationship is key for everyone from policymakers to investors, as treasury securities affect interest rates, inflation eats into purchasing power, unemployment rates affect economic volatility, and GDP reflects economic activity.

Our research aims to shed light on these connections through digging into loads of data and using machine learning algorithms / statistical methods. By analyzing daily changes in treasury securities and macroeconomic indicators, we hope to uncover hidden patterns and relationships. Ultimately, this can give us better insights into how markets behave, help policymakers make smarter choices, and guide us toward building a more resilient economy.

## **Data Sets**

For the purposes of this machine learning project, we will utilize Quandl package in python to load in an up-to-date US Treasury dataset, obviating the need for independent data collection. This dataset encompasses excess information. We will be cleaning the dataset to focus on four distinct categories of treasuries: 3-month T-bills, 5-year T-notes, 10-year T-notes, and 30-year T-bonds. Within each dataset, there are recorded data points including the date, high, low, open, and close values, spanning from 1990 to the present. In this project, we will only be utilizing the Close value. Additionally, we will import data from the Federal Reserve Economic Data (FRED) package in python for inflation, unemployment, and GDP data, which also dates from 1990 up to the present. Furthermore, our algorithmic framework is aimed to be designed with adaptability in mind, allowing the addition of new data points to ensure our analyses remain attuned to the dynamic nature of treasury markets.

## **Supportive Readings**

For background on treasuries, inflation, unemployment, and GDP, we will refer to analyses from reputable financial publications such as The Wall Street Journal. These sources offer insights into market trends, economic indicators, and policy developments relevant to our study. Additionally, we will explore various research papers dealing with analysis and machine learning algorithms. Drawing from a diverse range of academic sources allows us to explore alternative methodologies and theoretical frameworks that may enhance our understanding and analysis of the relationships between treasuries, inflation, unemployment, and GDP.

## **Proposed Algorithms to Implement**

Our objective is to use GDP, unemployment rate, and inflation rates to predict the yields of various Treasury Securities at different timestamps. To predict these values, we will be using the following Machine Learning models: Linear Regression, Recurrent Neural Networks (RNNs), Autoregressive Integrated Moving Averages(ARIMA), and Random Forest Regression.

## **Evaluation of Results**

Our evaluation approach integrates qualitative and quantitative analyses. Qualitatively, we'll employ visualizations to elucidate trends in U.S. treasuries, GDP, unemployment, and inflation over time. Regarding the models themselves, we will create residual plots to see randomness, which means our models are successfully capturing all the information within the data. Quantitatively, we'll assess prediction accuracy using metrics like mean squared error, particularly focusing on the algorithms' capability to predict treasury values and possibly determine optimal trading periods. We will also create scatter plots to show actual values versus predicted values, along with a regression line using Matplotlib and Seaborn.

## **Example Data Points**

Below are example tuples from the 10-Year Treasury Yield rates. When designing our machine learning algorithms, we will be primarily testing using the 10-Year T-note since it’s what is commonly referred to when professionals refer to the “bond market”. We have processed the dataset to track monthly closing prices instead of daily, given the nature of fixed income and many federal announcements and measurements are reported monthly. Also, there are tuples of the concatenated data frame consisting of the measures (GDP, Unemployment, and Inflation) we imported from Federal Reserve Economic Data (FRED).

A screenshot of a computer screen

Description automatically generated

A screen shot of a computer

Description automatically generated

# **Preparing Our Data**

To ensure all our data is as accurate and up to date as possible, we implemented the fredapi and quandl packages. FREDapi allowed us to access GDP, Unemployment, and Inflation measurements directly from the government’s database, ensuring that our statistical measures are current and reliable. These datasets provide valuable insights dating all the way back to 1946, offering a comprehensive historical perspective. In addition, we utilized Quandl to obtain the four U.S. Treasuries for analysis, ensuring our financial data is also up to date. However, since the Treasury yield information only dates back to 1990, we took the necessary steps to ensure consistency by modifying the GDP, Unemployment, and Inflation statistics dataset to begin in 1990 as well, maintaining a cohesive timeframe for our analysis. By starting all our datasets from the same year, we make sure they match up well. This helps us keep our analysis consistent and makes it easier to understand how economic trends affect Treasury yields.

## **Viewing Data Before**

Before implementing any machine learning algorithms, we first want to see the basic trends in our data over time. Below are outputs of some plots to see general trends in the data over time. To do this, we implemented matplotlib.pyplot to create visual representations of our data.

A graph of growth and inflation

Description automatically generated with medium confidenceGDP, Unemployment Rates, & Inflation:

A graph showing a line of growth

Description automatically generated with medium confidence10-Year T-notes:

## **Prior Correlation**

A red and blue squares with numbers

Description automatically generatedA correlation matrix is a valuable tool used in data analysis to understand the relationships between multiple variables. The quantitative interpretations of yield, GDP, unemployment rates, and inflation, from the correlation matrix provide insights into how these variables are related to each other. Each cell in the matrix represents the correlation coefficient, which quantifies the strength and direction of the linear relationship between two variables. The correlation coefficient ranges from -1 to 1: a value closer to 1 indicates a strong positive correlation, while a value closer to -1 indicates a strong negative correlation. A value near 0 suggests little to no linear relationship between the variables.

For instance, examining the correlation matrix revealed changes in GDP are very positively correlated with changes in yield, indicating that a growing economy leads to higher yields on investments. As you can see in the image of the correlation matrix above, we discovered that Inflation and GDP are highly correlated. These features have a Pearson’s correlation coefficient of 0.99. The correlation coefficient of Inflation and Unemployment and GDP and Unemployment are -0.2 and -0.14 respectively.

By calculating the correlation matrix before implementing our algorithms, we determined that none of the models we create will have these GDP and Inflation together as we believed it would be redundant due to their high correlation. This insight allows us to streamline our analysis and focus on the most relevant variables, enhancing the efficiency and effectiveness of our predictive models. Ultimately, leveraging the correlation matrix enables us to make informed decisions regarding feature selection, ensuring that our models are optimized for accuracy and relevance to the underlying economic dynamics.

## **Starting Models**

Using this information, we decided to develop three distinct models: one leveraging the features of Inflation and Unemployment, another utilizing GDP and Unemployment, and a third focusing solely on Inflation. Our decision to include a model based solely on inflation stemmed from its high correlation with the Yield of Maturity, registering at -0.84. Employing Linear Regression as our initial algorithm to forecast the Yield, we commenced without basis functions for GDP and Unemployment, Inflation and Unemployment, and Inflation alone, yielding R-Squared accuracy values of 0.669, 0.690, and 0.624, respectively. Leveraging Sci-kit learn facilitated our optimization process by directly solving for optimal weight values without the need for specifying learning rates or epochs, bypassing the gradient-based approach.

Dissatisfied with these initial accuracy metrics, we initiated data preprocessing to enhance predictive performance. Incorporating basis functions, we augmented feature sets in anticipation of improved accuracy. Through experimentation with polynomial degrees across our three models, we discovered optimal results with a 4th-degree polynomial basis function for the GDP and Unemployment model, a 6th-degree polynomial basis function for the Inflation and Unemployment model, and a 7th-degree polynomial basis function for the Inflation-only model. Consequently, our R-Squared accuracy metrics significantly improved, achieving values of 0.913, 0.931, and 0.892, respectively. This marked enhancement underscores the potency of incorporating polynomial basis functions compared to our initial Linear Regression models.

A screenshot of a computer

Description automatically generated Here you can see our initial best results:

# **Our Final Models**

For our final implementation we decided to go with the following 4 models: Linear Regression, LSTM (Long Short-Term Memory) Model, Random Forest, and ARIMA (Autoregressive Integrated Moving Average). In our previous implementation with linear regression, we realized we did not make use of the sequential aspect of our data. We initially tried to create a model that when given a combination of Inflation, GDP, and Unemployment Rate to predict the corresponding Yield of that month. However, we now reshaped our data into the following dimensions:

Number of Years, Number of Months (11), and Number of Features. Each example of our data corresponds to a year, and we are trying to predict the Yield of Maturity of 10-year bonds of the first day of December in that year. Furthermore, each example in our data consists of the Yield of Maturity of 10-year bonds, GDP, Unemployment Rate, and Inflation Rate of the first day of each month of the year. These are our features.

## **Approach**

In advance to running our models, we created different subsets of all our features. Some of these subsets involved just Yield, GDP, and Unemployment; Yield, Unemployment, and Inflation; GDP, Unemployment and Inflation; Yield; GDP; Unemployment; and Inflation.

Our reasoning for this was to see what specific values give the most accurate predictions. This approach was to first narrow down these top features, if possible, to get the best performing models.

We decided to first build our Linear Regression models and then build our more complicated models such as the LSTM model, ARIMA model, and Random Forest model. For our linear regression model our data which is in the form of Number of Years, Number of Months (11), and Number of Features, we decided to flatten our data and make it two dimensional instead of three dimensional. So, our data was now in the form of Number of Years x Data for 11 months. We ran this data using all our features and a subset of our features. We also applied polynomial basis functions to find our best model. With the LSTM models we also tried to use a combination of different subsets of our features. However, for our LSTM model it requires the original three-dimensional form of our data. For both our LSTM model and our Linear Regression model we allocated 80% of our data for training and 20% for testing.

For the ARIMA model we used only our Yield of Maturity for the first day of each month to predict the Yield of Maturity for the first day of the 12th month. To find the optimal hyperparameters for the ARIMA model it required creating Autocorrelation and Partial Autocorrelation plots to find q and p respectively. The q value in an ARIMA model is the number of previous residuals(lag) used to predict our future value and the p value is the number of past observations(lag) that are considered for predicting our future value. When we create these plots, we look for the correlation with this sudden drop off. The lag corresponding to the correlation before this sudden drop off is our desired value. We do this for the partial autocorrelation plot to find our p value and our autocorrelation plot to find our q value.

A graph of a function

Description automatically generated

A graph with blue lines and numbers

Description automatically generated

For the PACF plot there was a sudden drop off after the PACF corresponding to 1 lag. So, we chose 1 as our p value. For our ACF plot there was not a sudden drop off anywhere in the plot, so we arbitrarily chose 0 and 1 as our q values. Lastly, the ARIMA model requires a d parameter which describes the differences needed in our data so our data can become stationary (constant mean and constant variance across time). When we took the first derivative of our Yield data we got the following plot.

A graph showing a graph of yield

Description automatically generated with medium confidence

As you can see from the plot above the mean and variance looks constant throughout the plot. This is because there is not upward or downward trend in the graph. We also found the p-value of the 1st derivative of our Yield Data by doing a Augmented Dickey-Fuller (ADF) Test. We got a value of approximately 0 which is lower than our significance level of 0.05 allowing us to reject the null hypothesis that our data is not stationary. So, the d value we chose is 1.

Lastly for Random Forest we used the default hyperparameters to model our data. We flattened our data similar to what we did for linear regression, and we created model for Yield, GDP, Inflation, and Unemployment and also a subset of these features of size 3 such as Yield, GDP, and Inflation; Yield, Unemployment, and Inflation; and GDP, Unemployment, and Inflation.

## **Code Reuse**

Given the nature of computer science and this semester being the first time we have utilized machine learning algorithms, we looked at many other algorithms and ways of transforming data to understand both industry standards and help inspire new ideas.

In our research, we leveraged Python packages like fredapi and quandl for importing economic and financial data. These packages were necessary resources, which saved us the effort of writing custom data retrieval scripts and ensuring a streamlined process for accessing reliable datasets that stay up to date. Originally, we found downloadable dataset, but while researching how to correctly transform those datasets to our specifications, we learned we could directly import the same information using fredapi and quandl. The main aspect of our code reuse was looking on how exactly to import the data we wanted. We had to sign up for accounts on both fredapi and quandl to get API keys. Both websites had links that show the basics of how to access their databases through python and what parameters access what data.

However, our project's requirements extend beyond mere data importation; we must also align the datasets with our specific analytical framework. To achieve this, we've employed resampling techniques to adjust the temporal frequencies of the retrieved data. Resampling allows us to transform the raw data into more suitable time intervals, such as converting daily data points into monthly values. This process ensures that our analyses are conducted on data that aligns with the timing and granularity of our project objectives. To ensure correct incorporation, we looked at other projects trying to achieve a similar goal but with different datasets and incorporated what we needed to.

Again, to see what was possible, we researched examples of ways to show visualizations of our results quantitatively and qualitatively. Since our project focused on fixed income and predictions, we decided that scatterplots with a regression line could best compare each model to one another.

## **Final Code**

In our final presentation video, we review everything coded for this final project. Here is a quick overview:

-Imported all necessary packages and installed quandl and fredapi.

-Loaded in datasets and resampled to monthly values

-Created matplotlib graphs to show initial data trends, along with a correlation matrix

-Transformed data into NumPy X and Y sets for training/testing

-Created subsets of different combinations of our features

-Implemented Linear Regression model with scatterplot of the model with the lowest MSE

- Implemented RNN model with scatterplot of the model with the lowest MSE

- Implemented ARIMA model with scatterplot of the model with the lowest MSE, along with a residual plot since it was our most successful model.

- Implemented Random Forest Regression model with scatterplot of the model with the lowest MSE

# **Experiments**

As previously stated, our experiments on the datasets are strictly to predict the 12th month starting yield value, given the previous 11 months’ Yield, GDP, Unemployment Rate, and Inflation Rate data.

## **Quantitative Results**

For Quantitative Results, our main measurement was mean squared error. Here are our best testing results for each model:

Linear Regression MSE: 0.1191

Recurrent Neural Networks MSE: 0.5972

ARIMA MSE: 0.1118

Random Forest Regression MSE: 0.3181

## **Qualitative Results**

For Qualitative Results, our main analysis for each model was a scatter plot representing the actual yields versus the predictions our models gave. Also, we used Seaborn to add a regression line to represent the relationship between the variables.

Here are the plots for each model:

Linear Regression:

A graph with blue dots and a line

Description automatically generated

Recurrent Neural Networks:

A graph with a line

Description automatically generated

ARIMA:

A graph of a scatter plot

Description automatically generated

Random Forest Regression:

A graph of a scatter plot

Description automatically generated

Also, since our ARIMA model plotted well, we created a residual plot to help visualize the model’s complexity and ability to understand all the data. In the following plot, the residuals appear randomly distributed around the horizontal line at y=0. Since there are no discernable patterns, we believe the model has successfully captured all the information in the data.

A diagram of a plot

Description automatically generated

# **Conclusions**

This final project taught us both a lot. We chose our topic and objectives because it’s not something that is done publicly, so we wanted to see what kind of results we could get. Usually, in the securities market, only large companies with extreme computational power can even somewhat use machine learning algorithms to gain an advantage in trading.

However, since we are both interested in the financial industry and how our computer science degrees could be utilized to provide as much value as possible, we decided to do the best we could. In all honesty, we did not get the results we anticipated. We knew that was a possible outcome, given the timeframe and limitations of statistics we implemented.

Nonetheless, we have got to explore predictions models and dos and don’ts relating to securities. With more time, we would love to have been able to implement even more statistical measures that might have some correlation with treasury rates. Also, there’s a possibility of using even 3-month and 30-year yield rates and their predictions to help predict the 10-year T-note yield.

In conclusion, this project was a great way to get exposed hands-on and learn some standard industry practices. One main thing we learned was all the possibilities and complexity of writing price prediction machine learning algorithms. That must be why the professionals get paid so much.

## **Division of Labor**

Shane Rivera:

I wrote the code for our imports and implementations of the datasets. This involved creating fredapi and quandl accounts, then combining and transforming the data to only the features we decided to handle in this project. With this, I created some initial visuals to help one understand the data we are analyzing and an initial correlation matrix. Then, I helped transform the data into X and y sets as NumPy arrays to be fed into the machine learning algorithms.

For the algorithm parts I helped determine what we should be trying to accomplish and the best ways to handle each feature. I wrote the Linear Regression and Random Forest Regression algorithms, and helped where I could while Sujay was doing the other two models.

After successfully implementing the algorithms, we both evenly split up the work on producing quantitative and qualitative results.

Because of other exams and personal conflicts, I did most of the Proposal and Mid-Project reports. For this Final Report we split up the work evenly since we each had our two algorithms, their results, and approaches to talk about.

Sujay Jakka:

After Shane imported the datasets and resampled them, I helped with transforming the datasets to NumPy arrays for machine learning algorithm compatibility. I created all the combinations of datasets with 1-4 of the features as well.

For the algorithm parts I began writing the code for the other ones to go off. I wrote the Long Short-Term Memory Recurrent Neural Network and Autoregressive Integrated Moving Average models. Also, I helped Shane with his models when necessary. A lot of the work we did was a collaboration, such as the qualitative results graphs. After Shane made the scatterplots, I implemented the regression line for an even better visualization of our models. Finally, given all that, I wrote about it all in this report and talked about it during our final presentation video.

## **References**

1. Federal Reserve Bank of St. Louis and US. Office of Management and Budget, Federal Debt: Total Public Debt as Percent of Gross Domestic Product [GFDEGDQ188S], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred-redesign.stlouisfed.org/series/GFDEGDQ188S, June 13, 2016.

https://pypi.org/project/fredapi/

1. Quandl. "Quandl Documentation." Quandl, n.d., <https://docs.data.nasdaq.com/docs/python-tables>
2. “Time Series Analysis and Forecasting with ARIMA in Python”, <https://medium.com/datainc/time-series-analysis-and-forecasting-with-arima-in-python-aa22694b3aaa>
3. “sklearn.ensemble.RandomForestRegressor”, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html