Predicting Federal Funds Rate: Taylor's Rule Regression Model Validation and the Role of Target and Unemployment

DSC-590-O500: Capstone Project

Nicholas Pham

April 30, 2025

# Table of Contents

[Table of Contents 2](#_Toc196935776)

[Project Overview and Project Objectives 6](#_Toc196935777)

[State the Problem 6](#_Toc196935778)

[Background 6](#_Toc196935779)

[Project Objectives 8](#_Toc196935780)

[Challenges 9](#_Toc196935781)

[Benefits and Opportunities 9](#_Toc196935782)

[Project Scope 11](#_Toc196935783)

[Work Breakdown Structure 11](#_Toc196935784)

[Project Completion 13](#_Toc196935785)

[Project Completion Criteria 14](#_Toc196935786)

[Assumptions and Constraints 15](#_Toc196935787)

[Project Controls 17](#_Toc196935788)

[Risk Management 17](#_Toc196935789)

[Change Control Log 18](#_Toc196935790)

[Project Schedule 18](#_Toc196935791)

[Gantt Chart 20](#_Toc196935792)

[Cost Estimate 20](#_Toc196935793)

[Issues Log 21](#_Toc196935794)

[Requirements Analysis 22](#_Toc196935795)

[Use Cases 22](#_Toc196935796)

[System Design 28](#_Toc196935797)

[Technical Requirements 29](#_Toc196935798)

[Data Modeling and Analysis Process 30](#_Toc196935799)

[Reports 30](#_Toc196935800)

[Screen Definitions and Layouts 31](#_Toc196935801)

[Model Pipeline Design 32](#_Toc196935802)

[Design Planning Summary 32](#_Toc196935803)

[Overview of Design Concepts 33](#_Toc196935804)

[Deliverable Acceptance Log 36](#_Toc196935805)

[Overview of Model Pipeline Design 37](#_Toc196935806)

[Detailed Model Pipeline Design 38](#_Toc196935807)

[1. The data sources 38](#_Toc196935808)

[2. The dataset types and formatting 39](#_Toc196935809)

[3. The data cleaning procedure 39](#_Toc196935810)

[4. Method of initial data exploration and visualization 40](#_Toc196935811)

[5. The data model used and its nature (e.g., predictive) 40](#_Toc196935812)

[6. The methodology for interpreting the analysis results 42](#_Toc196935813)

[7. This section should also include any configuration changes that will be required to develop and implement the proposed solution. 43](#_Toc196935814)

[8. Describe the approach and resources required to assure system security, if applicable; otherwise, explain why security is not relevant. 44](#_Toc196935815)

[9. List the hardware and software technologies. 45](#_Toc196935816)

[Implementation 46](#_Toc196935817)

[Implementation Plan 46](#_Toc196935818)

[System Entities 48](#_Toc196935819)

[Functional Requirements 49](#_Toc196935820)

[Non-functional Requirements 53](#_Toc196935821)

[Source Code Listing 53](#_Toc196935822)

[Coding Improvements 54](#_Toc196935823)

[Results Analysis/Testing Components 55](#_Toc196935824)

[Module Test cases 55](#_Toc196935825)

[Test Case Name: Library Imports 55](#_Toc196935826)

[Test Case Name: Defining Functions 56](#_Toc196935827)

[Test Case Name: Data Wrangling 58](#_Toc196935828)

[Test Case Name: Exploratory Data Analysis (EDA) 59](#_Toc196935829)

[Test Case Name: OLS Modeling 61](#_Toc196935830)

[Test Case Name: OLS Regression Assumptions 62](#_Toc196935831)

[Test Case Name: Taylor and Karakas OLS Model Comparison 63](#_Toc196935832)

[Test Case Name: OLS Model Metrics 64](#_Toc196935833)

[Test Case Name: Neural Network Modeling 65](#_Toc196935834)

[Requirements Testing 68](#_Toc196935835)

[Component: Data Wrangling 68](#_Toc196935836)

[Component: Exploratory Data Analysis 69](#_Toc196935837)

[Component: OLS Regression Assumptions 70](#_Toc196935838)

[Component: Karakas and Taylor OLS Model Comparison (validation) 71](#_Toc196935839)

[Component: OLS Metrics 72](#_Toc196935840)

[Component: Neural Network Modeling 73](#_Toc196935841)

[Component: Neural Network Metrics 73](#_Toc196935842)

[Component: Neural Network Plots 74](#_Toc196935843)

[Component: Non-Functional Requirements 75](#_Toc196935844)

[Systems Testing 77](#_Toc196935845)

[User Guide 78](#_Toc196935846)

[Copyright 78](#_Toc196935847)

[Legal Notice 78](#_Toc196935848)

[Preface 79](#_Toc196935849)

[General Information 80](#_Toc196935850)

[System Summary 81](#_Toc196935851)

[System Architecture 81](#_Toc196935852)

[Using the System 84](#_Toc196935853)

[System Requirements 84](#_Toc196935854)

[Usage Instructions 84](#_Toc196935855)

[Key Features 85](#_Toc196935856)

[Accessibility Features 87](#_Toc196935857)

[Safety Information 88](#_Toc196935858)

[Troubleshooting 89](#_Toc196935859)

[FAQ (Frequently Asked Questions) 91](#_Toc196935860)

[Help and Contact Details 92](#_Toc196935861)

[Glossary 93](#_Toc196935862)

[Systems Administration Guide 95](#_Toc196935863)

[Copyright 95](#_Toc196935864)

[Legal Notice 95](#_Toc196935865)

[System Overview 95](#_Toc196935866)

[System Configuration 97](#_Toc196935867)

[System Maintenance 99](#_Toc196935868)

[Security Related Processes 100](#_Toc196935869)

[Table of Figures 101](#_Toc196935870)

[Capstone Project Presentation Link 103](#_Toc196935871)

[Appendix 104](#_Toc196935872)

[References 106](#_Toc196935873)

# Project Overview and Project Objectives

## State the Problem

This project aims to replicate and validate OLS regression models predicting the Federal Funds Rate using Taylor’s Rule, a policy guideline proposed by John Taylor from Stanford University in 1993 and equations based on this derived in Alper D. Karakas’ (2023) paper, “Reevaluating the Taylor Rule with Machine Learning.”

We will then construct other models by adding Federal Funds Target and Unemployment Rate to the Taylor Model to see if the addition of new features can improve the performance of regression models in predicting the Federal Funds Rate. We choose to build off the Taylor Model as this is the foundational model and Karakas (2023) found little difference in performance between this model and their model.

As the models use regression methods, we will check for the classical assumptions of regressions. We will additionally construct neural network models.

## Background

The federal funds rate is the interest rate at which banks lend to each other overnight. The reason banks do lend to each other is to ensure that they have enough money at hand. Before they removed reserve requirements in 2020, the Federal Reserve (2024) set legal requirements on the amount of money that banks had to keep in vaults at any time. Now, banks can use all their reserves and borrow from each other when needed. Banks do not keep all the money that people deposit in their vaults or accounts, but instead, use them for investment projects such as mortgage loans. The legal requirements on reserves are in place in part to prevent problems banks from spending all the reserves and avoid the situation where people come to withdraw money and find the bank does not have money to give them. Once that happens, people will panic. This can lead to a bank run, a situation where everyone runs to the bank to frantically withdraw money.

The rate is split between the Federal Funds Target and the Federal Funds Rate. The former is a rate set directly by a branch of the Federal Reserve called the Federal Open Market Committee (FOMC). The latter is a rate determined by the market, much like stock prices. The FOMC sets a target, then performs some operations like trading bonds to adjust the Federal Funds Rate and bring it towards their target.

The variables used will be:

* **Federal Funds Rate**: The interest rate that banks use when lending to each other overnight. This is determined by the market.
* **Inflation**: The percentage rate at which prices increase.
* **Unemployment**: The ratio of people without jobs to the total labor force.
* **Target**: The Federal Funds Rate that the Federal Reserve wants to set as part of economic and monetary policy.
* **Inflation Gap**: The difference between Inflation and the Inflation Target (the Inflation Target is treated as a constant 2% in this project).
* **Output Gap**: The difference between Real and Potential Gross Domestic Product (GDP). Measures the difference between how many goods and services a country produces each year vs how much it can produce in theory when using all available resources.
* **Inflation Lag**: The 1st lagged values of Inflation.
* **Output Gap Lag %**: The 1st lagged percentage of Output Gap.

## Project Objectives

* Wrangle the data
* Perform Exploratory Data Analysis
* Construct all the OLS models
* Check the classical regression assumptions for each model
* Check that the plots and relevant performance metrics of the Taylor Model and the Karakas Model match the results shown in Karakas’ paper.
* Compute and analyze performance metrics for each model
* Construct the Neural Network Models, make plots, and compute performance metrics for each model
* Ensure results are reproducible

## Challenges

* Each dataset has a date range of available data, and some datasets have smaller ranges than others
* The Karakas Model uses percentage units and the “vector form” of the variables used in their equation (Karakas, 2023). What the vector form is and how to find it was not stated clearly in the paper.
* The data varies in frequencies such as daily, quarterly, and yearly.
* Federal Funds Target used to have a single value, but the Federal Reserve switched to using ranges

## Benefits and Opportunities

Those that benefit from this project are economists, researchers, and those aiming to construct their own model to predict the Federal Funds Rate. This project could serve as a springboard for those who want to construct their own regression models to predict the Federal Funds Rate. This project will explore the contributions of selected features to predictive performance and may help others make better predictions for the Federal Funds Rate.

Financial institutions and lenders, such as auto loan lenders, can benefit because the Federal Funds Rate is linked to economic activity, affects various interest rates, and affects the value of US government bonds. Changes in the federal funds rate typically lead to similar changes in interest rates on credit cards and auto loans. These groups could then prepare for policy and interest rate changes.

Being able to predict changes in the federal funds rate in advance would present opportunities for investors to make some money in the money (bond) market or from trading futures, a financial derivative which tracks the value of the Federal Funds Rate. The Federal Funds Rate has an inverse relationship with the prices of government bonds. As the Federal Funds Rate rises, bond prices tend to fall until they mature. As federal funds rate decreases, bond prices rise. This happens because the government will continue issuing bonds, but the new bonds will have better yields, making the existing ones less attractive. Financial institutions and investors can also use predictions of the Federal Funds Rate to adjust their portfolios as part of risk management.

# Project Scope

This is a small research project, worked on solely by me, using publicly available datasets downloaded from a database, Federal Reserve Economic Data (FRED). There are no teams or stakeholders involved in this project’s development and completion.

The project uses various open-source Python libraries that are all highly permissive. Use of these libraries for this project will not violate any terms of the licenses. None of the licenses require making notifications of their usage.

We want to reproduce and validate existing models and explore relationships between variables by analyzing various metrics, statistics, and plots from the models, checking for feature contribution to predictions of the Federal Funds Rates. The model is not intended for production or for obtaining accurate real time predictions of the Federal Funds Rate.

## Work Breakdown Structure

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Task | Dependencies | Status | Effort Hours | Start Date | Planned Completion | Actual Completion |
| 1 | Capstone Milestone 1 | 1.2 | Complete | 9 | 10/7 | 10/30 | 10/30 |
| 1.1 | Rough Draft |  | Complete | 3 | 9/05 | 10/7 | 10/7 |
| 1.2 | Topic Change |  | Complete | 9 | 10/7 | 10/16 | 10/16 |
| 2 | Capstone Milestone 2 | 2.1 | Complete | 7 | 10/31 | 1/1 | 1/1 |
| 2.1 | Rough Draft | 1.2 | Complete | 4 | 10/31 | 1/1 | 1/1 |
| 3 | Capstone Milestone 3 | 3.1, 3.2, 3.3, 3.4 | Complete | 23 | 1/9 | 3/5 | 3/5 |
| 3.1 | Implementation Plan |  | Complete | 1 | 1/9 | 3/5 | 3/5 |
| 3.2 | Jupyter Notebook Report (to include Systems Entities and Functional Requirements) | 3.1 | Complete | 7 | 1/9 | 3/5 | 3/5 |
| 3.3 | Web Dashboard Source Code | 3.2 | Complete | 12 | 1/9 | 3/5 | 3/5 |
| 3.4 | User guide | 3.3 | Complete | 3 | 1/9 | 3/5 | 3/5 |
| 4 | Capstone Milestone 4 | 3.4, 4.1, and 4.2 | Complete | 8 | 4/3 | 4/16 | 4/16 |
| 4.1 | Results Analysis or Testing Components |  | Complete | 4 | 4/3 | 4/16 | 4/16 |
| 4.2 | Systems Administration Guide |  | Complete | 4 | 4/3 | 4/16 | 4/16 |
| 5 | Milestone Final Submissions | 5.1, 5.2, 5.3, and 5.4 | Complete | 36 | 3/19 | 4/16 | 4/30 |
| 5.1 | Capstone Milestone 1 | 1 | Complete | 10 | 3/19 | 3/19 | 3/19 |
| 5.2 | Capstone Milestone 2 | 2 | Complete | 8 | 3/20 | 3/26 | 3/26 |
| 5.3 | Capstone Milestone 3 | 3 | Complete | 12 | 3/27 | 4/2 | 4/2 |
| 5.4 | Capstone Milestone 4 | 3.4, 4, 4.1, and 4.2 | Complete | 8 | 4/3 | 4/16 | 4/16 |
| 6 | Complete Capstone |  | Complete | 30 | 4/14 | 4/30 | 4/30 |
| 6.1 | Milestone Component Combination | 6.1, 6.2, 6.3, and 6.4 | Complete | 16 | 4/14 | 4/30 | 4/30 |
| 6.2 | Digital Poster |  | Complete | 2 | 4/17 | 4/30 | 4/30 |
| 6.3 | Video Sample |  | Complete | 4 | 4/17 | 4/21 | 4/23 |
| 6.4 | Final Video | 6.3 | Complete | 8 | 4/22 | 4/30 | 4/30 |

## Project Completion

There are no missing values and duplicate rows, all data types are float64, and a table shows all the features needed.

Summary statistics, plots of the data over time, plots of the distributions of the data, and a correlation heatmap are key components of Exploratory Data Analysis.

Test statistics or p-values are obtained for the Jarque-Bera (normality of residuals), Breusch-Pagan (constant variance of residuals), Durbin Watson (independence), and Rainbow Test (linearity).

Variance Inflation Factors (VIFs) will be used to check for multicollinearity. The threshold is 5.

Comparison of the prediction plots for the Karakas and Taylor models with plots found in Karakas’ paper will be used to determine if we have reproduced the models by checking that the models capture the same patterns.

Performance metrics include R-Squared, Root Mean Squared Error, Mean Squared Error, Mean Absolute Percentage Error, Akaike Information Criterion (AIC), and Bayesian Information Criterion (BIC). Lower values for each of these, except for R-Squared, indicate better model performance or quality. Higher values of R-Squared values indicate more variance explained and indicate better performance.

P-values, if model assumptions are satisfied for the model, for each coefficient will be used for hypothesis testing and checking for statistical significance of the feature. The F-statistic and the Prob(F-stat) can be used to perform an F-test, a hypothesis test to check significance of the model by examining if the model performs better than a model with no variables.

For the Neural Network Models, we will use error metrics and R-squared.

## Project Completion Criteria

|  |
| --- |
| Datasets are loaded and combined into a single data frame. |
| Data is cleaned and necessary features are derived. |
| Exploratory Data Analysis is performed. |
| All models are fitted and predictions obtained. |
| Determination of whether the models violate any regression assumptions is completed |
| Plots to compare model predictions with are constructed for the Taylor and Karakas models and capture the same patterns as the plots in Karakas’ paper. |
| Performance metrics and statistics are computed and analyzed for each model, as appropriate. |
| Neural Network Models are additionally constructed. |
| Plots of predictions and performance metrics are obtained for each Neural Network Model |
| Summary of findings (Conclusion) is written |
| Application (Jupyter Notebook and Streamlit Web Dashboard) run without errors and produce consistent results. |

## Assumptions and Constraints

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Description | Comments | Type | Status | Date Entered |
| 1 | Variables | The variables used will be Real Interest Rate, Real Inflation Rate, Output Gap (and % of 1st lag), Inflation Gap (and % of 1st lag), Federal Funds Target  , and Unemployment | Constraint | Confirmed | 10/7, edited 3/19 |
| 2 | Deriving features | Inflation, Inflation Lag, Output Gap, and Output Gap Lag % need to be derived. | Constraint | Confirmed | 3/19 |
| 3 | Vector form refers to 1st lags of the variables | Karakas did not state how the vector forms of the features he used were derived, but we will assume it’s related to lags of the variables and that the 1st lag is sufficient | Assumption | Confirmed | 3/19 |
| 4 | Inflation target is a constant | This is set by the federal reserve at 2% | Assumption | Confirmed | 10/16 |
| 5 | Real interest rate is treated as a constant | If we were to do so, it would be set as 2%, like inflation target | Assumption | Confirmed | 10/16 |
| 6 | The regression equations have multiple constants. | The sum of the constants will be considered the intercept. | Assumption | Confirmed | 3/19 |
| 7 | The date range used will be less than the one Karakas used in their models. | The datasets of some variables have smaller date ranges. Rather than construct the models twice (before and after removing rows), we will work with a smaller date range, since we want to compare models. | Constraint | Resolved | 3/19 |
| 8 | Federal Funds Target has varying formats. | Part of the data on this uses single values, and the more recent parts use a date range. We will combine them by taking the midpoint of the ranges, then combining the 2 into a single feature/column. | Constraint | Resolved | 3/19 |
| 9 | Time series frequency differences | Some datasets are in quarterly frequency and others in yearly or monthly. We will convert all data to daily frequency by imputing values. For example, the value for one year becomes the observation for each day in the year. | Constraint | Resolved | 3/19 |

# Project Controls

## Risk Management

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Event Risk** | **Risk Probability**  **(high, medium, low)** | **Risk Impact** | **Risk Mitigation** | **Contingency Plan** |
| What is the risk? | What is the probability? | What is the impact if the risk occurs? | What can be done to minimize the risk? | What can be done to minimize the impact of the risk? |
| Data is not in consistent frequency | High | Models may become inaccurate | We will have to change all the datasets to have the same frequency | Impute values accordingly to match daily frequencies |
| Problems with Plotly Interactive Charts when exporting Jupyter Notebook | High | Plots do not render in HTML exports | Resolve technical difficulties by using nbconvert or embed the plots. | Switch to a different library such as Matplotlib or Seaborn and generate multiple plots. Or use static images of the interactive plots |
| Date ranges inconsistent across datasets | High | The offending datasets will have missing values outside of their range | This is how the data is, and we can only work with it. Delete rows to shrink the date ranges of every dataset to match the smallest date range. | We could try selecting date ranges when downloading the data, but that will create an additional missing value when we find the lags for the necessary features. |
| Missing Values | Low | Will raise errors if left unattended. | Check for missing values and impute with other values or delete the offending row from the data. | Deletion or imputation of values and rows until there are no missing values |
| Duplicate values might appear when merging the datasets | Low | Causes biases in results if left unattended. | Ensure that the code for merging the datasets is written correctly. | Check for duplicates before and after merging the datasets to ensure there are nmo issues |
| Application (web dashboard) breaks | Low | The application stops functioning | Check logs to determine the cause. | Remove the application from the cloud |

## Change Control Log

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Change Description** | **Priority** | **Date Entered** | **Status** | **Included in Rev. #** |
| 1 | Switched from Scikit-Learn to Statsmodels for fitting | High | 2/10 | Complete | 2 |
| 2 | Created a web dashboard | High | 2/10 | Complete | 2 |
| 3 | Deployed the dashboard | High | 2/14 | Complete | 2 |
| 4 | Added Plotly Interactive Charts for aesthetic and to reduce the amount of plots needed | Low | 3/13 | Complete | 2 |
| 5 | Scrapped dashboard application | High | 4/2 | Complete | 3 |
| 6 | Added model assumption checks, correlation heatmaps, and switched from Plotly to Matplotlib | High | 4/2 | Complete | 3 |
| 7 | Added plots of data distributions, neural network implementation, and converted some results to tables | High | 4/14 | Complete | 4 |
| 8 | Created another Streamlit Dashboard | Med | 4/28 | Complete | 4 |

## Project Schedule

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Task | Subtask | Priority | Start Date | End Date | Resources Required | Contingencies |
| 1 | Milestone 1 |  | High | 10/7 | 10/30 |  | File backups, work on another project that could be passable as another topic |
| 2 | Milestone 2 |  | High | 10/31 | 1/1 |  | File backups, work on another project that could be passable as another topic |
| 3 | Milestone 3 | Implementation Plan, Jupyter Notebook Report, User Guide, Functional Requirements, System Entities | High | 1/9 | 3/5 |  | File backups, work on another project that could be passable as another topic |
| 4 | Milestone 4 | Test case or results analysis, Systems Administration Guide | High | 4/3 | 4/16 |  | File backups, work on another project that could be passable as another topic |
| 5 | Explore publishing options |  | Low | 4/24 | 4/26 |  |  |
| 6 | Complete Capstone | Revision of all milestones, combination of milestones, video presentation, digital poster | High | 3/6 | 4/30 | Instructor feedback | File backups, work on another project that could be passable as another topic |

### Gantt Chart

## Cost Estimate

This is not applicable since the project will not require any materials that need to be paid for. Some of the resources used are:

* Jupyter Notebook
* Python and relevant libraries
* Streamlit and Streamlit Community Cloud for deployment
* Git for version control and project hosting
* MS Visual Code for coding
* The datasets were downloaded for free from the FRED database
* Computer, electricity, and time for work and testing on the project

## Issues Log

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Description** | **Project Impact** | **Action Plan/Resolution** | **Importance** | **Date Entered** | **Date Resolved** |
|  | What is the issue? | How will this impact scope, schedule, and cost? | How do you intend to deal with this issue? |  |  |  |
| 1 | Data is not in consistent frequency | This could affect model performance and lead to inaccurate results. | Impute values by rolling over existing ones. For example, take a yearly value and make it the same for each day in that year. | High | 10/16 | 2/5 |
| 2 | Voila framework cannot load the application | This prevents work on and completion of the web dashboard application | Treating time as a sunk cost and switching to a different framework like Streamlit | High | 2/3 | 2/5 |
| 3 | Streamlit application shows the welcome message instead of running locally | This prevents work on and completion of the web dashboard application | Use --server.headless true when running the application in command line | High | 2/5 | 2/5 |
| 4 | Model Plots look the same for Karakas and Taylor | Results do not match expected results | Reread Karakas’ paper and see if something was overlooked | High | 2/10 | 2/25 |
| 5 | Jupyter Notebook Export does not render interactive plots | If this issue cannot be resolved, we need to use static plots made with Matplotlib or Seaborn | Try using NBconvert or plot embedding. Worst case scenario is to switch to static plots made by Matplotlib and Seaborn (which have already been made). | Low | 3/17 | 4/2 |

# Requirements Analysis

## Use Cases

**Use Case 1: Karakas Model Replication and Validation**

An economist or someone interested in economic research, reads Karakas’ (2023) paper and wants to check if Karakas’ work is reproducible, which is an important concept in research. This requires two plots: the first of Taylor vs Actual and the second of Taylor vs Karakas vs Actual. Additionally, Karakas used the Sum of Residuals and Sum of Absolute Errors as metrics for their models. The economist accesses the web dashboard application through a URL link. The following is a list of things expected to happen:

* The system loads the data, preprocesses it, generates features, constructs the model, computes performance metrics, and makes visualizations
* The economist should be able to:
  + Click on the “Regression Assumptions” tab
    - View and expand the column width of the tables with the Test Statistics and P-values for Jarque-Bera, Breusch-Pagan, Durbin-Watson, and Rainbow tests

A screenshot of a computer

AI-generated content may be incorrect.

* + - View and expand the column width of the table of Variance Inflation Factors
    - Click on boxes each of the above tables that expand and contain analysis underneath

A screenshot of a computer

AI-generated content may be incorrect.

* + Click on the “OLS Model Results” tab and navigate to the “Validation of Models from Karakas' Paper” subheader

A screenshot of a computer

AI-generated content may be incorrect.

* + - Click on the "Taylor and Karakas Model Metrics Comparison" box to view the Sum of Residuals, Sum of Absolute Errors, and analysis of these for the Taylor and Karakas models.
    - Click on the “Taylor vs Karakas Predictions Plot" box to view a plot of the Taylor and Karakas predictions plotted against each other
    - Navigate to the “Model Predictions Plot” and select the Taylor Model to visualize (selected by default) and scroll down to see analysis on how it compares to the plot found in Karakas’ paper.

**Use Case 2: Jupyter Notebook Use**

A user (economist, educator, investor, or financial institution) wants to view this project in its entirety but does not want to use the Streamlit Web Application. Instead, the user wants to download and run all the code in the Jupyter Notebook.

The user downloads the .ipynb file from the GitHub Project Repository and stores it on their device. The user downloads Jupyter Notebook, Python and all necessary libraries if they do not have them installed already.

The user then runs Jupyter Notebook and opens the .ipynb file, then clicks on Run -> Run All Cells from the toolbar at the top.

A screenshot of a black screen

AI-generated content may be incorrect.

The following is a list of things expected to happen:

* The Jupyter Notebook executes all the code in order from top to bottom, performing the following tasks:
  + loads the data, preprocesses it, generates features, constructs the model, computes performance metrics, and makes visualizations
* All code outputs are generated, and all markdown cells are rendered
* The user should be able to:
  + Use a table of contents to see all sections and navigate the Notebook by clicking on View -> Table of Contents from the top toolbar or using keyboard shortcut Ctrl + Shift + K.

A screenshot of a computer

AI-generated content may be incorrect.

* + View, expand or collapse cells or code outputs

A screenshot of a computer

AI-generated content may be incorrect.

View Cells and Output

A screenshot of a computer

AI-generated content may be incorrect.

Collapse Output

A screenshot of a computer screen

AI-generated content may be incorrect.

Collapse Code Cell

* + Obtain identical results when rerunning the notebook by clicking on Run or Kernel -> Restart the Kernel and Run All Cells from the top toolbar
  + Edit the code and change model parameters

## System Design

## Technical Requirements

* Device with internet access since the application is deployed and the Jupyter Notebook file is hosted online
* Web browser such as, but not limited to, Google Chrome or Microsoft Edge. This will be used to run the application in the browser
* PDF viewer to view the PDF Jupyter Notebook export
* JavasScript for viewing interactive plots in the HTML Jupyter Notebook export
* Python libraries such as Matplotlib, Numpy, Pandas, Plotly, Scikit-Learn, Seaborn, Statsmodels, and Tensorflow installed on the system if using the Jupyter Notebook

## Data Modeling and Analysis Process

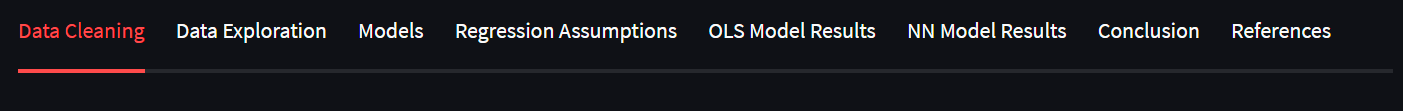
## Reports

The application does not generate reports. A Jupyter Notebook Report in HTML and PDF format will be created and provided in the repository.

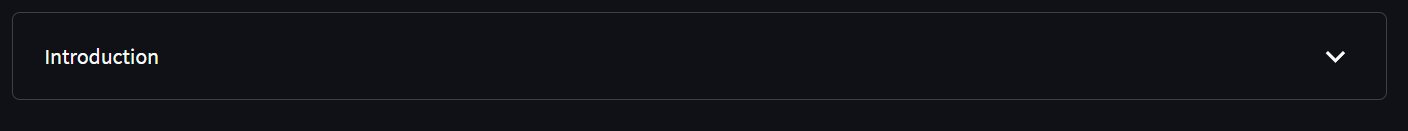
## Screen Definitions and Layouts

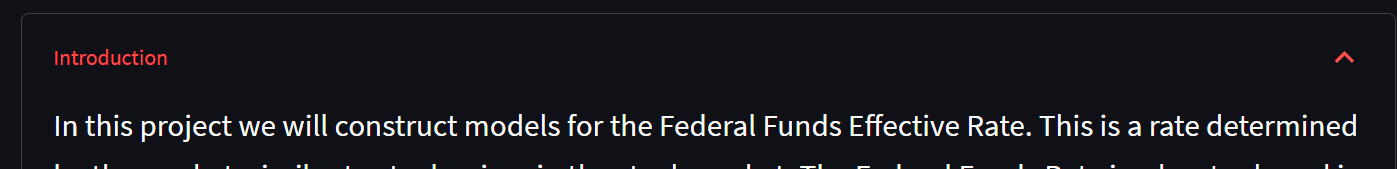
The dashboard will include the title, tabs, interactive tables (full screen mode, adjustable column width, table sorting), selectors, and expandable/collapsible boxes.

The user can navigate to different sections by clicking on the tabs and scrolling up and down to view contents within the tabs:

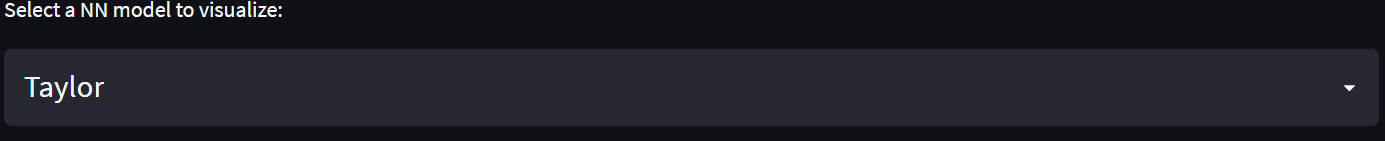


Some outputs are placed inside expandable and collapsible boxes to save screen space:





Selectors are available for the user to pick which model to plot:



# Model Pipeline Design

## Design Planning Summary

In this project we will construct models for the Federal Funds Rate, which we will just call the Federal Funds Rate. Predicting the Federal Funds Rate benefits educators, economists, investors, financial institutions, and policy planners.

We will attempt to reproduce the two regression models found in “Reevaluating the Taylor Rule with Machine Learning” by Karakas (2023), by modeling the equations stated in the paper. We will refer to those two models as the Taylor Model and Karakas Model.

In their paper, Karakas (2023) found that there was little difference in performance between the Taylor Model and their own regression model and neither predicted the Federal Funds Rate well, so Karakas switched to the use of neural network models, which performed better using the same features. Therefore, we want to see if we can add features such as Federal Funds Target and Unemployment Rate to the Taylor Model to see if this can improve model performance. While Karakas limited the neural network to the same features used in Taylor’s Rule, we will deviate from Taylor’s Rule and explore the impact other features could have on the Taylor Model.

We will then check the classical assumptions of regression for each model and compare the predictions of the Taylor and Karakas models to the plots found in Karakas’ paper. In research, it is important that work is reproducible and validated because it adds credibility and authority to solid work and screens out any unqualified work.

We will then compute and analyze performance metrics for each model. We will do the same for the Neural Network Models (they will not need assumption tests though).

## Overview of Design Concepts

Data will first be downloaded from Pandas, then uploaded to a Github Repository. Each dataset will then be loaded and merged through the Pandas library in Python.

Duplicate rows are checked, and the data is resampled to daily frequency. Rows will be deleted from the dataset until we have no rows with any missing values in any column. We expect there to be missing values since the data sets may have different ranges of available observations. We will restrict the data to the date range with all available data. Outliers will be capped via Winsorization, and necessary features will be derived.

A summary statistics table, plots of each variable over time, plots of their distributions, and a correlation matrix will be constructed for Exploratory Data Analysis. The data will be standardized, since the values can vary in magnitude, through MinMaxScaler.

Models will then be fitted using Ordinary Least Squares regression and classical regression assumptions will be tested by computing the corresponding test-statistics (Jarque-Bera, Breusch-Pagan, Durbin Watson, and Rainbow tests) and Variance Inflation Factors (VIFs).

Performance metrics and plots will be computed and generated after the models are fitted. Performance metrics will include Mean Squared Error, Root Mean Squared Error, Mean Absolute Error, Mean Percentage Error, Mean Absolute Percentage Error, R-squared, Adjusted R-squared, Akaike Information Criterion (AIC), and Bayesian Information Criterion (BIC). We will examine the metrics and interpret them as appropriate. If regression assumptions are violated, we will additionally fit neural network models for the data, using similar performance metrics.

The regression equations are derived as follows:

(1) it = r\* + πt + 𝛽π (πt- π\*) + 𝛽y (yt - yt\*)

Where:

it = federal funds rate at time t

πt\* = inflation rate at time t

π\* = inflation target [2% or 0.02, as set by the Fed]

r\* = real interest rate

yt = Real Gross Domestic Product (GDP)

yt\*= Potential GDP

(Karakas 2023).

The difference between Real GDP and Potential GDP (yt - yt\*) is the output gap and difference between Inflation Rate and Inflation Target (πt- π\*) is the inflation gap. Note that we have 2 constants in the equation. In this equation, r\* and πt are both constants. Running the regression will yield only one constant term. Mathematically, this constant would be the sum of the Inflation rate and the Real Interest Rate. So, for clarity, we could modify the equation by substituting the constant terms (πt + r\*) with c as follows:

(2) it = c + 𝛽π (πt - π\*) + 𝛽y (yt - yt\*)

The equation for Karakas’ (2023) Model is stated to be as follows:

(3) it = r\* + 𝛽1 (πt) + 𝛽π (πt - π\*) + 𝛽y (yt - yt\*)

Karakas (2023) mentions this equation suffers from multicollinearity issues between πt and (πt - π\*), so further operations are performed below:

(4) it = r\* + 𝛽1 (πt) + 𝛽π (πt) - 𝛽π (π\*) + 𝛽y (yt - yt\*)

(5) it = r\* - 𝛽π (πt\*) + (𝛽π + 𝛽1) (πt) + 𝛽y (yt - yt\*)

(6) it = r\* - 𝛽π (π\*) + (θπ) (πt) + 𝛽y (yt - yt\*), where θπ = (𝛽π + 𝛽1)

Since Karakas (2023) treated the term r\* - 𝛽π (π\*) as the constant intercept, for clarity, I would substitute that term with e and change the equation to:

(7) it = e + (θπ) (πt) + 𝛽y (yt - yt\*)

Inflation and Output Gap for this model will need to be converted to vector form and the Output Gap needs to be converted to a percentage, since Karakas (2023) mentioned doing so in their paper. We will find the first lag as part of conversion to vector form.

Taylor Model:

it = c + 𝛽π (πt - π\*) + 𝛽y (yt - yt\*)

Karakas Model:

it = e + (θπ) (πt) + 𝛽y (yt - yt\*)

We want to add features, such as Federal Funds Target and Unemployment Rate, to the model in equation 2. We will use 𝛽tar to represent the Target and 𝛽u to represent the Unemployment Rate. We will construct 3 additional models by adding the Target or Unemployment Rate individually to the model, then build a third model (named ‘Both’) by adding both features simultaneously.

Target Model:

(8) it = c + 𝛽π (πt - π\*) + 𝛽y (yt - yt\*) + 𝛽tar

Unemployment Model:

(9) it = c + 𝛽π (πt - π\*) + 𝛽y (yt - yt\*) + 𝛽u

Both Model:

(10) it = c + 𝛽π (πt - π\*) + 𝛽y (yt - yt\*) + 𝛽tar + 𝛽u

We will model each equation using the appropriate features for each variable. However, we are interested in differences in model performance when different features are used rather than in the estimations of the coefficients.

## Deliverable Acceptance Log

|  |  |  |
| --- | --- | --- |
| ID | Deliverable Description | Comments |
| 1 | Jupyter Notebook Report | A copy of the Jupyter Notebook, showing the code to load and prepare the data, build models, display regression results and visualizations, and analysis of results. |
| 2 | Application | A file containing the source code for the web dashboard application to show regression results and visualizations. |
| 3 | Links to the Repository | A link to the GitHub Repository where someone could find the datasets, Jupyter Notebook file, library dependencies document (requirements.txt) and application source code will be provided. |
| 4 | Link to WebApp | A link to the Streamlit Web Application will also be provided. Enter the URL into a web browser with internet access to run the application. |
| 5 | Documentation | User guide, functional requirements, system entities, implementation plan, systems administration guide, project proposal, pipeline design, and results analysis/testing components. |
| 6 | Video | Video presentation of project, procedure, and analysis of results. |
| 7 | Digital Poster | Quick snapshot of the project and references on a single page or slide |

## Overview of Model Pipeline Design

The design prepares the data for exploration and regression before constructing multiple models for the Federal Funds Rate. Two of those models are based on given equations for the purpose of reproducing procedure and results. The other three models include additional features added to the Taylor Model. The models are then compared to each other using various performance metrics and the plots of their predictions. Since these additional models differ from the Taylor Model with the addition of one or two features, we can evaluate the impact of those features on the model. We will be fitting each model twice, once with Ordinary Least Squares Regression and another time with simple Neural Networks.

## Detailed Model Pipeline Design

### 1. The data sources

All the necessary datasets can be found on FRED. This is a database for economic data where most, if not all, of the data is collected by government agencies, central banks, and financial institutions which makes them very credible compared to other sources.

The data that we will be using comes from sources like the Board of Governors of the Federal Reserve System (U.S), U.S. Bureau of Economic Analysis, U.S. Bureau of Labor Statistics, U.S. Congressional Budget Office, and World Bank. Please refer to the appendix for citations and links for the data used.

The datasets include Federal Funds Rate, Federal Funds Target (old, upper bound, and lower bound), Real Consumer Price Index, Real GDP, and Potential GDP.

The data will be downloaded as Excel CSV files, then uploaded to a Github Repository. Pandas will read URL links to the datasets through the read\_csv() function.

Inflation, Inflation Gap, Output Gap, Inflation Lag, and Output Gap Lag % need to be computed. Inflation will be computed by using the pct\_change() function over 12 periods, then multiplying by 100%.

Inflation Gap can be found by subtracting 2% from Inflation, since we are treating the inflation target as a constant 2% like Karakas did.

Output Gap is computed by subtracting Potential GDP from Real GDP.

The shift function will be used to find Inflation Lag and Output Gap Lag, then multiply Output Gap Lag by 100% (inflation is already a rate) to get Output Gap Lag %.

### 2. The dataset types and formatting

All data should include dates as strings and time series data on each variable in float64. The data sets will differ in frequency.

Quarterly: Real and Potential GDP

Monthly: Consumer Price Index, Unemployment Rate

Daily: Federal Funds Rate and Target

### 3. The data cleaning procedure

First, each data set is checked for any duplicate rows or missing values (there should not be any of them). Next, all datasets that do not use daily frequency are resampled based on previous observations to match daily frequency. For example, values for one quarter will be treated the same for every day in that quarter. The data is then merged around the dates, which are then converted to an index. To ensure that the data is merged as expected, duplicates and missing values are double checked for. There should be no duplicates, but there should be missing values since the datasets have different ranges of available data.

Next, rows with missing values are deleted from the data frame, leaving only date ranges with available data. This is done since there will be missing values as the datasets have different time ranges and observations outside of that range automatically become missing values.

Additional features will be derived. We will then check the types of each feature the data frame. The ideal data type is float for each feature and the data type should be changed to this as needed.

Outliers will be capped using Winsorization to the 5th and 95th percentiles. Our interest lies in the features and their relationships rather than optimizing prediction accuracy.

MinMaxScaler will be used to standardize the data before modeling since the magnitude of values can vary. RobustScaler is not used because we will have dealt with outliers already and StandardScaler is not used since we will not be assuming the data is normally distributed.

### 4. Method of initial data exploration and visualization

A table of the data and the summary statistics will be created, including measures such as the mean, min, and max values. Line plots of each feature over time and plots of their distributions will be constructed. A correlation heatmap for the features will be created. The plots and heatmap will be analyzed in depth.

### 5. The data model used and its nature (e.g., predictive)

We will be making five models that are fitted twice, once with Ordinary Least Squares (OLS ), and another time with a simple Neural Network. Each model will be regression models. These models will be predictive in nature rather than explanatory, since we are interested in examining and comparing predictive performance with different features. We are uninterested in estimates of the coefficients.

The dependent variable for all the models will be the Federal Funds Rate.

Independent variables to be used include Inflation Lag, Inflation Gap, Output Gap, Output Gap Lag %, Federal Funds Target, and Unemployment Rate.

We will fit the models with OLS using the Statsmodels library and with Neural Networks using the Tensorflow library.

Model List:

Taylor Rule Model:

* Dependent variable: Federal Funds Rate.
* Independent variables: Output Gap, Inflation Gap

Karakas Model:

* Dependent variable: Federal Funds Rate.
* Independent variables: Output Gap Lag %, Inflation Lag

Target Model:

* Dependent variable: Federal Funds Rate.
* Independent variables: Output Gap, Inflation Gap, Target

Unemployment Model:

* Dependent variable: Federal Funds Rate.
* Independent variables: Output Gap, Inflation Gap, Unemployment

Both Model (uses both Unemployment and Target):

* Dependent variable: Federal Funds Rate.
* Independent variables: Output Gap, Inflation Gap, Target, Unemployment

### 6. The methodology for interpreting the analysis results

The performance metrics computed are Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Percentage Error (MPE), Mean Absolute Percentage Error (MAPE), R-squared, Adjusted R-squared, Akaike Information Criterion (AIC), and Bayesian Information Criterion (BIC).

For the error metrics, values closer to 0 are better since that means fewer errors. The Information Criteria are used to compare different models, with models having lower values being better than models having higher values. For the R-squared and Adjusted R-squared, values closer to 1 are better, as these measure how much variation a model explains.

For the line plots, which show the actual rates and each model’s predictions, we can visually judge the predictive strength of each model and how they compare to each other, which may be more intuitive than looking at numbers. We will compare the plots of the Taylor Model and Karakas Model to the plots shown in Karakas (2023) paper.

Hypothesis testing will be performed on the significance of the feature coefficients (T-test) and the model itself (F-test), and various test statistics for regression assumptions.

The Jarque-Bera test checks if the distribution is normal, the Breusch-Pagan test checks for homoscedasticity, and the Rainbow Test checks for linearity.

Significance level (α) of 0.05 will be used for the hypothesis test when comparing p-values, rejecting the null hypothesis at p-values less than 0.05.

The null hypothesis is that the data satisfies the assumption, and the alternative hypothesis is that the data violates the assumption. Rejection of the null hypothesis, therefore, would be bad for the above tests.

The Durbin Watson test will check for autocorrelation of residuals. Values close to 2 mean that there is no autocorrelation of residuals. Values deviating from this by over of 0.5 indicate autocorrelation. We determine if the data violates the assumptions by checking if the Durbin Watson test statistic lies outside of the range 1.5 to 2.5.

### 7. This section should also include any configuration changes that will be required to develop and implement the proposed solution.

The following list includes the version of Python, and the libraries used to develop and run the application:

* Python version 3.9.7
* altair==4.2.2
* ipython==8.18.1
* keras==3.9.2
* matplotlib==3.9.4
* notebook==7.4.1
* numpy==2.0.2
* pandas==2.2.3
* pyarrow==20.0.0
* scikit-learn==1.6.1
* scipy==1.13.1
* seaborn==0.13.2
* statsmodels==0.14.4
* streamlit==1.12.0
* tensorboard==2.19.0
* tensorflow==2.19.0

The data will be uploaded to a Github Repository so we can obtain raw file URLs, which will be used to load the data. The source code and requirements.txt files will also be uploaded to the repository detailing configurations.

### 8. Describe the approach and resources required to assure system security, if applicable; otherwise, explain why security is not relevant.

The datasets that will be used are all publicly available and do not contain any private or sensitive information from any individual or institution. Several of them are collected and shared by government sources.

The application uses a fixed set of data sets. User uploads or user inputs (for URLs) will not be necessary or implemented, which eliminates some vulnerabilities and security considerations through user interaction.

The model’s source code is not secretive. Also, Streamlit uses HTTPS and encrypts data in transit (“Streamlit Docs,” 2025).

System security is not relevant for these reasons.

### 9. List the hardware and software technologies.

|  |
| --- |
| 1. Python and necessary libraries, such as Matplotlib, Numpy, Pandas, Plotly, Scikit-Learn, Seaborn, Statsmodels, and Tensorflow |
| 1. Jupyter Notebook for code and project documentation |
| 1. Streamlit for application (dashboard) development |
| 1. Streamlit Community Cloud for deployment |
| 1. Github to upload the data and necessary files, such as requirements.txt and the application source code, to a repository |
| 1. Computer or laptop to develop the application |
| 1. Device with web browser and internet access to run the application |

# Implementation

## Implementation Plan

The applications will take the form of a Jupyter Notebook “.ipynb” file and a Streamlit Web Application. Either can be used according to preference.

The applications use Python and the following libraries: Pandas, Numpy, Matplotlib, Seaborn, Statsmodels, Scikit-Learn, Scipy, Streamlit, and Tensorflow libraries. Most of the Python libraries are used to manipulate the data, create models, compute metrics, check regression assumptions, and construct visualizations.

The Jupyter Notebook will Jupyter, Python, and all the previously mentioned libraries except for Streamlit to be installed to run locally. A PDF copy of the Jupyter Notebook will only require a PDF reader to read all the code, analyses, and view code outputs.

In Jupyter Notebook, the user will be able to run the code and obtain all the intended outputs. This includes collecting the data, cleaning it, constructing models and visualizations, computing metrics, and checking assumptions. The Jupyter Notebook also contains written analysis in the markdown cells underneath coding cells and outputs as appropriate for a narrated presentation.

A Streamlit Web Application will also be developed. With the Streamlit Web Application, the user will not need to have Python, or any libraries installed on their system, but they will require a web browser with internet access to open and run the application. The Streamlit Web Application organizes the outputs and visualizations and allows some user interactivity such as selecting which model to plot or sorting tables by columns.

The applications will use data on Consumer Price Index, Federal Funds Rate (FFR), Target, Real GDP, Potential GDP, Real GDP, and Unemployment to construct Regression models. The Consumer Price Index is used to derive the Inflation rate. Potential and Real GDP are used to derive the output gap. Inflation is then used to find the Inflation Gap, and Inflation Lag. We will then find percentage versions of Output Gap and Inflation Lag to replicate the model created by Karakas (2023). The Federal Funds Target needs some additional processing to combine the old and new datasets.

With these variables we plan to fit 5 regression equations twice, using Ordinary Least Squares first, then a Neural Network.

The first and second equations are listed in Karakas’ Paper (2023). The first equation uses Output Gap and Inflation Gap, while the second equation uses Output Gap % and Inflation Lag% as independent variables. The third, fourth, and fifth equations will add new independent variables (Federal Funds Target and/or Unemployment Rate) to the ones used in the first model. The models of these equations are intended to explore how well Taylor’s Rule predicts the Federal Funds Rate and how the addition of the Federal Funds Target and Unemployment Rate affect the predictive power.

This project will attempt to replicate and validate some of Alper D. Karakas’ (2023) work in their paper. We will check for the classical regression assumptions, fit the OLS models, find performance metrics, and make plots of their predictions. We will repeat these steps excluding checking for assumptions with a Neural Network model implementation.

## System Entities

Libraries/Tools

* Matplotlib, Numpy, Pandas, Plotly, Scikit-Learn, Seaborn, Statsmodels, and Tensorflow
* Jupyter Notebook Environment

Cleaning

* Winsorization for outliers
* Duplicate and missing value check
* Data resampling

Exploratory Data Analysis

* Table of values (head and tail for first and last 5 values)
* Summary Statistics Table
* Line Plots of variables over time
* Plots of data distributions
* Correlation Heatmap

Models

* Ordinary Least Squares Regression
* Neural Network

Data Transformation Methods/Tools

* MinMaxScaler

Performance Metrics

* R-squared, adjusted R-squared, Akaike Information Criterion (AIC), and Bayesian Information Criterion (BIC) Assumption Tests
* Mean Squared Error, Root Mean Squared Error, Mean Absolute Error, Mean Percentage Error, and Mean Absolute Percentage Error
* Sum of Residuals and Sum of Absolute Error only for Taylor and Karakas model comparison

Assumption Test Statistics and P-values

* Normality (Jarque-Bera)
* Homoscedasticity (Breusch-Pagan)
* Autocorrelation (Durbin-Watson)
* Linearity (Rainbow)
* Multicollinearity (Variance Inflation Factors)

Visualizations

* Plots of model predictions against actual FFRs
* Plot of Taylor vs Karakas Predictions

## Functional Requirements

|  |  |
| --- | --- |
| Data Collection | Under “Data Wrangling”:   * Datasets dictionary * Dataframe, df * read\_csv() |
| Data Cleaning | Under “Data Wrangling”:   * df.dropna() * df.duplicated() * df.isna().sum() * df.concat() * df.merge() * winsorize() |
| Exploratory Data Analysis | Under “Exploratory Data Analysis”:   * df.shape * df.dtypes * df.describe() * sns.lineplot() * sns.histplot() * sns.heatmap(df.corr) |
| OLS Modeling | Under “OLS Modeling”:   * sm.OLS() * defined function: fit\_ols\_model (X, y, model\_name) |
| Regression Assumptions Checking | Under “Regression Assumptions”:   * het\_breuschpagan() * linear\_rainbow() * durbin\_watson() * jarque\_bera() * variance\_inflation\_factor() * defined function: calculate\_vif(X, model\_name) |
| Neural Network Modeling | Under “Neural Network Modeling”:   * defined function: fit\_nn\_model(X, y, model\_name) * 1 hidden layer with 32 nodes * 20% neuron dropout * L2 Regularization * 100 epochs with EarlyStopping |
| Metrics Computation | Under “OLS Model Metrics” and “Neural Network Model Metrics”:   * mse = round(mean\_squared\_error(y, y\_pred), 3) * rmse = round(np.sqrt(mse), 3) * mae = round(mean\_absolute\_error(y, y\_pred), 3) * mpe = round(np.mean((y - y\_pred) / y) \* 100, 3) * mape = round(np.mean(np.abs((y - y\_pred) / y)) \* 100, 3) * r2\_score(y, y\_pred) * defined function:error\_metrics(y, y\_pred) |
| Model Visualization | Under the following sections:  “Model Comparison of Taylor and Karakas Models to Karakas' Work” (Jupyter Notebook),  “Model Predictions Plot” (Streamlit), and  “Neural Network Prediction Plots”:     * plt.plot() [Jupyter Notebook] * st.pyplot() [Streamlit] |
| Documentation and Analysis | Explanations of results in markdown cells located below various code outputs as appropriate (Jupyter Notebook)  In Streamlit:   * st.markdown() |
| Streamlit Widgets | In Streamlit source code:   * st.markdown() * st.expander() * st.dataframe() * st.selectbox() |

## Non-functional Requirements

* Written analysis is correct and thorough
* Research questions answered (whether Karakas’ work can be replicated, validation results, impacts of additional features on performance is explained)
* Applications use accessible options
* Applications are organized
* Results are reproducible

## Source Code Listing

The Jupyter Notebook .ipynb file and the source code for the Streamlit Web Application are in a GitHub Repository. Here is the link to the Repository: <https://github.com/nvpham12/Capstone-Project>

## Coding Improvements

* Changed color palette to use dodgerblue, darkorange, and purple. Also used cividis colormap.
* Used loops and dictionaries to work iteratively on multiple models simultaneously.
* Set better hyperparameters for Neural Networks models.
* Implemented countermeasures for overfitting in Neural Network models.
* Used session states to save models and prevent the Streamlit Web Application from repeatedly fitting the models when changing models to plot.

# Results Analysis/Testing Components

## Module Test cases

### Test Case Name: Library Imports

Priority: Low

Module: 1

Objective: Load all the necessary libraries and functions for the application

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Test Detail | Expected Results | Problem/Issue |
| 1 | Import libraries: Matplotlib, Numpy, Pandas, Plotly, Scikit-Learn, Seaborn, Statsmodels, and Tensorflow | Loads without raising an error. | None |
| 2 | Import functions: mean\_squared\_error, mean\_absolute\_error, r2\_score, winsorize, variance\_inflation\_factor, durbin\_watson, jarque\_bera, het\_breuschpagan, linear\_rainbow, MinMaxScaler, Sequential, Dense, Input, EarlyStopping | Loads without raising an error. | None |

### Test Case Name: Defining Functions

Priority: Low

Module: 2

Objective: Create functions for modeling, computing Variance Inflation Factors (VIFs), and computing error metrics

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Test Detail | Expected Results | Problem/Issue |
| 1 | Define function to fit the model and get predictions. | Function is defined without errors. Errors are not raised when the function is used later. | None |
| 2 | Define function to calculate VIFs. | Function is defined without errors. Errors are not raised when the function is used in later modules. | None |
| 3 | Define function to compute error metrics such as MAE, MSE, etc. | Function is defined without errors. Errors are not raised when the function is used later. | None |
| 4 | Define function to fit the neural network model and get predictions. Uses 1 hidden layer, Relu, Linear, Adam optimizer, MSE loss, and MAE metrics. Also trains for 50 epochs and implements early stopping. | Function is defined without errors. Errors are not raised when the function is used later. | None |

### Test Case Name: Data Wrangling

Priority: Medium

Module: 3

Objective: Load all datasets and clean them

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Test Detail | Expected Results | Problem/Issue |
| 1 | Load all datasets with pandas.read\_csv() through a For Loop. | Load without raising an error and prints a message that each dataset loaded successfully. | None |
| 2 | Check each dataset for duplicate rows. | Datasets have no duplicate rows. | None |
| 3 | Derive necessary features, resample all datasets to daily frequency, merge all datasets, set date index, rename columns, drop missing values, drop unneeded columns and print the data table. | The table should be generated with the first date should be 1982-09-27, while the last date should be 2024-10-01.  The columns should be:   * Federal Funds Rate * Inflation (%) * Unemployment Target * Inflation Gap * Output Gap * Inflation Lag * Output Gap Lag % | None |
| 4 | Check for duplicates and missing values in merged dataframe. | There should be no duplicates or missing values. | None |
| 5 | Cap outliers using Winsorization. | Obtain a Winsorized dataframe without raising errors. | None |
| 6 | Check Data Types. | Data should be in float or int. | None |

### Test Case Name: Exploratory Data Analysis (EDA)

Priority: Low

Module: 4

Objective: Perform EDA by finding summary statistics, making plots of the data, and making a correlation heatmap

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Test Detail | Expected Results | Problem/Issue |
| 1 | Compute Summary Statistics with df.describe(). | Generates a summary statistics table. | None |
| 2 | Plot each feature over time. | Generates a line plot for each variable over time. The titles for each plot should have the variable name in the title. | None |
| 3 | Make plots of each feature’s distribution | Generates a histogram for each variable with kde curves. The titles for each plot should have the variable name in the title. | None |
| 4 | Generate a correlation heatmap using Seaborn. | Generates a colorful heatmap, including all column names and correlation values for each cell. | None |

### Test Case Name: OLS Modeling

Priority: High

Module: 5

Objective: Fit an OLS model for the 5 equations at once

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Test Detail | Expected Results | Problem/Issue |
| 1 | Apply MinMaxScaler | Data is scaled and dataframe overwritten | None |
| 2 | Set the variables for each model. | The dependent variable is set and a dictionary of features for each model is created without errors. | None |
| 3 | Initialize dictionaries to store results (fitted models, predictions, vifs, metrics, coefficient test statistics and p-values). | Empty dictionaries are generated. | None |
| 4 | Loop to fit the models, compute predictions, VIFs, metrics, and statistics. Store all results in the dictionaries. | Results are stored in the previously generated dictionaries. | None |

### Test Case Name: OLS Regression Assumptions

Priority: High

Module: 6

Objective: Compute all test statistics for each assumption test for every model. Then, check if the assumptions are valid.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Test Detail | Expected Results | Problem/Issue |
| 1 | Compute test statistic for Normality Test (Jarque Bera). | Statistic is computed successfully, without raising any errors. | None |
| 2 | Compute test statistic for Homoscedasticity Test (Breusch-Pagan). | Statistic is computed successfully, without raising any errors. | None |
| 3 | Compute test statistic for Autocorrelation Test (Durbin-Watson). | Statistic is computed successfully, without raising any errors. | None |
| 4 | Compute test statistic for Linearity Test (Rainbow). | Statistic is computed successfully, without raising any errors. | None |
| 5 | Compute test statistic for Multicollinearity (VIFs). | Statistics are computed successfully, without raising any errors. | None |

### Test Case Name: Taylor and Karakas OLS Model Comparison

Priority: High

Module: 7

Objective: Create compute specific error metrics and generate plots of predictions against actual FFRs for Karakas and Taylor Models.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Test Detail | Expected Results | Problem/Issue |
| 1 | Compute Sum of Residuals and Sum of Absolute Errors for Taylor and Karakas models. | Metrics are computed and printed for each model. | None |
| 2 | Plots of the Taylor and Karakas Model predictions are created. | There should be 2 plots:  Taylor vs Actual FFR and  Taylor vs Karakas vs Actual FFR. | None |

### Test Case Name: OLS Model Metrics

Priority: Low (since assumptions were violated)

Module: 8

Objective: Compute various performance metrics for each model (MAE, MSE, RMSE, MAPE, MPE, R-squared, AIC, BIC, f-stat, and p-values)

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Test Detail | Expected Results | Problem/Issue |
| 1 | Initialize dictionary for the model statistics. | Empty dictionary created. | None |
| 2 | Extract some performance metrics from the model results and store them in the model\_statistics dictionary. | Dictionary filled. | None |
| 3 | Make a table with the extracted performance metrics in the dictionary. | Table created and printed. Should have Adjusted R-squared, AIC, BIC, f-stat, their p-values and the  p-values for the  t-statistics. | None |
| 4 | Make another table by creating a dataframe from the ols\_error\_metrics dictionary (created when fitting the model). | Table created and printed. Should have MSE, RMSE, MAE, MAPE, MPE, and R-squared | None |

### Test Case Name: Neural Network Modeling

Priority: High

Module: 9

Objective: Since the OLS model assumptions are violated, we will use other methods, namely neural networks to model the data.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Test Detail | Expected Results | Problem/Issue |
| 1 | Initialize dictionaries to store results (fitted models, predictions, and metrics) | Empty dictionaries are generated. | None |
| 2 | Loop and store results for: all 5 fitted models, predictions, and metrics | Results are stored in the previously generated dictionaries. | None |
| 3 | Loop and print all metrics and values | Metrics (MAE, MSE, RMSE, MAPE, MPE, R-squared) are computed and printed for each model. | None |
| 4 | Plot Predictions for each model | Plots of the predictions are generated, with one plot per model. A For Loop is used to plot all of them at once, so they should all be in a single output. | None |

## Requirements Testing

### Component: Data Wrangling

Developer and Reviewer: Nicholas

|  |  |  |
| --- | --- | --- |
| Type | Pass | Comments |
| Datasets are loaded | Yes | Data loaded successfully. |
| Check for duplicate values | Yes | No duplicate values in any dataset. |
| Derive Inflation and merge datasets. Set date as an index | Yes | Inflation derived and all datasets merged into a single dataframe on date. Table of dataframe shows all variables from each dataset. Date is converted to an index. |
| Features (Inflation Gap, Output Gap, Inflation Lag, Output Gap Lag %) are created | Yes | Features created and shown on the table. |
| Check for duplicates again and missing values | Yes | No missing values or duplicates (no issues after merging). |
| Cap outliers. | Yes | Winsorization used to cap outliers based on 5% of extreme values. |
| Data Types are all in appropriate types | Yes | All columns are float64. |

### Component: Exploratory Data Analysis

Developer and Reviewer: Nicholas

|  |  |  |
| --- | --- | --- |
| Type | Pass | Comments |
| Summary statistics table generated | Yes | Table shows Count, Mean, STD, Min, Max, and percentiles. |
| First and last 5 rows of data are given as tables | Yes | Tables are generated correctly, based on the year in the date index. |
| Plots of each feature over time generated | Yes | All plots are labeled appropriately. |
| Distribution plots are generated for each feature | Yes | All plots are labeled appropriately. |
| Correlation Heatmap Generated | Yes | Heatmap generated with values in each cell. All variables are listed. |

Component: OLS Modeling

Developer and Reviewer: Nicholas

|  |  |  |
| --- | --- | --- |
| Type | Pass | Comments |
| Model is trained | Yes | All models were fitted using OLS without errors. |
| Model predictions are obtained and stored | Yes | All model predictions were stored in a dictionary without errors. |

### Component: OLS Regression Assumptions

Developer and Reviewer: Nicholas

|  |  |  |
| --- | --- | --- |
| Type | Pass | Comments |
| Compute test statistic for Normality Test (Jarque Bera) | No | Rejected null hypothesis (p-value < significance level) |
| Compute test statistic for Homoscedasticity Test (Breusch-Pagan) | No | Rejected null hypothesis (p-value < significance level) |
| Compute test statistic for Autocorrelation Test (Durbin-Watson) | No | Test statistics outside of bounds for Durbin-Watson test statistic. |
| Compute test statistic for Linearity Test (Rainbow) | No | Rejected null hypothesis (p-value < significance level) |
| Compute test statistic for Multicollinearity (VIFs) | Yes for all models except Both Model. | One VIF over 5, which indicates multicollinearity. This appeared for Output Gap in the Both Model. Other VIFs were below the threshold. |

### Component: Karakas and Taylor OLS Model Comparison (validation)

Developer and Reviewer: Nicholas

|  |  |  |
| --- | --- | --- |
| Type | Pass | Comments |
| Plot of Taylor vs Actual FFR created and resembles the one in Karakas’ paper | Yes | The plot shows the same patterns as the ones in Karakas’ paper |
| Plot of Taylor vs Karakas vs Actual FFR created and resembles the one in Karakas’ paper | Yes | The plot shows the same patterns as the ones in Karakas’ paper. |
| Sum of Residuals and Sum of Absolute Error metrics computed and results match what was found in Karakas’ paper. | No | Sum of Residuals was much lower than Karakas’ models. Both metrics were lower for the Taylor Model indicating it has better performance, a result contrary to what Karakas found. However, both metrics were extremely close indicating little difference in model performance, which is consistent with findings. |

### Component: OLS Metrics

Developer and Reviewer: Nicholas

|  |  |  |
| --- | --- | --- |
| Type | Pass | Comments |
| Adjusted R-squared, AIC, BIC, f-stats, and p-values are obtained. | Yes | All these metrics are used to construct a table. |
| Mean Squared Error, Root Mean Squared Error, Mean Absolute Error, Mean Percentage Error, Mean Absolute Percentage Error, and R-squared are obtained. | Yes | All these metrics are used to construct another table. |
| Analysis of all metrics is performed. | Yes | Constant of Taylor model is not significant. Model performance from least to greatest is Karakas < Taylor < Unemployment < Target < Both |

### Component: Neural Network Modeling

Developer and Reviewer: Nicholas

|  |  |  |
| --- | --- | --- |
| Type | Pass | Comments |
| Model trained | Yes | All models were fitted using a neural network without errors. |
| Predictions obtained | Yes | All neural network model predictions were stored in a dictionary without errors. |

### Component: Neural Network Metrics

Developer and Reviewer: Nicholas

|  |  |  |
| --- | --- | --- |
| Type | Pass | Comments |
| Mean Squared Error, Root Mean Squared Error, Mean Absolute Error, Mean Percentage Error, Mean Absolute Percentage Error, and R-squared are obtained. | Yes | A table is generated with these metrics. |
| Analysis of all metrics is performed. | Yes | Model performance from least to greatest is Karakas < Taylor < Unemployment < Both < Target |

### Component: Neural Network Plots

Developer and Reviewer: Nicholas

|  |  |  |
| --- | --- | --- |
| Type | Pass | Comments |
| Plots are generated for each model | Yes | Plots are generated and show patterns consistent with performance metrics. Models with better performance metrics have curves closer to actual data. |
| Analysis of plots is conducted | Yes | All models tend to under predict between 1980 to 2000, then overpredict thereafter. |

### Component: Non-Functional Requirements

Developer and Reviewer: Nicholas

|  |  |  |
| --- | --- | --- |
| Type | Pass | Comments |
| Written analysis is correct and thorough | Yes | Some mistakes were made and corrected. Also switched from Residual Sum of Squares to Sum of Residuals, which is the correct metric. |
| Research questions are answered | Yes | Models replicated, regression assumptions are checked, and model performance differences with additional features analyzed |
| Applications use accessible options | Yes | Colors include dodgerblue, darkorange, and purple. Cividis color map is used for colorblind users. Line styles are used in plots. |
| Applications are organized | Yes | Headers are used in the Jupyter Notebook for table of content accessibility and outputs are organized with tabs and expanders in Streamlit |
| Results are reproducible | Yes | Metrics are the same when running either the Notebook or the Streamlit Application. They are the same when rerun. |

# Systems Testing

Both the Jupyter Notebook and Streamlit Web Application runs successfully from start to finish like a cookbook.

In the Jupyter Notebook, the ‘Run All’ button runs all the code, generates all outputs, and does not raise any errors (‘ignore warning’ is not used). As this takes a top-down approach, the pipeline must function correctly, and each step must be carried out in the intended order to get this result. The data is wrangled successfully, tables of the data and summary statistics are generated, the data and distributions are plotted, a correlation heatmap is created, all models are fitted, all metrics are computed, and all visualizations of predictions are generated without issues.

The same goes for the Streamlit application. The model successfully runs from start to finish without errors. Additionally, users can change the models to plot with needing to retrain any of the models since session states are saved as intended. All output is generated, and the user has access to Streamlit specific user interactions such as expanding and collapsing boxes with outputs, selecting which model to plot, navigating across different sections by clicking on tabs, and expanding table column width or sorting tables.

All written analyses, which answer our research questions, are included in both the Jupyter Notebook and the Streamlit Web Application as markdown outputs.

# User Guide

**Version:** 4.0

**Date:** April 30, 2025

## Copyright

© 2025 Nicholas Pham All rights reserved.

No part of this guide may be reproduced without permission.

## Legal Notice

**Disclaimer:** This application uses data from the Federal Reserve Economic Database (FRED). If you use this application, please refer to FRED’s terms of use [here](https://www.stlouisfed.org/about-us/legal-information).

## Preface

**Purpose of this Guide:** To help users understand and effectively use the Federal Funds Rate Regression Modeling Notebook

**Intended Audience:** Non-technical users, first-time users with background knowledge of statistics and interest in economics or finance.

**System Requirements:**

* Computer or laptop with internet browser
* HTML required
* Internet access required for using the Streamlit Application
* Requirements for running the application locally with Jupyter Notebook:
  + Python version 3.9.7
  + Packages: Pandas, Matplotlib, Numpy, Seaborn, Statsmodels, Scikit-Learn, Scipy, and Tensorflow

## General Information

**Overview:** The application will attempt to replicate and validate some models, visualizations, and metrics of a researcher, Alper D. Karakas (2023) in the paper titled “Reevaluating the Taylor Rule with Machine Learning.” It will compute performance metrics, construct visualizations, and find the p-values of test statistics used to check model assumptions. The application will then construct neural network models for each set of independent variables and compute performance metrics.

**Target Audience:** Anyone interested in validating the work of Karakas, in predicting the Federal Funds Rate, or is learning regression modeling and neural network modeling. Requires some background knowledge of statistics.

**Key Benefits:** Users can see the replication and validation of work in literature. Those newer to Python and data science can learn how to construct some load data from URL links, construct OLS regression models, compute regression performance metrics, verify model assumptions, and construct neural network models. Performance differences can be observed between the regression models and neural network models.

## System Summary

## System Architecture

**Main Components:**

* Data Retrieval Layer – GitHub Raw Data
  + Fetches CSV from GitHub raw URLs using pandas.read\_csv().
  + Loads multiple datasets as Pandas DataFrames.
* Data Processing Layer – Feature Engineering & Merging
  + Cleans and preprocesses the data (handling missing values and scaling)
  + Merges datasets into a unified DataFrame for model training.
  + Derives new features (such as lagged features and percentage versions of features).
* Exploratory Data Analysis Layer
  + Shows first and last 5 rows of the data table
  + Constructs a summary statistics table
  + Constructs line plots showing how each variable changes over time
  + Constructs distribution plots of the data
  + Constructs a correlation heatmap
* Modeling Layer
  + Selects relevant columns as model inputs (features).
  + Fits the models with Ordinary Least Squares (OLS) Regression
* Regression Assumptions Layer
  + Computes test statistics and p-values for the following assumptions:
  + Normality (Jarque-Bera test statistic)
  + Homoscedasticity (Breusch-Pagan test statistic)
  + Autocorrelation (Durbin-Watson test statistic)
  + Linearity (Rainbow test statistic)
  + Multicollinearity (Variance Inflation Factors)
* Metrics and Model Comparison Layer
  + Computes model statistics (AIC, BIC, f-stat, p-values)
  + Computes performance metrics (MSE, RMSE, R², etc.).
  + Creates plots of predictions.
* Neural Network Layer
  + Fits models with a Neural Network
  + Computes performance metrics (MSE, RMSE, R², etc.).
  + Creates plots of predictions.
* **Data Flow:**

1. The user opens the application and runs the Jupyter Notebook.
2. Raw data is fetched from GitHub URLs using pandas.read\_csv(url)
3. Duplicates rows are checked for in each dataset.
4. Data is processed and combined into a single dataframe, (missing values are handled, new features are derived, and duplicates are checked for again).
5. Data is used to conduct Exploratory Data Analysis.
6. Data is scaled then used to fit the OLS regression models.
7. Assumption test-statistics and p-values are computed, performance metrics & visualizations are generated from the model predictions and residuals.
8. Data is reused to fit neural network models, generating additional metrics.

**Getting Started**

**System Requirements**

* **Operating Systems Supported:** Windows, macOS, Linux, etc.
* **Jupyter Notebook**
* **Python (**version 3.9.7 or later)
* **Python Packages:** Pandas, Matplotlib, Numpy, Seaborn, Statsmodels, Scikit-Learn, Scipy, and Tensorflow

## Using the System

### System Requirements

* Computer or laptop with internet browser
* HTML required
* Internet access required for using the Streamlit Application
* Requirements for running the application locally with Jupyter Notebook:
  + Python version 3.9.7
  + Packages: Matplotlib, Numpy, Pandas, Plotly, Scikit-Learn, Seaborn, Statsmodels, and Tensorflow

### Usage Instructions

**Jupyter Notebook:**

1. Download Python and necessary packages (Matplotlib, Numpy, Pandas, Plotly, Scikit-Learn, Seaborn, Statsmodels, and Tensorflow)
2. Refer to Jupyter Notebook installation instructions at: <https://jupyter.org/install> or install the Anaconda distribution at: <https://www.anaconda.com/download> (you can click skip registration to download without signing up)
3. Download the .ipynb file and save it on your machine
4. Open Jupyter Notebook, locate and open the file FFR\_regression\_modeling.ipynb
5. From the toolbar at the top, click on “Run,” then click on “Run all cells.”

A screenshot of a computer

AI-generated content may be incorrect.

**Streamlit:**

1. Open a web browser on a device with internet access
2. Enter the following URL in the web browser:

* <https://ffr-modeling-5n85wvsg3gdnvd4z4yz3kh.streamlit.app/>

1. Wait a minute for the application to finish running

### Key Features

**Jupyter Notebook:**

* **Code Cells:** All Python code is visible in code cells. Their outputs, once run, are immediately below.
* **Markdown Cells:** These cells enable text input instead of code. They are used mainly for providing information and interpretation of code outputs and results.
* **Section Headers:** Sections in the application are listed clearly, enabling the user to use the search function of their browser (Control + F) to quickly navigate between sections of the notebook.
* **Table of Contents:** Enables the user to view all sections in the Notebook
* **Reference List:** The user can find the links to all datasets used in the development of this project and application.

**Streamlit:**

* **Tabs:** Content is organized into different sections. Each section is navigable via a tab bar. Users can click on the section they want to view. The tab bar highlights which section is currently being viewed and can be scrolled using Shift + Mouse Scroll or swiping if on a mobile device with a touchscreen.
* **Expandable Boxes:** Boxes can be clicked on or tapped to expand and show hidden content. They can collapse to hide content.
* **Model Selector for Plotting:** Plots can be viewed for each model by clicking on or tapping a selection box and choosing the model to plot. Due to special reasons, the Taylor Prediction plot will display unique markdown text below it if it is selected. These are notes on comparison with Karakas’ work.
* **Interactive tables:** Tables in the application can have their column width extended to view the column names if they are truncated. Users can sort the tables by clicking on any column name.

### Accessibility Features

* **For Low Vision Users:**
  + Dark Mode option in settings

In Jupyter Notebook:

A screenshot of a computer

AI-generated content may be incorrect.

In Streamlit:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* **For Color-Blind Users:** Problematic color combinations are avoided (dodgerblue, darkorange, and purple are used in plots), cividis colormap is used, and line styling is used.

### Safety Information

For security, make sure only to download files from links provided in the documentation. Please avoid downloading and installing files from unknown sources.

## Troubleshooting

**Jupyter Notebook Common Issues:**

**Issue:** ImportError (Missing Attribute or Function), ImportError (Module Not Found), or TypeError (Different Function Signature)

* **Solution:** Check that the required libraries are installed.

**Issue:** ImportError (Module Deprecated)

* **Solution:** Check to see if outputs are generated and the rest of the code in the Notebook is run. If they are, this simply means that some coding methods used were out of date and use new functions or methods. Please notify the developer if this is the case.

**Streamlit Common Issues:**

**Issue:** The application suddenly has rendering issues when scrolling.

* **Solution:** Try clicking somewhere on the application, then using hitting keyboard shortcut Ctrl + A if on a computer or laptop. For mobile users, try selecting some text then tapping on ‘Select All.’

**Issue:** Widget Interaction Issues

* **Solution:** Check that Java Script is enabled within your browser.

**Issue:** The application looks broken when first starting it up.

* **Solution:** This may be due to browser compatibility. Try using a different web browser and see if the issue has been resolved. Please notify the developer if this is the case.

## FAQ (Frequently Asked Questions)

**Question 1:** How do I expand the output cells? Some cells have long outputs, and you can scroll through them, but I want to expand them and see them.

* **Answer:** By hovering your cursor over the left side of the output cell, you can see an icon. Clicking the bar with that icon will expand it. It can collapse by clicking again.

A screen shot of a computer

AI-generated content may be incorrect.

**Question 2:** I clicked on a markdown cell, and it opened some text editor, making the text smaller. How do I fix this?

* **Answer:** Click on the markdown cell and check for the blue light on the side to ensure the cell is selected. In the toolbar on the menu there is a play/pause icon. Clicking on that will run the cell. Alternatively, hitting Ctrl + Enter will do the same thing. This should exit the markdown editing.

**Question 3:** The Notebook is not rendering. How can I fix this?

* **Answer:** Check to see if your web browser has HTML features and that they are enabled.

**Question 4:** I ran some cells out of order and want to refresh the numbers on the left of each cell. How do I reset them?

* **Answer:** You can click on the “Kernel” option from the top toolbar and click on one of the “Restart Kernel” options.

**Question 5:** Can I use your code and edit it?

* **Answer:** Sure. Have fun.

**Question 6:** I have made some changes and improvements to your code and want you to look at it, and merge the changes if you find them acceptable.

* **Answer:** Submit a pull request and I’ll take a look.

**Question 7:** Is there a widescreen option in Streamlit?

* **Answer:** Yes, click on the 3 lines in the corner to view options, click on settings, then select “wide mode.”

**A black background with white text

AI-generated content may be incorrect.**

## Help and Contact Details

For assistance, reach out to the developer:

* **Support Email:** nickpham12@gmail.com
* **Phone Support:** 510-825-440

## Glossary

* **Breusch-Pagan test:** A test used to check for whether the regression assumption of homoscedasticity of residuals has been met.
* **Durbin-Watson test:** A test used to check for whether the regression assumption of autocorrelation of residuals has been met.
* **Federal Funds Rate:** The interest rate at which banks charge when they lend to each other overnight. This is determined by the market, like stocks prices.
* **Federal Funds Target:** This is a rate set by the Federal Reserve. The Federal Reserve aims to adjust the Federal Funds Effective Rate to this through some financial operations. They do this to manage economic conditions and inflation.
* **GDP:** Gross Domestic Product. The value of all goods and services produced by a country in a year. Split into Real and Potential GDP. Real GDP is the actual GDP of a country, while Potential GDP is the GDP a country can achieve in theory when going all out in production
* **Jarque-Bera test:** A test used to check for whether the regression assumption of normality of residuals has been met. Works for larger datasets than Shapiro-Wilk test.
* **Inflation:** The increase in prices over time. A gallon milk may cost 3 dollars one year, and 4 dollars in the future. The difference is due to inflation. Inflation changes the purchasing power of currency.
* **Inflation Gap:** The difference between Inflation Rate and Inflation Target
* **MAE:** Mean Absolute Error.
* **MAPE:** Mean Absolute Percentage Error.
* **MPE:** Mean Percentage Error.
* **MSE:** Mean Squared Error.
* **Multicollinearity:** The degree to which independent variables are correlated to each other.
* **Output Gap:** The difference between Real and Potential GDP
* **P-value:** The probability value of a test-statistic. This is primarily used for quickly performing hypothesis testing
* **Rainbow test:** A test used to check for whether the regression assumption linearity has been met.
* **RMSE:** Root Mean Squared Error.
* **Taylor’s Rule:** An economic policy guideline used to set the federal funds rate. It uses Inflation, Inflation Target, Real GDP, and Potential GDP as estimators. We use it as a predictor in some of our models.
* **Variance Inflation Factor (VIF):** A measure of the degree of multicollinearity among variables.

# Systems Administration Guide

**Version:** 4.0

**Date:** April 30, 2025

## Copyright

© 2025 Nicholas Pham All rights reserved.

No part of this guide may be reproduced without permission.

## Legal Notice

**Disclaimer:** This application uses data from the Federal Reserve Economic Database (FRED). If you use this application, please refer to FRED’s terms of use [here](https://www.stlouisfed.org/about-us/legal-information).

## System Overview

**Function:** This application(s) explores the data, constructs OLS and Neural Network models for predicting the Federal Funds Rate, visualizes results, and prints analyses as markdown.

**Components:** Jupyter Notebook, Python, libraries (Numpy, Pandas, Scipy, Scikit-Learn, Seaborn, Statsmodels, Streamlit, and Tensorflow)

**Data:** The data was downloaded from the Federal Reserve Economic Data (FRED) database and uploaded to a Github repository. The application fetches the data from the repository.

**Architecture:** Standalone Jupyter Environment, possibly running on local machine, virtual environment, or server. Streamlit Web Application hosted on Streamlit Community Cloud.

## System Configuration

**System Requirements:**

* HTML required
* Computer or laptop with internet access and web browser
* Operating Systems Supported: Windows, macOS, Linux, etc.
* Jupyter Notebook
  + Refer to installation instructions at <https://jupyter.org/install> (Project Jupyter, n.d.)
  + Download as part of the Anaconda distribution at <https://www.anaconda.com/download> (Anaconda, n.d.).
  + If installing Anaconda, click on skip registering to download without signing up. Refer to figure 1.
* Python version 3.9.7
  + Install at <https://www.python.org/downloads/> (Python Software Foundation, n.d.). Refer to figure 2.
  + Or download as part of the Anaconda distribution at <https://www.anaconda.com/download> (Anaconda, n.d.).
* Python Packages: Pandas, Matplotlib, Numpy, Seaborn, Statsmodels, Scikit-Learn, Scipy, Streamlit, and Tensorflow
  + Altair below version 5 is required for Streamlit compatibility
  + List of specific package versions used:
    - altair==4.2.2
    - ipython==8.18.1
    - keras==3.9.2
    - matplotlib==3.9.4
    - notebook==7.4.1
    - numpy==2.0.2
    - pandas==2.2.3
    - pyarrow==20.0.0
    - scikit-learn==1.6.1
    - scipy==1.13.1
    - seaborn==0.13.2
    - statsmodels==0.14.4
    - streamlit==1.12.0
    - tensorboard==2.19.0
    - tensorflow==2.19.0
* Pip for installing packages if needed
  + Pip is automatically installed if Python was downloaded from <https://www.python.org/downloads/>
  + Installation instructions available at: <https://pip.pypa.io/en/stable/installation/>

## System Maintenance

All necessary files are to be uploaded to the Github Repository at: <https://github.com/nvpham12/Capstone-Project>.

Github allows for version control and file backup, enabling the upload of newer versions of the Notebook files and Streamlit application source code while retaining the older versions as downloadable backups. Changes are documented when made and pushed as commits.

After making changes to the code, restart the kernel and clear all outputs, then run the entire notebook. Refer to figure 3. Then check to see that all outputs are given. This checks that the entire pipeline functions in order without errors.

Ensure that the libraries and packages are up to date. One way to do this is using pip-review. After installing it, it can be used in the command line to quickly check all installed packages, their versions, and whether there is a newer version.

For Streamlit Application development, Altair below version 5 is required.

If there are still issues with libraries and package dependencies, refer to requirements.txt for specific library versions or the list in System Configuration.

## Security Related Processes

This project uses a locally run Jupyter Notebook. To minimize security vulnerabilities, please make sure to only use files downloaded from the official GitHub Repository: (<https://github.com/nvpham12/Capstone-Project>).

Regularly update software and packages to ensure the latest security updates are available. ‘pip-review’ can be used to check the versions of all libraries, while ‘pip-review --auto’ can be used to update all of them at once. However, be mindful of package compatibility. This can be checked using ‘pip-check.’

Clear all outputs before uploading or sharing the Jupyter Notebook and avoid use of absolute paths to avoid disclosing personal system paths, environment variables, or other user specific information.

Avoid running any code with administrator privileges.

The datasets used are all listed in the appendix, with the link to where the data was obtained and including the data sources.

## Table of Figures

Figure 1:

A screenshot of a email

AI-generated content may be incorrect.

Figure 2:

A screenshot of a computer

AI-generated content may be incorrect.

Figure 3:A screenshot of a computer

AI-generated content may be incorrect.

# Capstone Project Presentation Link

https://youtu.be/TryTsQNlqis

# Appendix

Board of Governors of the Federal Reserve System (US). (2025). *Federal Funds Effective Rate [DFF]*. Federal Reserve Bank of St. Louis. Retrieved from https://fred.stlouisfed.org/series/DFF

Board of Governors of the Federal Reserve System (US). (2025). *Federal Funds Target Rate (DISCONTINUED) [DFEDTAR]*. Federal Reserve Bank of St. Louis. Retrieved from https://fred.stlouisfed.org/series/DFEDTAR

Board of Governors of the Federal Reserve System (US). (2025). *Federal Funds Target Range - Lower Limit [DFEDTARL]*. Federal Reserve Bank of St. Louis. Retrieved from https://fred.stlouisfed.org/series/DFEDTARL

Board of Governors of the Federal Reserve System (US). (2025). *Federal Funds Target Range - Upper Limit [DFEDTARU]*. Retrieved from Federal Reserve Bank of St. Louis. https://fred.stlouisfed.org/series/DFEDTARU

U.S. Bureau of Economic Analysis. (2025). *Real Gross Domestic Product [GDPC1]*. Federal Reserve Bank of St. Louis. Retrieved from https://fred.stlouisfed.org/series/GDPC1

U.S. Bureau of Labor Statistics. (2025). *Consumer Price Index for all urban consumers: All items in U.S. city average [CPIAUCSL]*. Federal Reserve Bank of St. Louis. Retrieved from https://fred.stlouisfed.org/series/CPIAUCSL

U.S. Bureau of Labor Statistics. (2025). *Unemployment Rate [UNRATE]*. Federal Reserve Bank of St. Louis. Retrieved from https://fred.stlouisfed.org/series/UNRATE

U.S. Congressional Budget Office. (2025). *Real Potential Gross Domestic Product [GDPPOT]*. Federal Reserve Bank of St. Louis. Retrieved from https://fred.stlouisfed.org/series/GDPPOT

# References

Anaconda, Inc. (n.d.). *Download Anaconda Distribution*. Anaconda. <https://www.anaconda.com/download>

Federal Reserve. (2024, January 22). *Federal Reserve Board - Reserve Requirements*. Board of Governors of the Federal Reserve System. https://www.federalreserve.gov/monetarypolicy/reservereq.htm

*Inflation (PCE)*. (2024). Board of Governors of the Federal Reserve System. https://www.federalreserve.gov/economy-at-a-glance-inflation-pce.htm

Karakas, A. D. (2023). *Reevaluating the Taylor Rule with Machine Learning*. ArXiv.org. <https://arxiv.org/abs/2302.08323>

Project Jupyter. (n.d.). *Installation*. Jupyter. <https://jupyter.org/install>

Python Packaging Authority. (n.d.). *Installation*. pip. <https://pip.pypa.io/en/stable/installation/>

Python Software Foundation. (n.d.). *Download Python*. Python.org. <https://www.python.org/downloads/>