COMS 4995 - Fall 2023
Project Group 23 - David Davila-Garcia, Rain Wei, Yuyi Tang, Shrinjay Kaushik, Ziyao Tang

I. Introduction - David

Time-series forecasting holds significant relevance across various fields, including finance, healthcare, and politics. Accurate and robust time-series algorithms, which hold minimal assumptions about the underlying data, show promise for high-impact applications. Following the related work by Chatterjee et al. [1], the goal of our project is to train and evaluate the performance between classic regression-based (OLS, Ridge, Lasso), econometric-based (ARIMA), and deep learning-based (LSTM) models in forecasting the **US Dollar return on fixed receiver position (DU05YXR_VT10)** macroeconomic indicator.

II. Data Description

All models were trained on daily US Dollar macroeconomic time-series data, collected between January 2000 to August 2023, using the JPMaQS Quantamental Indicators dataset [2]. This dataset contains values for 20 distinct macroeconomic indicators over 6155 distinct days. The forecasting target variable, **DU05YXR_VT10**, represents the estimated returns for a given investment, assuming risk equals 10% [3].

Given the high-quality data curation process outlined in [4], it was unreasonable to impute missing values from the curated dataset - all dates (rows) containing missing values were dropped (19 rows, 0.3% of records). Since each record in a time series dataset is dependent on each other, imputing the missing values by column means or medians disregards the intrinsic temporal correlations and patterns. As such, these imputation approaches are likely to introduce unnecessary noise and increase model bias.

We used a 90-10 temporal split to generate the development and test sets. We used temporal splits on the development set to tune the optimal hyperparameters. We refit the models on the entire development set with the optimal hyperparameters from validation and reported MAE, MSE and R^2 on the unseen test set.

III. Methodology

Our ML models use prediction loss minimization and mean squared error (MSE) as the objective function. Since the target variable is continuous, we set **ordinary least squares (OLS)** as a baseline model. This method will create a linear model that uses the linear combination of macroeconomic factors on a given day or over a rolling window to predict the return on fixed receiver position. We trained Ridge and Lasso Regression models to avoid overfitting the training data and generalize performance on the unseen test set. Moreover, we implemented **ARIMA models**, such as autoregressive models and moving-average models, to capture the autocorrelation in data. Lastly, we used a **neural network** LSTM model which we hypothesized would be able to capture the underlying patterns from our data better than our baseline models.

Baseline OLS, Ridge, and Lasso models:

The Ordinary Least Squares (OLS) method, Ridge Regression model, and lasso model are linear models that are used for predictive modeling, especially when we expect to have multicollinearity in our dataset. The OLS model primarily focuses on minimizing the residuals for the best-fit line whereas the Ridge Regression model introduces a regularization parameter that can control the complexity of the

model. This parameter is denoted by the symbol 'α'. By controlling the value of the alpha parameter we can better generalize the model on our dataset. The lasso model involves adding a penalty term to constrain the complexity of the model. This penalty term is the sum of the absolute values of the model coefficients, known as L1 regularization.

Before using the model it is essential to split the dataset using the time-series split and preprocess the data to ensure that the model is learning new features from the parameters provided. In the case of the OLS model, it is imperative to ensure that the data has linearity, independence, and homoscedasticity. Whereas, for Ridge Regression the only parameter to be controlled is the value of alpha, which is the same for the Lasso model. For evaluation of the trained models, MSE(Mean Squared Error), MAE(Mean Absolute Error), and Average R-squared error metrics have been used to ensure that the trained models are not overfitting, which is a common problem when dealing with datasets consisting of a large number of independent variables.

ARIMA model :

The ARIMA (Autoregressive Integrated Moving Average) model is a method for time-series data forecast, combining the autoregressive model and moving average model. The model regresses the time difference in the response variable (DU05YXR_VT10) on its prior time and prior prediction errors. When fitting the model with the development set, we aim to learn three parameters - the number of autoregressive terms (p), the number of time differences (d), and the number of lagged forecast errors (q). These parameters control how prior information affects the current prediction. We need to note that for the ARIMA model, we do not use features as predictors, which is different from all other models we used.

Before using the model, I ran model diagnostics to ensure that statistical assumptions are met with the dataset. As shown in the appendix, diagnostics include ensuring that residuals are normally distributed and independent from the response variables. With different time series split, I fitted the autoregressive model to learn the sets of parameters. I then tested models by making forecasts on unseen future dates to obtain mean squared error, mean absolute error.

LSTM model:

The LSTM model is built on top of the recurrent neural network model with memory, input, and forget gates, which allows it to model long-distance dependencies better. The LSTM model is beneficial for predicting time series data since it can remember longer pieces of information. In my implementation, I built a neural network with 2 LSTM layers and 1 dense layer1: with the first layer encoding the data into sequences so it can be stacked on top of the second layer, and the second layer outputting the actual vector concerning the input data. The last layer is used to build a regressor for predicting the final result. To prevent overfitting, I added dropout layers and batch normalizations after every LSTM layer. Dropout layers randomly deactivate some neurons to help with generalizations and overfitting while batch normalization stabilizes the input fed into the later layers. The early stop parameter is turned on for this network so it will not overfit the training noises once the validation loss is no longer decreasing. The LSTM model takes all of the macroeconomic indicators as input and predicts the values of the DU05YXR_VT10 variable.

For the hyperparameter tuning process, I applied a keras tuner using random search to find the best hyperparameter combination of several units and model optimizers, for the validation loss. Concerning each choice of optimizer, I use a different learning rate as a single learning rate can cause oscillations or even divergences during gradient backpropagation for different optimizers. Lastly, I

retrained the best model in every fold on the train data and evaluated the test set to get the mean squared error, mean absolute error, and R^2 score.

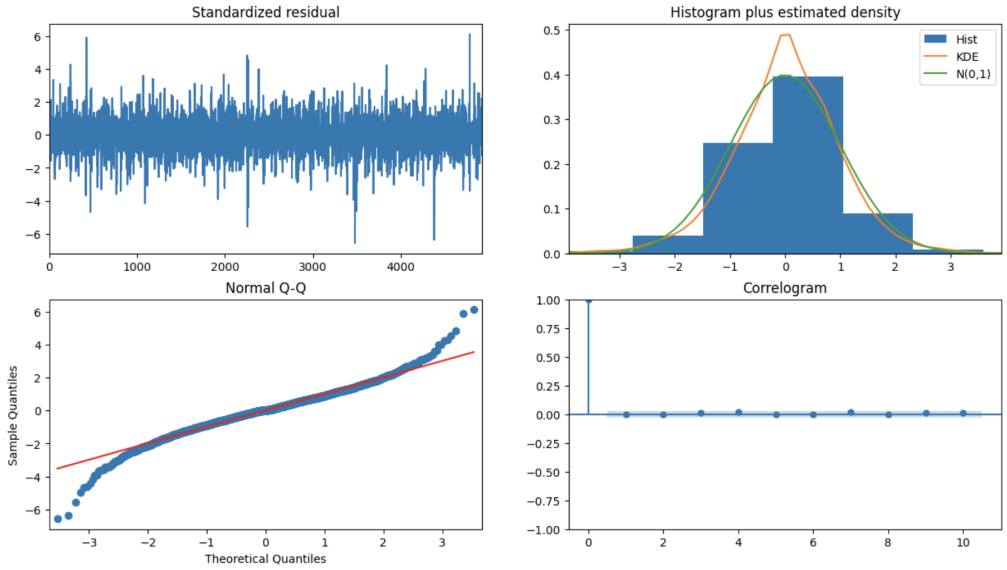
IV. Model Results

Table 1. Model Results on the Test Set (10%)			
Model	**MSE**	**MAE**	**R^2**
OLS	0.02	0.11	0.95
Ridge (alpha = 0.01)	0.02	0.10	0.96
Lasso (alpha = 0.0001)	0.02	0.10	0.96
ARIMA	0.56	0.75	N/A
LSTM	0.04	0.13	0.92

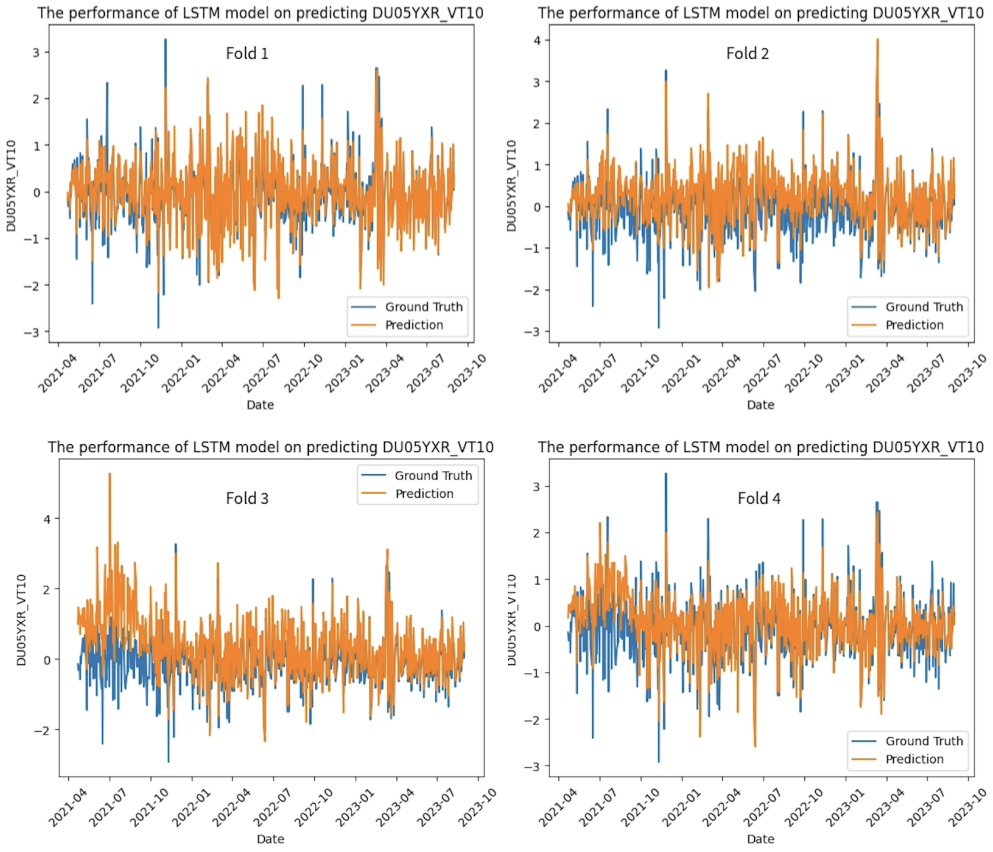
V. Conclusion

The results from Table 1 above show that the Baseline OLS, Ridge, and Lasso Regression Models outperformed the ARIMA and LSTM models. These results can be explained by the complexity of each model and their ability to overfit on the training set. The simplistic linear models already do a fairly good job at capturing the patterns from the data as shown by the low MSE, MAE, and R^2. The ARIMA model only uses the previous values from the target variable as the input feature, so it is not fair to compare it against the linear models or LSTM. Due to the different input for the ARIMA model, its performance was worse compared to the other models. Moreover, the LSTM had similar performance to the baseline linear models, but the hyperparameters found during random search did not yield better results on the test set as we had hypothesized. The LSTM's performance might be negatively impacted by the smaller size of the dataset, given that it is only about 6000 records.

ARIMA model visualizations:



LSTM results visualizations:



VI. References

[1] Chatterjee, Ananda, Hrisav Bhowmick, and Jaydip Sen. "Stock Price Prediction Using Time Series, Econometric, Machine Learning, and Deep Learning Models." In *2021 IEEE Mysore Sub Section International Conference (MysuruCon)*, 289–96. Hassan, India: IEEE, 2021. https://doi.org/10.1109/MysuruCon52639.2021.9641610.

[2] "JPMaQS Quantamental Indicators." Accessed December 8, 2023. https://www.kaggle.com/datasets/macrosynergy/fixed-income-returns-and-macro-trends.

[3] Macrosynergy Academy. "Macrosynergy Academy." Accessed December 9, 2023. https://academy.macrosynergy.com/academy/Themes/Generic%20returns/_build/html/notebooks/Duration%20returns.php.

[4] Macrosynergy Academy. "Understanding Quantamental Indicators." Accessed December 9, 2023. https://academy.macrosynergy.com/quantamental-understanding/.