Evaluating Treasury Yield Projections via Machine Learning

Project Introduction and Personal Context

The modeling, forecasting, and prediction of the US treasury yields have been significant for economic and financial research. The US treasury market has been a critical driving force of economic growth, with significant influence over monetary policy, and is consistently utilized to predict economic recessions. While traditional statistical techniques, such as linear regression models, have provided reliable predictions for yield behavior, the emergence of neural network machine learning models, such as that of long short-term memory (LSTM) provides an opportunity to explore whether deeper temporal dependencies can provide even further accurate yield predictions. This is especially important given the autoregressive nature of predicting future treasury yields. This project aims to compare these two statistical methods by recording and comparing the individual performances of each model and how well they can predict the 10-year treasury yield.

Beyond the implications of evaluating the statistical influence and economic effectiveness of predicting treasury yields, this project also serves as an excellent opportunity to explore simple machine-learning applications within broader financial contexts. With an undergraduate degree in Finance and Economics and two years of work experience in Finance, my experience utilizing machine learning and exposure to coding has mainly been self-taught with minimal practical applications. This project is a steppingstone to gaining practical experience in coding-based research work, primarily focusing on learning data science techniques and leveraging statistical methodologies to develop financial prediction models.

TLDR - Project Question Recap: *Which machine learning model (linear regression or LSTM) is more effective in predicting 10-year U.S. treasury yields using financial and macroeconomic data?*

Sources, Reasoning, and Specifying the Number of Data Samples

To build both machine learning models, this project utilizes web scraping and calling on APIs for data retrieval from multiple publicly available data sources. The focus of this paper is to accurately predict treasury yields by utilizing historical yield data and macroeconomic factors that influence yields to feed into both models and determine predictions.

Treasury Yield Explanation

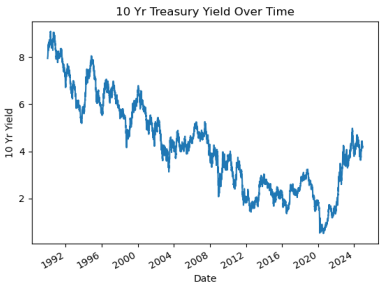
Prior to explaining the data retrieval process, it will be appropriate to discuss how treasury yields essentially function. The U.S. Treasury is a government entity responsible for managing the federal government's finances. The U.S. treasury raises debt by issuing government bonds to fund government operations. Each institutional government functions in this manner to raise money. Still, the securities issued by the U.S. treasury are often regarded as the "safest" because the full faith and credit of the U.S. government backs them. There are three main types of treasury securities: 1) Treasury Bills (T-Bills) are short-term securities that mature within 1 year or less, 2) Treasury notes are medium-term securities that mature within 2, 3, 5, 7, or 10 years, while 3) Treasury bonds are long-term securities maturing in 20 or 30 years. The 10-year Treasury Note is widely used as a benchmark in the financial world as it balances short- and longer-term securities. It reflects economic expectations that impact both, and it is a market standard utilized by many institutional investors due to its versatility in impacting a wide range of interest rates across the economy. The yield is *an investor's return* from investing in treasury security, influenced by market demand, federal reserve interest rate hikes, economic conditions, and general sentiment.

Data Retrieval and Cleaning Methodology

1. **U.S. Treasury Website:** The primary target variable was the 10-year treasury yield, which was scraped from the U.S. Treasury website (Daily Treasury Par Value Data - Link) using the Beautiful Soup library.
    a. *Treasury Yield Scrape Method:* The US treasury website organizes yield data into a singular HTML datatable for each year with headers representing the typical maturity period (e.g. "1 Mo", "10 Yr", etc.). Using a *'for loop'* the function parses through a URL for each year from 1990 to 2024, retrieving the page using *'requests'* library, identifying the <table> element on the page and extracts the column headers from the first row using <th> tags. Next, using another for loop, the function iterates through the remaining rows, collecting data from each <td> element and cleaning the text to remove whitespace. The data is iteratively stored in a DataFrame (df), and the function finally returns the df for use. If no table is found for a given year, the function returns an empty DataFrame and prints a warning. A cumulative DataFrame is returned using pd.concat(),

b. *Why Daily Treasury Par Value Data?* There are five different treasury statistics that one can retrieve from the treasury website. This project utilizes Daily Treasury PAR Yield data largely due to the desire of predicting the non-inflation-adjusted (nominal) value: allowing for clearer interpretation of yield prediction, avoiding overcomplications, and reducing risk of multicollinearity.

c. *Samples:* The initial WebScrape DataFrame resulted in 8,733 unique rows and 23 columns. The rows represented each working day (excl. weekends and public holidays) from Jan 1st 1990 to Nov 25th 2024. The WebScraped result provided columns that were not initially visible in the data table from the U.S. Treasury website (e.g. "8 WEEKS BANK DISCOUNT", "Extrapolation Factor", etc.) columns that had only null values. The cleaned DataFrame (after selecting the column headers that only included treasury yield values ["1 Mo"..."30 Yr"]) had 10 columns and 8,733 rows.



2. **FRED Data:** The macroeconomic indicators such as consumer price index (a.k.a. CPI, a measure of inflation), Federal Funds Rate (the overnight lending rate that the US Treasury controls), and GDP growth rate (the rate at which the economy grows) were retrieved from the Federal Reserve's FRED API using the "fredapi" Python library. This data source was chosen primarily due to its comprehensive and reliable repository of economic data, as well as its ease of integration with Python.
   a. *Retrieval Method:* Initialized the FRED API call method through applying for a valid API key. Using the get_series method, three key economic indicators were fetched "*CPIAUCSL*", "*DFF*", and "A191RL1Q225SBEA" for retrieval of CPI, Fed Funds Rate and GDP Growth Rate data respectively
   b. *Influence of these variables on Treasury Yields*: CPI captures inflation, which is crucial for investors to assess return on investment. Federal Funds Rate directly affects short-term interest rates and influences bond yields on all maturities. Finally GDP growth rates overall economic performance, influencing investor's expectations of future growth potential and strength of the US economy.
   c. *Samples:* CPI and GDP Growth Rate data are typically reported on a monthly or quarterly basis, while the Federal Funds Rate is available daily. Since, the focus of this project is to predict daily 10-Year Treasury yields via linear regression model, the missing daily values for CPI and GDP Growth Rate were filled using forward-fill ("*ffill*"). This method propagates the last known value forward, ensuring that each day has a corresponding macroeconomic data assigned to it. The three economic indicators were concatenated into a single pandas DataFrame using pd.concat(), resulting in a consolidated DataFrame with 12,749 daily observations from Jan 1st 1990 to Nov 26th 2024.
3. **Yahoo Finance:** To incorporate broader market context into the analysis, financial data such as the daily S&P 500 index, crude oil prices and other commodity prices were retrieved via the Yahoo Finance API through the yfinance Python library.
   a. *Retrieval Method:* began by identifying the relevant financial instruments and corresponding symbols for these commodities via scraping the Wikipedia page listing traded commodities using pandas.read_html(). Symbols were standardized by replacing periods with hyphens to match Yahoo Finance's ticker format (e.g., changing C/ZC to ZC=F). S&P 500 was taken via the YF ticker symbol of ^GSPC. The yfinance Python library, is not affiliated with Yahoo Inc, yet is an open-source tool that uses Yahoo's publicly available APIs and does not require a personalized API key for usage.
   b. *Data Selection, Processing and Feature Engineering:* S&P 500 represents the performance of the 500 largest U.S. companies within the public equity market. It represents largely investor sentiment and corporate economic health. Corn Futures, Crude Oil Futures, and Gold Futures represent agricultural, energy prices, and precious metals commodity market sentiment to understand broader future economic expectations. The downloaded data included columns such as Adj Close, Close, High, Low, etc. For analysis purposes, the focus was on Adj Close prices, which account for corporate actions like dividends etc. For ease of interpretability, the ticker symbols were mapped to their respective names (e.g. ZC=F was assigned "Zinc Futures"). To assess market performance the percentage change in adjusted close prices (Adj Return) was calculated using ".pct_change()"
   c. *Samples:* Since yfinance collates the tickers in a stacked manner, the resulting dataset comprises 27,076 rows and 5 columns, representing daily observations from 1989 to Nov 25, 2024 (excl. Weekends and holidays). Furthermore, the commodity futures markets were only collected beginning the year 2000, resulting in missing values for these instruments prior to that year.
4. **U.S. Bureau of Economic Analysis (BEA):** To incorporate slightly different economic indicators that can imply US economic strength and investor sentiment, data was retrieved from the Bureau of Economic Analysis (BEA) using the pybea Python library. The BEA API was chosen for its comprehensive coverage of national economic accounts.
   a. *Retrieval Method:* The data collection process targeted the National Income and Product Accounts (NIPA) dataset, specifically table "T10106" that provides critical economic metrics. To align with the Treasury Yield dataset timeframe, a "*for loop*" was used to iterate the retrieval process between 1990 to 2024. Retrieval method construction utilized API

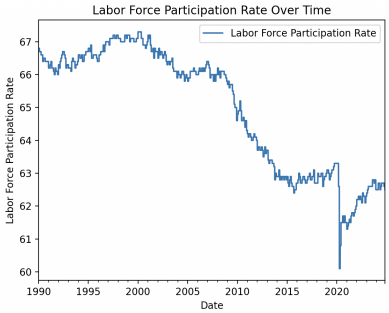
requests with parameters such as datasetname, “TableName”, “Frequency (quarterly)”, and “Year”. The “*requests.get()*” function sent these queries to the BEA API endpoint, and the JSON responses were parsed into pandas DataFrames.

b. *Rationale for Variable Selection:* Residential spending provides insights to private savings and housing market dynamics are closely tied to consumer confidence. Changes in Private Inventories provide insights into business operations and working capital cycle health (short-term cash reserves). Personal Consumption Expenditures (PCE) directly reflects consumer demand and economic health. Finally, Intellectual Property Products, signals investments into innovation and can help determine the sentiment regarding long-term economic growth.

c. *Samples:* The final BEA dataset comprises 2,479 rows and 6 columns, representing daily observations from Jan 1st 1990, to Mar 1st 2024 (excluding weekends and public holidays). The quarterly data was then converted to a daily frequency using forward-fill (ffill) to maintain consistency with the daily Treasury yield data. A pivot table was created to restructure the data, setting "TimePeriod" as the index and spreading the selected "LineDescription" entries into separate columns.

5. **Bureau of Labor Statistics (BLS):** The final data retrieval was data on the Labor Force Participation Rate (LFPR) was retrieved from the Bureau of Labor Statistics (BLS) using the BLS API.

a. *Retrieval Method:* The LFPR retrieval was performed via the series ID "LNS11300000". The retrieval spanned from 1990 to 2025 via a “*for loop*” to align with the Treasury yield data timeframe. For each year range, a POST request was sent to the BLS API endpoint with the necessary parameters, including the series ID, start year, end year, and API registration key.



b. *Rationale for Variable Selection:* The LFPR provides a broader view of the labor market by measuring the percentage of the working-age population that is either employed or actively seeking employment. Unlike the Unemployment Rate, which only accounts for those actively looking for work.

c. *Samples:* The final BLS dataset consists of 12,693 rows and 3 columns, representing daily observations from January 1, 1990, to October 1, 2024 (excluding weekends and public holidays)

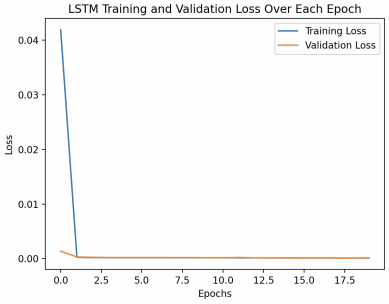
Data Cleaning, Analysis & Visualization

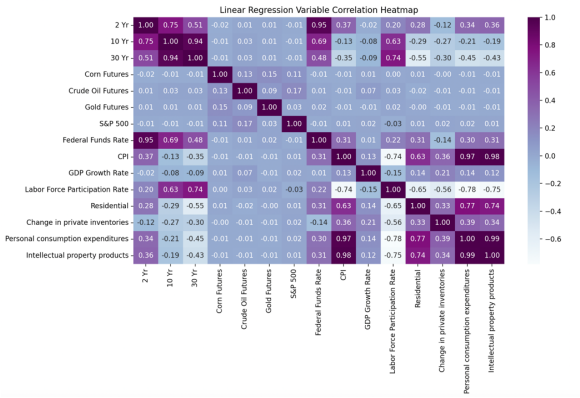
Due to the nature of this study, the datasets mentioned above required a few more data cleaning steps to build processed data for both LSTM and Linear Regression models. Prior to beginning the data cleaning process, the **Initial Hypothesis** for the paper was that the LSTM model would outperform the linear regression model, due to the model’s ability to capture temporal dependencies and autoregressive relationships which are vital to predict *daily* treasury yield values.

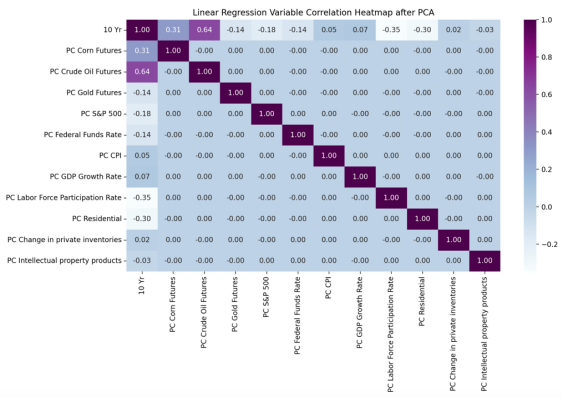
LSTM Model Analysis: The data preparation for the LSTM model required isolating a single variable (the 10-Yr Treasury Yield historical data) from the raw treasury CSV data table. *Initial premise*: the LSTM model was designed to capture and predict the ‘*percentage changes*’ in the 10-year Treasury yield. The initial reasoning for this was due to the intuition that percentage changes could capture short-term movements and eliminate potential noise caused by absolute levels. To pursue this, percentage change for the “10 Yr" "Adj Close" prices was calculated using the pandas *‘pct_change’* function. The issue when utilizing percentage change (despite seeming as a logical metric) the model produced unstable and less interpretable predictions. The percentage changes exaggerated small variations between absolute values on a daily basis. Thus, the LSTM wasn’t able to capture the underlying trend of treasury yields, leading to an issue of overfitting. The LSTM was then redesigned to only predict the absolute value of 10-Year yield. This adjustment leveraged the inherent strengths of LSTM by accounting for the autoregressive nature of the treasury yield behavior. To construct the LSTM model, it required transforming the initial data into time-series sequences. For this project, a sequence length of 10 days (selecting a period of 2 weeks) was chosen as it provided enough temporal context via the ‘*creating_sequences*’ function created, which took the dataset as a List[List[float]] input as well as sequence_length: int and outputted a tuple of Numpy Arrays of sequences and labels accordingly. The model mechanics consisted of one LSTM layer with 50 units, followed by a dense layer to predict a single output (10-Year Treasury Yield), compiled via the Adam optimizer (an optimizer generally chosen as the best overall choice for LSTM Models over RMSProp and AdaDelta). To ensure that overfitting was minimized, a version of k-fold cross-validation for LSTM models was adopted by using the ‘*sliding_window*’ function. Unlike traditional datasets where rows are independent, time series has temporal dependencies. The sliding_window method essentially trained on [1,2,3] of the dataset and tested on [4] only to continue the slide to the next variable for the second fold: Train on [2,3,4], Test on [5], utilizing this GitHub repository as

inspiration - Link. The model trained on scaled input data using '*MinMaxScaler*' to normalize the Treasury Yield values. The batch size and learning rate were monitored by visualizing the training & validation losses over each epoch.

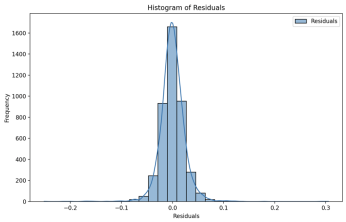
Linear Regression Model Analysis: The data preparation required all raw datatables to be merged into a unified DataFrame using the 'Date' column as the key. Utilized a left join to combine the data and ensure the values are consistent with the Treasury Yield Data dataframe. To ensure this, all raw datatables were standardized by converting the date column to "*datetime"* object for consistency. Missing values were also forward-filled to align with daily treasury data. The Yahoo Finance datatable required further data cleaning steps, where the tickers were initially stacked in the raw data table and required to pivot out all financial market data (S&P 500 and commodity prices) using the "*.pivot*" function. This pivot step allowed for a long-dataset to transition to a wide dataset ensuring consistency with the Treasury Yield DataFrame and guaranteed for no duplicate 'date field' values during the merge process. Certain macroeconomic indicators from each data source were not recorded/available during the early 1990s. Thus, when combining the dataset together, and ensuring that the linear regression model only made predictions on available data, the initial Treasury DataBase table with over 8,700 rows was truncated into a merged dataset with 4,285 rows via the "*.dropna()*" method, with the first datapoint introduced in beginning on Jan 3rd 2007 to March 1st 2024 providing over 15 years of daily yield data and macroeconomic indicators. As mentioned in the data retrieval process, the macroeconomic indicators that are captured on a monthly or quarterly basis (CPI index, GDP Growth Rate, etc.) were forward filled using the '*ffill*' method to ensure that there were no null values on a daily basis. The treasury yield was initially captured for a 30-year dataset to ensure capturing of economic cycles, however within this model, two major economic events (2008 financial crisis, COVID pandemic) were both recorded and captured. Further tests were performed to ensure robustness and decrease the inheritance of bias within the linear regression model. First was the influence of multicollinearity, where utilizing independent variables is a core assumption for any linear regression model. To ensure this, a correlation matrix heatmap was conducted as shown below on the figure to the left. Columns with strong multicollinearity with the dependent variable was removed (2 Yr Treasury Yield, and 10 Yr Treasury Yield). Furthermore Personal consumption expenditures was highly correlated with CPI, asserting they measured the same values. Another extremely high correlation was witnessed between intellectual property and CPI as well, however both variables were chosen to remain in the model as the definition presumably recognized two different categories of macroeconomic variables. CPI measured price of goods over time (exhibiting the perceived price of goods for end consumer) while intellectual property spend was utilized within this model to determine how much spend on innovation was being performed on an aggregate level to determine sentiment on long-term growth prospects. After removal of columns with strong multicollinearity, there still maintained a few variables that were highly correlated with each other. To adjust for this, a Principal Component Analysis (PCA) method was applied to reduce multicollinearity, by importing the PCA module from sklearn.decomposition library. The adjusted correlation heatmap is shown below on the right.

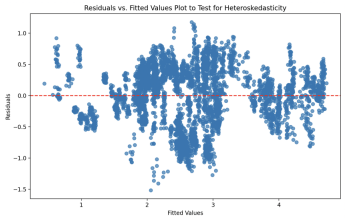






Next evaluation of residuals and histograms was performed to address heteroskedasticity. A histogram of residuals was plotted to check for normality, and a plot of residuals vs. fitted values confirmed that the model effectively addressed heteroskedasticity. Finally, the model addressed autocorrelation that was detected using the Durbin-Watson test, with a score of 0.023, suggesting positive autocorrelation was present. To adjust for this, the dependent variable was first log-transformed to linearize a non-linear relationship and interpret the coefficient with elasticity (a percent change in the dependent variable for a 1% change in independent variable. Secondly, a lagged variable of the dependent variable was added via the "*.shift(1)*" function to explicitly incorporate the influence of past values of a variable into





the model. The resulting OLS regression results provided insight that the model was robust and took care of the biases that were present earlier via the transformations performed. The OLS results improved the R^2 from 72.6% (an already well-fitted model) into 99.7% R^2 value. The lagged_log(10_yr) variable with a coefficient of 0.9963 highlights the extremely strong autoregressive nature of daily treasury yield values. The significant predictors of the model , outside of the lagged_log(10_yr) variable, was determined by Gold Futures, S&P 500 performance, Federal Funds Rate as well as CPI.

Comparing Results: Both models demonstrated exceptional performance with both model R^2 values near 1. The linear regression model slightly outperformed LSTM in terms of precision (lower MAE and MSE), but the LSTM had a better approach in capturing temporal dependencies, as specified in the initial hypothesis test.

Model	MAE	MSE	R2
Linear Regression	0.005	0.0	0.999
LSTM	0.094	0.013	0.997

Changes from Original Proposal

There wasn't much variation from my initial project proposal. The slight variations lied in the data variables that I selected. For example, from the FRED API, I decided not to incorporate the unemployment rate as I was gathering that data from the monthly labor force participation rate from BLS API. From Yahoo Finance, outside of Crude Oil prices, I incorporated corn and gold futures to improve commodity price expectations. Finally, a rolling error plot was not produced to visualize the model comparison, rather a simple table comparing model performance (MAE, MSE, and R^2), I believed was enough in visualizing the difference.

Mention of Future Work

If more time and resources were available, I would explore several avenues. 1) More rigorous bias elimination and addressing potential data leakage in the model pre-processing stage. I would establish a testing system on completely unseen data. 2) I would explore other well-established parametric models (such as the Nelson-Siegel Model) to benchmark against my ML-based methods to view comparative effectiveness. 3) I would investigate ways to combine the strengths of Linear Regression and LSTM and see if a hybrid model (such as feeding LSTM predictions into a linear regression model) could be explored.

Beyond the suggestions listed above, I need to address that I am a novice machine learning student, and much of my work or understanding of time series data and bias testing came from: the statistics classes I took as part of my economics undergraduate background, online resources, tutorials, forums and even friends within the industry to understand research methodologies. These come with the limitation of not fully understanding the underlying intricacies or potential pitfalls. Therefore, while the high R^2 values for both models seem successful, they raise a significant concern that is likely "too good to be true." I would try to study temporal splits, better forward-fill methods, and understand how PCA interpretation works in building a linear regression model. Overall, I enjoyed the opportunity to learn Python and test my understanding via this project, and look forward to future research opportunities such as this one.