Julian Wiley

CSCI E-86 Deep Learning

12/13/22

Final Project – Case Study in Economics – Predicting Economic Recessions using Macroeconomic Indicators and Gold Prices

# Abstract

Predicting economic recessions is one of the most important applications of machine learning for economics. Recessions are major events that have significant implications for all manner of economic policies and business decisions. Currently, making predictions about future economic conditions is done by economists using traditional statistical models. This approach is slow and subjective, as experts can disagree until well into the start of a recession. This project aims to use machine learning models to predict future economic conditions quickly so that policymakers and corporate actors can begin making plans to mitigate the consequences of these shifts as early as possible.

The project lifecycle is relatively straight forward, with the workflow closely adhering to the basic machine learning project skeleton. We begin by downloading each indicator separately using the Fred API and yfinance libraries. We then explore each variable before joining them into a single data frame. After the variables are joined, we create the univariate and multivariate feature and label sets, reshaping the dataset into the format required by RNNs. Both datasets are shaped such that the models will be making predictions based on the previous 60 days of data. After, we build and train the univariate and multivariate LSTM, GRU, autoencoder, and CNN models. Our results show that the univariate LSTM has the best performance, with its predictions most closely resembling the test set.

Given the structure of our dataset, we chose deep learning models that perform well in time series forecasting. We implement each model on both univariate and multivariate datasets, first using just the probability that we are in a recession and then using the full dataset. The first type of model that we implement is an RNN, building several different LSTM variations. Secondly, we built a variational autoencoder (VAE) on both datasets that performed poorly. Lastly (too chunky), we built out convolutional neural networks (CNNs), which also performed quite poorly.

* Two minute (short): https://youtu.be/50ws4F889gU
* 15 minutes (long): https://youtu.be/-9HwrHZVedg

# Setup

To standardize our setup and ensure there are no package conflicts, we used anaconda and a python 3.8 virtual environment to ensure only the packages we needed would be installed. For our main libraries, we used NumPy, pandas, TensorFlow, and plotly. As previously mentioned, we used Fred Api and Yfinance to download data which required little consideration for package dependencies. For TensorFlow, in order to ensure response times, we used the standard installation to compile it with support for GPU acceleration. We also used Pyarrow, a python binding for the arrow tool, to manipulate parquet files which enabled us to achieve a relatively small file size for our data set with fast loading times and quick computation. Everything was installed using pip install with pypi wheels using standard installation procedures; versions were not specified for the installation.

# Dataset

Our dataset consisted of a combination of macroeconomics indicators and gold commodities prices, obtained through the aforementioned software tools and libraries. It was cleaned with basic preprocessing steps. Gold prices came in standard OHLCV (open/high/low/close/volume) format and was of daily frequency. The following macroeconomic indicators were used: S&P500 for baseline performance , Consumer Price Index, Personal Consumption Expenditures, Unemployment Rate, GDP , Inflation, M2 Money Stock, Federal Funds Rate, Market Yield on U.S. Treasury Securities at 10-Year Constant Maturity Quoted on an Investment Basis, Smoothed US Recession Probabilities, NBER Recession, and Total nonfarm employment.

# Models

The first models we used were LSTM’s that predicted the probability of a recession at any given time based on the last 60 days of data. As with every model, we trained univariate and multivariate versions. These all shared the same architectures, the only difference being the input shapes. They all had 5 layers with 4 LSTM layers and 1 dense layer, each with a dropout after it with a rate of 0.5. Surprisingly, the univariate LSTMs performed the best with the predicted recession probabilities very closely matching the actual probability when predictions were modeled on the test sets. This is antithetical to the results as judged by the root mean squared error in the training epochs, with the univariate LSTM having a RMSE of 0.0009809 and the multivariate LSTM having a RMSE of 0.00079603. Given the visible differences between the closeness of predictions on the test set, with the univariate more closely fitting to the actual predictions.

Next, we built GRUs, which are another form of time series specific models that have been shown to do well in forecasting. Again, we built out univariate and multivariate architectures, each with 5 layers. We added 4 GRU layers, each with 50 units, followed by a dropout layer for each. They performed quite well, but still failed to model the test set as accurately as the LSTMs.

The next model used variational autoencoders to try to map the inputs to our outputs with some lower dimensionality. These we tried to use because variational autoencoders have been shown to do well at pre4dicitng anomalies in time series. Given the relative sparsity of events in this data (there were only 2 recessions in the 15-year time span) this architecture was shown in an attempt to model it. The variational encoder consisted of 3 LSTM layers with 64 32 and 16 nodes, respectively. While the variational decoder also had 3 layers with 16 32 and 64 nodes, respectively. These models both performed very poorly with losses showing them to be very poor predictors. The univariate model had an RMSE of 0.0756 while the multivariate model had an RMSE of 0.00053676.

Lastly, we used CNN’s to predict our time series in an attempt to assess the performance of our other models against an architecture style that is not well suited for time series. Our CNN’s had different architectures and both required extensive adjustments to ensure the networks worked with our time series. The univariate model had 4 layers, a Conv1D layer with 20 nodes, kernel size of 4, and stride length of 4, 2 GRUs with 20 nodes each, a time-distributed dense output layer. The multivariate model was considerably deeper, having 10 layers. The first layer was a Conv1D layer with 128 nodes and a kernel size of 12, followed by a MaxPooling1D layer with a pool size of 2. Next Conv1D layer has 64 units and a kernel size of 3, followed by another MaxPooling1D layer with a pool size of 2. Last convolutional layer had 32 units and kernel size of 3 followed by the last MaxPooling1D layer that also has a pool size of 2. Next there are 2 GRUs with 16 nodes each, followed by a flatted layer and dropout at 0.1. Lastly the output layer is a one unit dense layer. The univariate model had an RMSE of 0.0758 while the multivariate model’s was 0.00025634.

# Visualizations

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedChart, line chart

Description automatically generatedChart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Chart, histogram

Description automatically generatedChart

Description automatically generatedChart, histogram

Description automatically generatedA picture containing icon

Description automatically generated

Figure - Univariate LSTM Loss

Shape

Description automatically generated with low confidence

Figure - Univariate LSTM Predictions

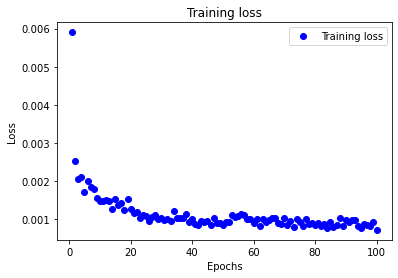


Figure - Univariate GRU Losses

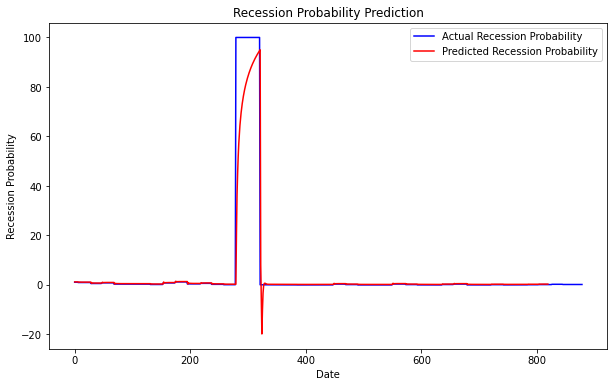


Figure - Univariate GRU Predictions

Text

Description automatically generated with low confidence

Figure - Multivariate LSTM Losses

Histogram

Description automatically generated with low confidence

Figure - Multivariate LSTM Predictions

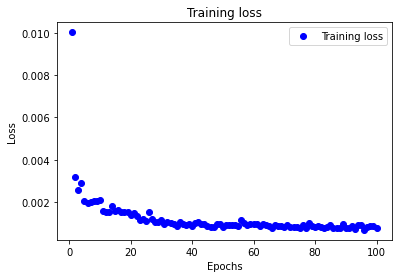


Figure - Multivariate GRU Losses

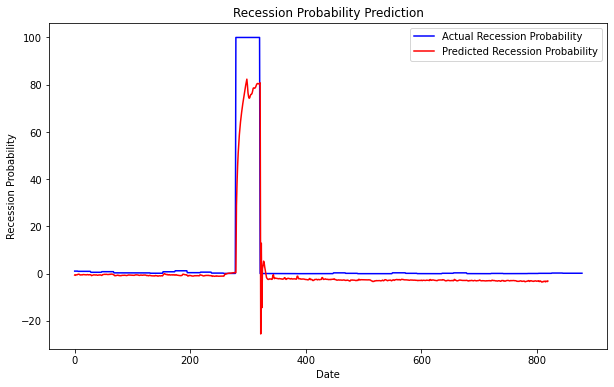


Figure - Multivariate GRU Predictions

A picture containing chart

Description automatically generated

Figure - Univariate CNN Losses

A picture containing icon

Description automatically generated

Figure - Multivariate CNN Losses

A picture containing text

Description automatically generated

Figure - Multivariate CNN Predictions

# Conclusions

To conclude, the models that per5formed the best were times series specific architectures that previously shown good performance in similar tasks. LSTMs performed the best, which was unsurprising, h however the fact that the univariate outperformed the multivariate was surprising. We learned that using regressions to predict the probability of a recession is likely not the best way to run these predictions. Other works have shown that classifiers using binary outcomes have shown good performance, better than the methods tested here (citation needed). In the future, I would have liked to more extensively test different model architect5ues, as well as more complex modeling of macroeconomic activity using more targeted variables.