Birkbeck, University of London

Department of Computer Science & Information Systems

MSc Data Science Project Report

Nowcasting US Industrial Production Using Machine Learning

September 2021

Author: Peter Colson

Supervisor: Prof George Magoulas

Abstract

Nowcasting seeks to forecast the current point in time by leveraging higher frequency data to predict series that may lag several months in publication. This project aims to identify a machine learning model that can produce accurate nowcasts of US industrial production, driven by the experience of first-hand demand for timely forecasts at the onset of the coronavirus pandemic. A high frequency dataset is constructed and then used to develop several machine learning models including regularised linear regression, artificial neural networks, and long short-term memory networks. Each of these models are found to outperform an autoregressive specification of industrial production, with the optimal model in terms of minimal root mean squared error found to be an artificial neural network trained using just two interest rate features.

Table of Contents

1 Introduction

This project seeks to identify a machine learning model that can provide accurate nowcasts of US industrial production. The model should also have a relatively short runtime to enable daily updates of the nowcast.

A nowcast seeks to forecast the current point in time and does so by making use of higher frequency data that can bridge the information gap caused by economic series, such as industrial production, being published on a lag of several months.

Nowcasts are required by policymakers to make timely, informed, and effective policy decisions, with the events of the last eighteen months following the coronavirus pandemic highlighting the importance of this. Indeed, whilst working as an economist during the initial months of the pandemic, I have seen the demand for timely economic information first-hand, with industrial production being particularly important in helping businesses avoid supply chain disruptions.

Various machine learning models are trialled, including regularised linear regression, artificial neural networks, and long short-term memory networks. Additionally, several feature selection techniques are tested to find the optimal subset of higher frequency series from a dataset constructed using all available daily and weekly series from the Federal Reserve Bank of St. Louis Economic Database (FRED) [1] with a consistent time series from January 1980 to the current day.

Ultimately, we find all models tested beat the benchmark of an autoregressive specification for industrial production with the artificial neural network using F regression feature selection providing the most accurate nowcasts.

The rest of the report is structured as follows: Section 2 provides background information on industrial production and nowcasting. Section 3 clarifies the aims of the project before Sections 4 and 5 highlight the data collection and pre-processing required. Sections 6, 7 and 8 explain the pipeline, feature selection and models used, and Section 9 analyses the results of these models. The final section concludes and highlights some areas of potential future research.

2 Background

2.1 Industrial Production

Industrial production (IP) measures the economic output of the industrial sector of an economy. Whilst the sectors that IP is composed of vary from country to country, in the US, which is the focus of this project due to the reliability, wide variety, and timeliness of the country's data, IP contains the extraction, manufacturing, and utilities subsectors.

The US industrial production index is primarily calculated across 296 subsectors from output of physical units, such as tonnes of steel. For sectors where output data is unavailable, their output is estimated from production-worker hours in that sector [2]. Each sector is subsequently weighted by their total share of output within the overall industrial sector to create the final industrial production index.

Whilst the share of the industrial sectors within gross domestic product (GDP) has typically decreased in western economies as they transition to be more service orientated (Value added as a share of GDP for the sum of the extraction, manufacturing and utilities sectors fell in the US to 13.3% in 2020 from 19.2% in 1997 [3]), the industrial production index is still a widely observed measure. Due to its diverse composition, covering both consumer facing sectors such as food, beverage, and tobacco, as well as business facing industries such as machine tools, industrial production is sensitive to economic drivers including investment, interest rates and consumer spending. As a result, IP is often an important variable in forecasting GDP, whilst it is also observed by financial markets and policy makers at central banks as an indicator of inflation.

2.2 Nowcasting

Nowcasting is a forecasting approach which seeks to forecast the current point in time or the very near-term. As releases of economic data typically lag by several months, policymakers, central bankers, and financial sector workers, rely on nowcasts to bridge this knowledge gap, and provide real time information on how the economy is performing.

Nowcasting is typically done by using the higher frequency data that is available between releases of the lower frequency target series. For instance, if the target series is released at a quarterly frequency, as is the case with GDP, then monthly, weekly, and daily series could be used to drive the nowcast.

US industrial production is published at a monthly frequency, typically around the 15th of each month, with an initial estimate of the month prior. Thus, if on the 10th of July someone required a nowcast of July’s IP figure, there are 10 days of daily and weekly frequency series already available within July that they can use to produce the nowcast. Additionally, there are a further 21 days within the month where additional releases of higher frequency data can inform the IP nowcast ahead of the official release of the July IP figure in mid-August.

Nowcasts at institutions such as central banks are typically performed using econometric methods, such as the dynamic factor model used to nowcast US GDP at the Federal Reserve Bank of New York [4]. However, there is a growing literature on using machine learning models for nowcasting. Loermann and Maas [5] illustrate effective use of a feed forward artificial neural network for nowcasting US GDP, whilst Richardson et al [6] find success using a collection of machine learning models for nowcasting New Zealand GDP, including a feed forward neural network, regression SVM and ridge, lasso and elastic net regression. Additionally, Hopp [7] uses long short-term memory (LSTM) neural networks to nowcast global merchandise exports in both value and volume terms, as well as global services trade data, finding better empirical performance in terms of root mean squared error than dynamic factor models.

As such, this project contributes to the research in economic nowcasting by using machine learning models to nowcast a previously uncovered target variable of industrial production. Additionally, this project introduces a new aspect of research by nowcasting a monthly frequency variable using daily and weekly frequency series, whereby the previous target variables in [5, 6, 7] were at a quarterly frequency and relied on monthly frequency variables to drive the nowcast.

3 Aims and objectives

The aim of this project is to identify a machine learning model which can produce accurate nowcasts of US industrial production, based on evaluation of the root mean squared error (RMSE) between the nowcast and actual data. Ideally, the chosen model should outperform an autoregressive model of IP which will serve as an econometrics benchmark. The model should also be relatively efficient to run to enable the nowcast to be performed daily and account for new releases of higher frequency data.

The nowcast models should also make use of data science techniques to identify relevant features from the higher frequency dataset for the nowcast, as opposed to using traditional economic theory to identify the regressors. As such, several feature selection techniques will be trialled as well as testing all available features in each model.

4 Data collection

4.1 Data sources

Higher frequency data is sourced from the Federal Reserve Bank of St. Louis Economic Database (FRED) [1]. As of September 2021, this database contains 815,000 series from 107 sources, covering international, national, and regional statistics across the economic spectrum including labour market conditions, prices, and financial indicators such as exchange and interest rates.

The target US industrial production series has been sourced from the FRED MD database [8], which carries monthly observations from January 1959 to the current month. This database is frequently used for macroeconomic research as it is a reliable source of data and is maintained by 'data specialists' at FRED. This results in consistent series with no missing values which is a requirement for many machine learning models and reduces the amount of pre-processing required for the target IP series.

4.2 Constructing the higher frequency dataset

Higher frequency data was obtained from the FRED database using their application programming interface (API). A combination of the third-party API wrapper fredapi [9] and the python requests library was used to download the data.

As fredapi does not have a facility to download data via the "tags" attribute of each series, which contains their frequency information, the relevant URLs were generated for identifying series with the daily and weekly tags by following the FRED API documentation. The requests library was then used to access these URLs and download all the mnemonics of the series tagged with daily and weekly frequencies in the FRED database and store these in respective lists. These lists were then used by the fredapi module to download the data of each series to respective daily and weekly pandas data frames, with observations from January 1st 1959 to the latest release at that point in time. This methodology obtained daily and weekly frequency data frames containing 1088 and 3498 series respectively. Finally, these data frames can be output to a CSV file or SQL database for use in the machine learning pipeline.

To ensure there are a significant number of observations to train the machine learning models for nowcasting, a cut off starting point of at least January 1st 1980 was chosen with consistent time series out to the current day required. This results in a significant omission of series from the original data download.

For instance, series that have time series from January 1st 1980 but have since been discontinued are removed from the downloaded dataset. For example, DBKAC (3-month bankers acceptance rate) is removed, which carries daily observations from May 1966 to July 2000. Dropping discontinued series such as this one maintains the observations required for training and ensures that we will only be using features that can provide additional information to the nowcast, with discontinued series unable to provide new information at the current point in time as they are no longer released between observations of the lower frequency target series.

This cut off also means that we must omit newer but potentially useful series such as OVXCLS (Chicago Board Options Exchange's Crude oil exchange traded funds volatility index). This series carries daily observations from May 2007 to the latest working day and may have been a useful feature in explaining the extraction subsector of industrial production.

Whilst this cut off and subsequent removal of higher frequency series may seem harsh, it is vital to ensure the reasonable training sample for the machine learning models. Additionally, by choosing this cut off, we lose 20 years of data from the target monthly IP series which could be used for training. However, by taking the cut off back to the same start date as IP of January 1st 1959, there are only 14 available higher frequency series with consistent time series out to the current point in time. Thus, the cut off on January 1st 1980 provides a middle ground in length of the time series for training and the number of higher frequency features available.

This results in a final higher frequency dataset containing 140 series, composed of 53 daily and 87 weekly series. Information on the final higher frequency series in the dataset can be found in the Appendix Table A1.

4.3 Vintagisation

Vintagisation refers to the creation of different vintages of the dataset, with each vintage containing the data that was available at that particular point in time. This is relevant when it comes to economic data as series are often revised following their initial release to account for changes and corrections to the underlying component series and in some case changes in methodology. US IP faces monthly revisions for recent observations and undergoes an annual revision to incorporate new or revised annual benchmark data, including data that was reported outside of the designated reporting window and changes to the methodology for calculating the index [10].

Both [5] and [6] use vintagisation to evaluate their nowcast models using the data that was available to the practitioner at the original point in time, avoiding any changes that may have come from revisions.

Whilst outlined as a stage in the project proposal, vintagisation was not deemed possible due to the significant time involved in its collection. The average time taken to download the latest daily and weekly series was found to be 20 and 40 minutes respectively. To perform vintagisation, the vintage of the dataset would need to be downloaded for every working day from January 1st 1980 to the current day. A rough estimation of vintages required to 31st December 2020, assuming there are an average 261 working days in the US each year, results in 10,701 vintages for each frequency, with each frequency taking approximately one hour on average to download. Furthermore, connection with the API would often timeout after downloading several vintages making sustained overnight running difficult.

Instead, a quasi-vintage approach is taken by cutting the dataset at each date and taking the daily and weekly series available up to and including that date. The end point for IP in each vintage will be determined using historical IP release dates [11], with the release in each month deemed to be the initial estimate for the month prior.

This was deemed an acceptable alternative due to the reliability of initial IP releases, with initial estimates only being revised by 0.23 percentage points between first and fourth estimates on average over the 1987-2020 period [2]. Additionally, some higher frequency series which are recorded on financial markets, such as DEXCAUS (daily Canadian to US dollar foreign exchange rates), are not likely to face revision as they are based on market movements from that day in question.

5 Data Pre-processing

Before we can begin training the machine learning models there are several pre-processing steps that must be performed.

5.1 Frequency aggregation

As we will be using models that cannot handle mixed frequencies of data, the first pre-processing step involves aggregating the higher frequency series to the lower monthly frequency. The aggregation method depends on the units of the data, with three different methods identified as average, sum and close. The aggregation method of each series is stored in a dictionary within the python programs and can also be found in Appendix Table A1.

Aggregation by taking the monthly average of the higher frequency series is performed on series which have units in an index or a percentage, such as NFCI (the Chicago Fed National Financial Conditions Index), as aggregation with a sum results in noninterpretable data for these units. Exchange rate series are also aggregated by taking the monthly average.

Similarly, monthly aggregation with summation is used for series that have units as an absolute value such as billions of US$. An example series is TOTCI (Commercial and Industrial loans, all commercial banks).

Finally, the close aggregation method is used for higher frequency series that are reported as their value at market close at the end of their frequency. This is typically used as a measure for financial prices or indices such as WILL5000PR (The Wilshire 5000 price index) which is a market-capitalisation weighted price index of actively traded stocks for companies headquartered in the US. To aggregate a higher frequency close series to monthly frequency, the last reported observation in the month is taken.

Once the higher frequency series are aggregated, they are merged with the monthly US IP series to create a monthly frequency data frame of all series.

5.2 The ragged edge problem

The ragged edge problem refers to the different endpoints between series caused by the lags in publication of economic data. As previously discussed, US IP is released around the middle of each month for the month prior. Thus, at the bleeding edge of the forecast we will have a ragged edge with higher frequency data available from the beginning of the month for the month of interest, but US IP lagging one month behind.

The nowcasts will be performed by training each model on all monthly observations with IP available and then using the model to predict the periods where a ragged edge exists with only higher frequency series available. The nowcasts will be performed on a rolling basis for each day in the month prior to the IP release. This allows us to observe the changes in nowcast performance as more higher frequency information is introduced, with the assumption being that the nowcasts become more accurate the later they are performed in the month as there is more information for them to leverage in its prediction.

Whilst the aggregation methods for series that are aggregated as an average or close value remain the same within the nowcast month where there may yet to be a complete month of data, aggregation via sum is adjusted to avoid artificially deflating the value in the monthly aggregation. For example, if the aggregation is performed with only the first week of higher frequency series available for that month, average and close values will be relatively consistent with the final monthly aggregation assuming there are no outliers, but variables aggregated as a sum will be substantially smaller than if they were calculated using a month of available higher frequency variables.

As such, for daily series, the number of working days in each month is calculated as a proxy for release days. If the number of observations available in the month in question are less than the number of working days, then the sum of the available days is scaled up by the ratio of working days in the month to days observed thus far:

$$\begin{aligned} &Monthly\ Sum\ (daily) \\ &\quad = Sum\ of\ observations\ so\ far * \frac{No.of\ working\ days\ in\ the\ month}{No.of\ observations\ so\ far} \end{aligned} \tag{1}$$

A similar calculation is followed for weekly series, but due to the differing release days of the week for weekly series, a consistent number of releases each month cannot be obtained due to changes in how the calendar falls. A simplifying assumption of at least four monthly releases of each weekly series is made. Thus, making the weekly sum aggregation equation in the month of the nowcast:

$$Monthly\ Sum\ (weekly) = Sum\ of\ observations\ so\ far * \frac{4}{No.of\ observations\ so\ far} \tag{2}$$

5.3 Transformations for stationarity

A stationary time series refers to one which has constant statistical properties over time, such as a constant mean and variance. Stationary time series are often easier to analyse and model as underlying trends and seasonality are removed, giving models a clearer signal to learn. Additionally, stationarity is an assumption of many classical time series methods.

Transformations to make the series stationary are applied after the higher frequency series have been aggregated and combined into one data frame with the monthly IP series. This was done as it was not clear how you would aggregate the differenced higher frequency series.

The first transformation applied to the dataset is to take natural logarithms of all series. This corrects for non-stationarity in the variance of series. For example, WILL5000PR can be viewed to have a non-stationary variance with the variance increasing over time and a left skew in its distribution (Figure 1). Indeed, the variance of the first half of the observations is 12119870.4, rising to 69701555.5 in the second half.

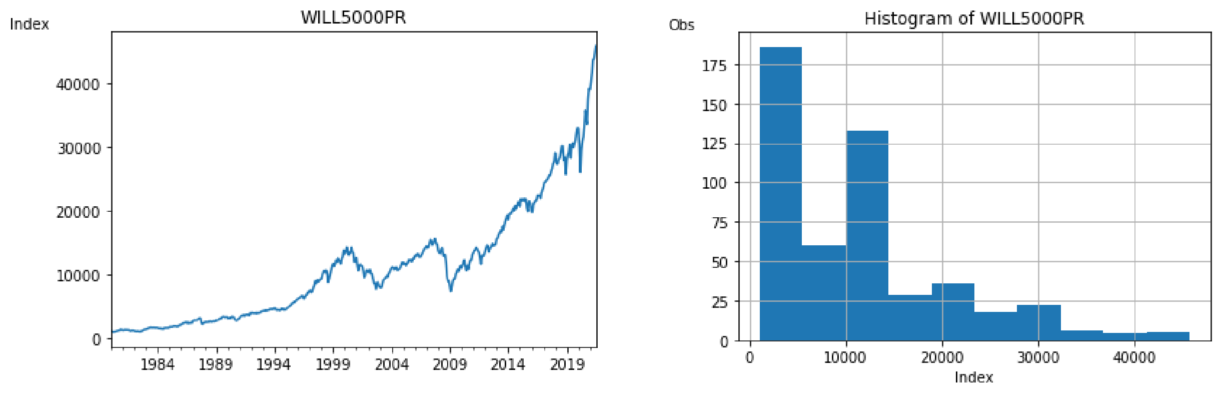


Figure 1 – plot and histogram of WILL5000PR

After taking the natural logarithm of this series (Figure 2), the series appears to have a more constant variance over time, with the variance in the first half of the observations (0.513) being much closer to that of the second half (0.186). Furthermore, the distribution of the series resembles a more normal distribution.

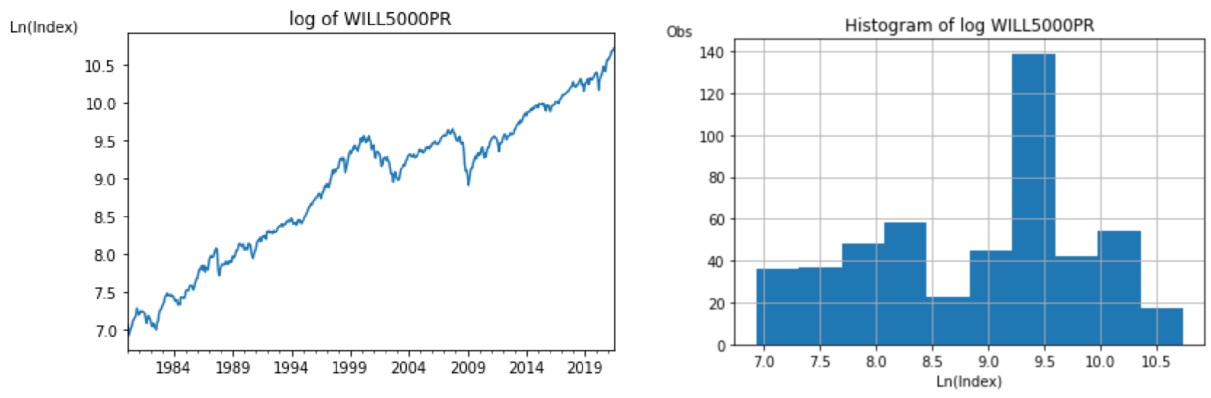


Figure 2 – plot and histogram of ln(WILL5000PR)

One downside to applying the natural logarithm to all the series in the dataset is that it cannot be run on series with negative values. As such, we drop the 13 series containing negative values at this stage leaving us with 128 series, including IP. These series are identified in Appendix Table A1.

Following the log transformations, the augmented Dickey-Fuller (ADF) test is applied to each series. The null hypothesis of the ADF test suggests that there is a unit root present in the time series and as a result the series contains a time dependent trend and is therefore non-stationary. The alternative hypothesis suggests there is no unit root, and the series is stationary. Should the critical values of the ADF for a series fail to record a p-value lower than 0.05 for the 5% significance level, we fail to reject the null hypothesis and we view the series as non-stationary. As such, we take the first difference of the series in a bid to remove the trend. The ADF test will then be used on the series again and further failures to reject the null hypothesis, stating that the differenced series is still non-stationary, will result in further differencing.

We find that of, the remaining 128 series, 115 of these including the target IP series, fail to reject the null hypothesis of the ADF test on their log-levels and as such are deemed to contain a time dependent trend. However, following first differencing, all 115 differenced series reject the null hypothesis and are now deemed to be stationary. WILL5000PR is one of the series that required one difference and can be seen in Figure 3 to now be stationary with relatively constant mean and variance over time. The other series that require a single difference are outlined in Appendix Table A1.

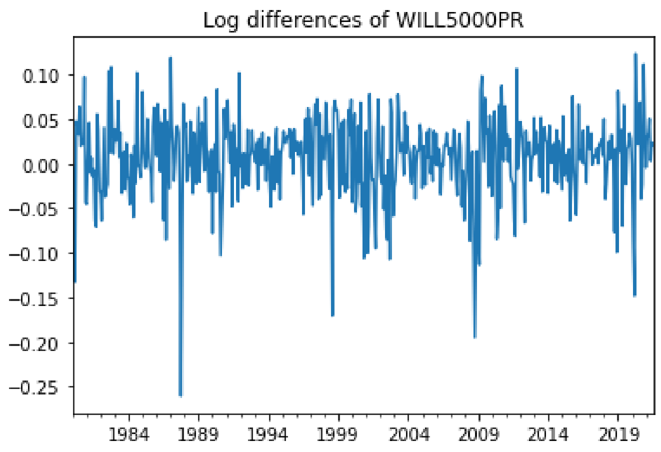


Figure 3 – plot of the log difference of WILL5000PR

Once, all series have been determined stationary by the ADF test, we are ready to begin model tuning.

6 Machine learning pipeline

The pipeline for this project will be composed of two sections. The first will use the whole dataset to perform feature selection and tune the machine learning models. The second pipeline will be used to perform and evaluate the nowcasts on a rolling basis across the quasi-vintages, using the feature selection and model hyperparameters determined by the first pipeline.

6.1 Model tuning pipeline

Following the collection of the higher frequency dataset as outlined in section 4.2, this pipeline will use the latest available vintage of the data and combine each frequency of data into one data frame by aggregating all series to the monthly frequency. Following this, the ragged edge will be removed so there are no NaN values in the time series. Next, the time series will undergo the log-difference transformations to make the series stationary.

Using this pre-processed data, F regression and mutual information regression feature selection techniques will be performed to identify optimal subsets of features.

Both the features selected by these two methodologies and all features in the dataset will be used to train the machine learning models to see if performance can be improved by using a subset of features.

Ahead of hyperparameter tuning for the artificial neural network and LSTM network, all features will be scaled with a min max scaler to normalise the data to values between 0 and 1. This will control for the differing scales from the raw data and resolve large differences in values even after log-differencing the data. Furthermore, using normalised data is likely to result in improved model training and performance.

Hyperparameter tuning will be performed using cross-validation. However, standard k-fold cross-validation cannot be used as the random sampling for each of the k folds will result in a breaking up of the contiguous time series and as such, we may lose some of the sequential and intertemporal information inherent in the order of the time series, resulting in models with poor generalisation performance when applied to the original time series outside of the cross-validation environment.

Instead, we will use scikit-learns TimeSeriesSplit function [12]. This creates a number of train and test splits where at the kth split, the first k folds are returned as the training data set and the (k + 1)th fold as the test set. When we move into the next split, the successive training set is the superset of the training and testing sets that came before them. Figure 4 illustrates the behaviour of the training and testing sets at each fold in the TimeSeriesSplit. As a result, this methodology allows us to tune machine learning models for time series across a variety of different training sets, in a bid to produce models with a low generalisation error.

Once the optimal hyperparameters for each model have been determined, these are used as inputs into the nowcasting pipeline, alongside the feature selections and information on the number of differences required to achieve stationarity.

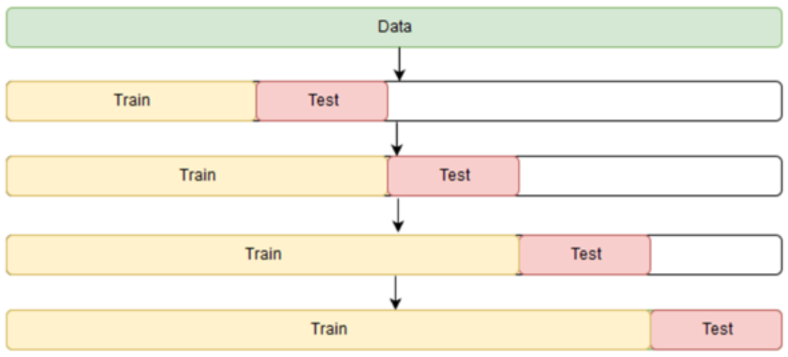


Figure 4 – Illustration of TimeSeriesSplit cross-validation folds, image obtained from [13]

The model tuning pipeline has been visualised in Figure 5.

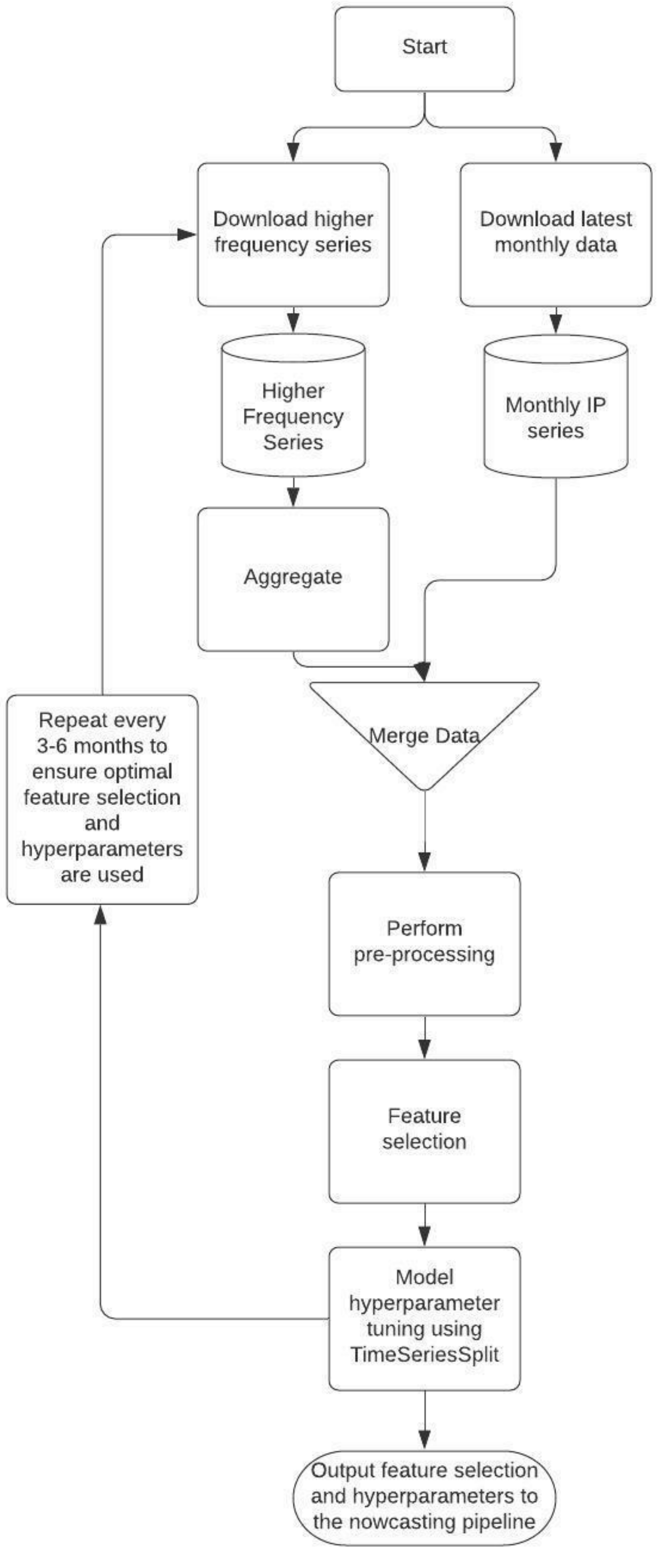

Start
Download higher frequency series
Download latest monthly data
Higher Frequency Series
Monthly IP series
Aggregate
Merge Data
Repeat every 3-6 months to ensure optimal feature selection and hyperparameters are used
Perform pre-processing
Feature selection
Model hyperparameter tuning using TimeSeriesSplit
Output feature selection and hyperparameters to the nowcasting pipeline

Figure 5 – Model tuning pipeline

6.2 Nowcasting pipeline

The purpose of this pipeline is to perform nowcasts at specified vintages, using the hyperparameters obtained from the model tuning pipeline. Performance of these nowcasts will be evaluated to determine the optimal model and feature selection for nowcasting US IP. This pipeline is written as the python functions *rolling_nowcast_v3 and nowcast_vintage_v3* in the project_library.ipynb file. There are also model specific versions in this file, where input parameters or the pipeline may differ depending on the model.

Given a specific vintage date to perform the nowcast at, this pipeline will create quasi-vintages of the higher frequency datasets by cutting them at the vintage date and will ascertain the vintage of IP based on the most recent release of IP to the date. The higher frequency series are then aggregated to monthly frequency and are combined with the IP series. This results in a data frame containing consistent time series, including IP, prior to the month of the vintage, with a ragged edge in the month of the specified vintage date, at which the higher frequency series can be used to inform the nowcast of IP in that month.

This data frame is then transformed with log-differencing to ensure stationarity, with the level of differencing to specific series identified in the model tuning pipeline.

Next, the data frame is cut into training and test datasets whereby all observations prior to the ragged edge of the vintage are used for training and just the aggregated higher frequency variables in the ragged edge will be used in the test set to drive the prediction of the nowcast. Each train and test set is split into the target IP variable and the specified features from the feature selection methods of the model tuning pipeline. If the model is a neural network or LSTM, the data is also normalised with a min max scaler at this stage.

Finally, the model for the nowcast is trained using the hyperparameters determined in the model tuning pipeline. The trained model is then used to predict the higher frequency values in the ragged edge which produces the nowcast for that vintage.

If given several rolling nowcasts to perform, the original vintage date will then be incremented by one day and the pipeline will repeat, but with an additional day of higher frequency information to inform the IP nowcast.

The nowcasting pipeline is presented graphically in Figure 6.

The main reason for not doing hyperparameter tuning at each vintage was to maintain a relatively quick and efficient run time and allow automated rolling nowcasts with the pipeline moving from one vintage to the next without having to stop for hyperparameter tuning. Thus, it is vital to obtain a model with a low variance in the model tuning pipeline for it to not overfit to the training data.

This pipeline has been designed in such a way that it can be easily updated daily and provide real time nowcasts as features are released. Additionally, the model tuning pipeline could be rerun every 3-6 months to incorporate new data releases and check for changes to optimal feature selections and hyperparameters.

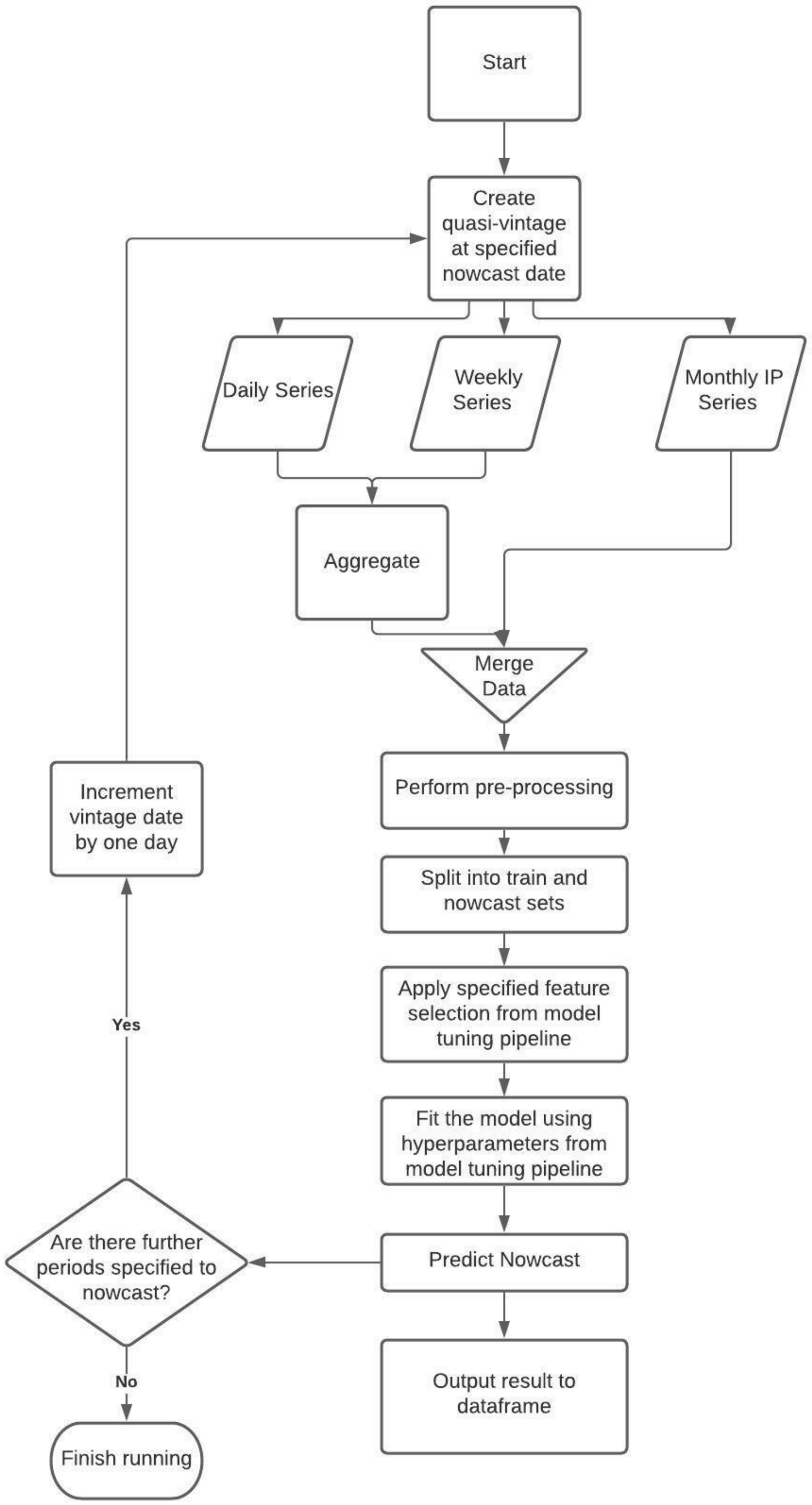

Start
Create quasi-vintage at specified nowcast date
Daily Series
Weekly Series
Monthly IP Series
Aggregate
Merge Data
Perform pre-processing
Split into train and nowcast sets
Apply specified feature selection from model tuning pipeline
Fit the model using hyperparameters from model tuning pipeline
Predict Nowcast
Output result to dataframe
Are there further periods specified to nowcast?
Yes
Increment vintage date by one day
No
Finish running

Figure 6 – Nowcasting Pipeline

7 Feature Selection

Feature selection is a process which seeks to find the optimal features for use in a machine learning model. In this case, we are looking to find the optimal subset of features that results in a lower root mean squared error for the nowcast of IP. Additionally, choosing a smaller subset of features reduces computational costs and can improve the runtime of the model.

Feature selection is performed on the combined aggregated dataset with log-differences taken to ensure the data is stationary. Train and test splits are created with the training dataset containing the first 80% of observations and the testing dataset containing the 20% of observations thereafter. A SelectKBest framework is used with differing scoring functions to fit a linear regression model with the k best features and record its predicted root mean squared error on the training dataset. Different values of k are trialled, starting with all features then decrementing to one feature in a bid to find the optimal subset of features for each scoring function. The optimal subset of features is determined by the value of k which results in the lowest root mean squared error of the benchmark linear regression model for IP.

7.1 F regression score function

The first scoring function used in this framework is F regression [14]. This function computes the correlation coefficient between the target and each feature in turn and then uses the correlation to calculate the F statistic of each feature.

The F statistic tests against the null hypothesis that the target variable does not depend on any of the predictors. A high F statistic means that we can reject this null hypothesis and assume that at least some of the variables are significant.

The k features with the highest F statistics are then selected. As such, this scoring function effectively picks the k features with the strongest correlation with the target variable. As a result, this feature selection methodology may fail to capture non-linear relationships between IP and the features.

The root mean squared error of the linear regression model is found to be the smallest under this scoring function when k = 2, resulting in two features selected by this methodology: DFF (The effective federal funds rate) and DPRIME (the bank prime loan rate).

7.2 Mutual information score function

The second scoring function used for feature selection is mutual information regression [15], which computes the mutual information between each of the features and the target variable. Mutual information measures the dependence between two variables and tells us how much information one random variable can gain from another. If no information is shared between the two variables, then the two variables are independent and have no influence over one another.

The feature selection framework finds that the root mean squared error of the linear regression model is lowest when the top 97 features based on mutual information between the target and the feature are chosen.

One reason why this feature selection methodology results in a much larger subset of features selected than when using F regression as the scorer is that mutual information is not restricted to the linear correlation coefficient of F regression and can capture more complex dependency between variables. The full list of variables selected using mutual information scoring can be found in Appendix Table A1.

8 Models

The nowcast pipeline requires a new model to be trained at each vintage, resulting in the training of many models with gradually changing training data. As such, we seek models that are quick to run and with a low generalisation error. For the nowcasts to be effective we will need to tune models so they do not overfit the training data and thus provide reasonably consistent predictions of the evolving higher frequency data in the ragged edge.

All models are trained using the log-differenced dataset. Data for the artificial neural network and LSTM network is scaled with a min max scaler ahead of training.

8.1 Autoregressive IP model

Autoregressive (AR) models are a linear regression model that takes lagged values of the target as input:

$$y_t = \beta_0 + \beta_1 y_{t-1} + \cdots + \beta_p y_{t-p} + \epsilon_t \qquad (3)$$

The AR model will not produce nowcasts of IP as it does not use the higher frequency in its prediction, but they can still provide a benchmark root mean squared error for the machine learning models to beat.

AR models assume that there is correlation between the variable and its lagged value, known as autocorrelation. To fit the AR model, we need to determine the lag length, p. The easiest way to determine this is by plotting the autocorrelation and partial autocorrelation functions of IP. The autocorrelation function calculates the correlation between the variable at time t and at successively lagged periods. Similarly, the partial autocorrelation function calculates the correlation between the variable at time t and at successively lagged periods but removes the influence of the periods between time t and the lag in question, leaving only the correlation not explained by the previous lags.

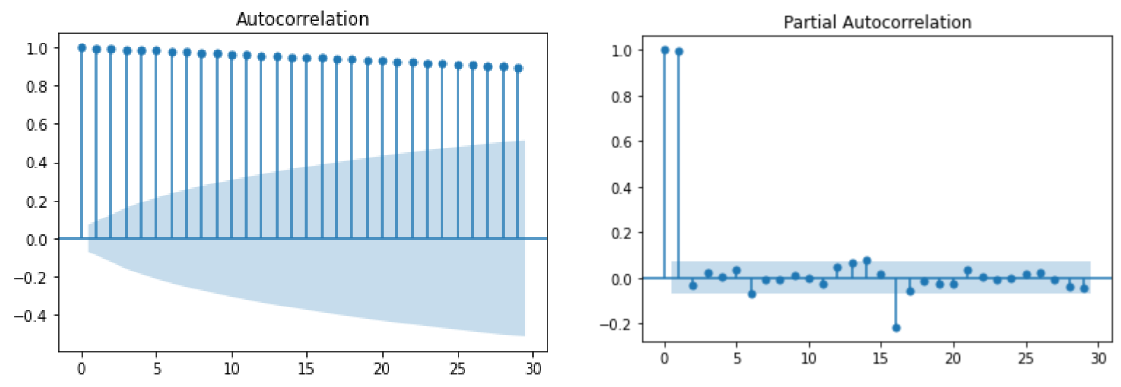


Figure 7 – ACF and PACF plots of IP

The ACF plot in Figure 7, illustrates a slow decay in autocorrelation between periods for IP, with the autocorrelation deemed statistically significant out to lag 30 as the points lie outside of the shaded area. However, the PACF plot only shows significance at the first lag, suggesting that all subsequent autocorrelations can be explained by the first lag. The slow decay in the ACF plot suggests that differencing should be applied to remove the trend and make it stationary. This supports the results of the ADF tests in section 5.3 that one difference of IP is required to make it stationary.

Replotting the ACF and PACF plots using log-differenced IP (Figure 8), we now observe significant autocorrelation and partial autocorrelation only at the first lag.

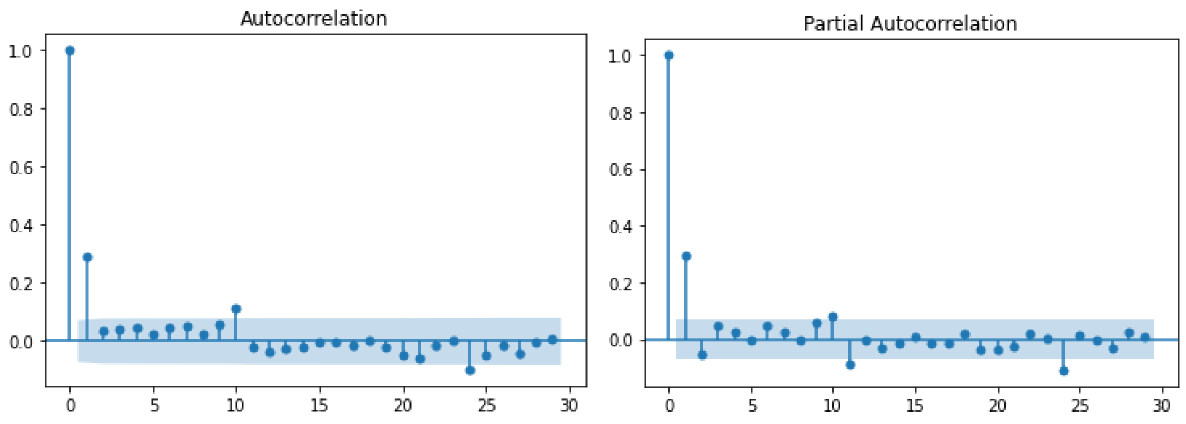


Figure 8 - ACF and PACF plots of log-differenced IP

Thus, the optimal AR model to use for our benchmark will be an AR(1) model with the log-differenced IP series regressed on one lag of itself:

$$IP_t = \beta_0 + \beta_1 IP_{t-1} + \epsilon_t \qquad (4)$$

8.2 Ridge (L2) Regression

Ridge regression is an adaptation of linear regression that seeks to improve long run predictions and reduce the variance of the model by introducing a small amount of bias in a regularisation term.

For a linear regression, we aim to minimise the sum of squared residuals cost function, where we have M observations and p features:

$$\sum_{i=1}^{M} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{P} w_j * x_{ij} \right)^2 \qquad (5)$$

When performing ridge regression, a penalty is introduced equivalent to the sum of the squares of the coefficients, with the lambda term controlling the size of its impact:

$$\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{P} w_j * x_{ij}\right)^2 + \lambda \sum_{j=0}^{P} w_j^2 \qquad (6)$$

As such, coefficients are penalised when they are large, resulting in the loss function of the ridge regression shrinking the magnitude of coefficients when compared to those of the linear regression. This results in the model being less sensitive to particular features and allows the model to generalise better to unseen data. The shrinkage of coefficients also helps to resolve issues of multicollinearity, where the independent features are highly correlated with each other, as multicollinearity causes coefficients to be large and sensitive to changes in the data.

To tune a ridge regression model, we need to determine the optimal value of lambda (alpha in the scikit-learn implementation). When lambda is 0, the ridge regression is equivalent to a linear regression, with the strength of regularisation increasing with lambda.

To determine the value of lambda we use the cross-validation procedure outlined in section 6.1 across a time series split of 10 splits for each value of lambda tested. A wide range of lambdas were initially tested. Following these initial results, the precision of the lambda values tested were increased and centred on a smaller range around the initially chosen lambda. The optimal value of lambda for each set of features can be found in Table 1.

Features	**Lambda**
All	25.9
F	0.37
MI	30.0

Table 1 – Tuned ridge regression hyperparameters

Both the models for all features and the MI features see a relatively strong L2 regularisation penalty applied. Meanwhile, the lambda value for the model using F features is relatively small at 0.37. This is not surprising considering this model only contains two features and so is already less complex when compared with the other models.

8.3 Lasso (L1) Regression

Lasso (least absolute shrinkage and selection operator) regression is also a regularised linear regression model which seeks to improve the generalisation of the model, but the penalty is now on the absolute value of the sum of the coefficients in the cost function:

$$\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{P} w_j * x_{ij}\right)^2 + \lambda \sum_{j=0}^{P}\left|w_j\right| \tag{7}$$

The key difference between lasso and ridge regression is that the regularisation in lasso regression can shrink coefficients to 0, removing less useful features and producing a sparse solution, whilst ridge regression can only shrink coefficients asymptotically close to zero, meaning the influence of a feature is never fully removed. As such, lasso regression can provide an additional layer of feature selection by setting the coefficient of some variables to 0 during regularisation and removing their impact on the model.

Lasso regression can struggle with multicollinearity. If two features are highly correlated and the L1 penalty deems their coefficients large enough to remove, lasso regression will determine the coefficient to shrink to 0 randomly. As such, a feature may be removed that is highly correlated with another feature but has higher predictive influence over the target series.

As with ridge regression, lasso regression models only require hyperparameter tuning of lambda, with a lambda of 0 resulting in a model equivalent to linear regression and strength of regularisation increasing with lambda. The optimal values of lambda were found to be very small for each set of features and can be observed in Table 2.

Features	**Lambda**
All	1E-06
F	1E-06
MI	1E-06

Table 2 – Tuned lasso regression hyperparameters

8.4 Elastic Net Regression

Elastic Net regression expands on the regularised linear regression models by combining the L1 and L2 regularisation penalties of lasso and ridge regression. A new hyperparameter, alpha, is used to determine the contribution of each penalty to the loss function. An alpha of 0.5 would give equal weighting to each penalty, whilst an alpha of 1 would give all the weighting to the lasso (L1) penalty and an alpha of 0 would give all the weighting to the ridge (L2) penalty. As such, the loss function that elastic net regression seeks to minimise is:

$$\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{P} w_j * x_{ij}\right)^2 + \lambda \sum_{j=0}^{P}\left(\alpha\left|w_j\right| + (1-\alpha)w_j^2\right) \tag{8}$$

By combining the two regularisation penalties, elastic net regression can overcome the downsides of ridge and lasso regression when used independently. For instance, ridge regression only shrinks coefficients and as such does not completely remove irrelevant features, whilst lasso regression can wrongly remove useful features in situations of multicollinearity. By combining the techniques, elastic net can perform well with multicollinearity by shrinking highly correlated coefficients (instead of randomly removing one of them) and remove irrelevant features entirely (instead of leaving them with small coefficients from ridge regression).

However, this comes at a cost of computational complexity, particularly when tuning the hyperparameters as there are now two hyperparameters to deal with. As with previous regularised linear models, the value of lambda controls the strength of the regularisation penalty. However, we also need to determine the value of alpha (l1_ratio in the scikit-learn implementation). Over the 10 time series splits alpha was trialled in increments of 0.01 from 0 to 1 whilst a wide range of lambdas were initially tested, before the precision of the lambda values were increased and centred on a smaller range following initial tests. The optimal hyperparameters for each set of features can be found in Table 3.

Features	**Lambda**	**Alpha**
All	0.47	0.001
F	0.0001	0.01
MI	0.0001	0.95

Table 3 – Tuned elastic net hyperparameters

The alpha values for all features and the F features suggest that the L2 penalty contributes to most of the elastic net regularisation in these models, whilst the model for the MI features has a much stronger input from the L1 penalty. However, the overall strength of regularisation is very low for the F and MI features with a lambda value of 0.0001 for both models.

8.5 Artificial Neural Networks

Whilst ridge, lasso and elastic net regression models provide regularisation techniques to reduce the variance of models over the rolling nowcasts, they only capture linear relationships. As such, we use an artificial neural network (ANN) to capture the potentially more complex relationships between the higher frequency series and IP.

An ANN is a series of connected nodes which seek to learn the underlying structure of the training data by mimicking the neurons of the human brain. The ANN is made up of a series of layers of neurons. The first is the input layer which contains the same number of nodes as features in the dataset and is where the data is input into the network. Following this, are the hidden layers, the dimension and number of which should be determined during the network design and hyperparameter training stage. The final layer is the output layer which returns the output of the ANN, with the dimensions of this varying depending on the nature of the problem your network is trying to learn.

Each neuron in the network outside of the input layer takes the weighted sum of inputs from the previous layer and adds a bias term. The weighted sum plus bias is then passed into an activation function which applies a non-linear transformation to the value. Output of the activation function is then passed into the next layer where the process repeats until we reach the output layer.

We use backpropagation to train an ANN. First, we specify a loss function which calculates the error of the ANNs predictions. Training the ANN then seeks to minimise this loss function. In regression problems, such as this one, the mean squared error loss function is typically chosen. On the first full pass through of the ANN (known as an epoch), the weights and bias for the first hidden layer are randomly initialised and used to create the weighted sum that is passed to the activation function in each node and onto the next layer. Once we reach the output of the final layer, the loss is calculated between the ANNs output and expected values, with this error then fed back into the network where the weights and bias are adjusted in a bid to minimise the error. The ANN determines how much to adjust the weights and bias using an optimisation algorithm such as gradient descent. This process iterates for the number of epochs determined by the practitioner.

In designing the ANN for the nowcasting problem we want to ensure that the network does not overfit the training data so it can produce accurate nowcasts of the evolving higher frequency data. As such, the more complex we make the network architecture the more likely it is to overfit the training data and have increased training times. As a result, we opt for an architecture containing a single hidden layer. This choice is supported by Richardson et al [6] who found a 10 node, 1 hidden layer architecture produced very similar results at a significantly reduced runtime to an optimal 10 node, 2 hidden layer neural network for nowcasting New Zealand GDP.

The activation function in the hidden layer will use the rectified linear unit function (ReLU). This adds nonlinearity by returning the value of the weighted sum of inputs plus bias if it is positive or returning 0 if it is negative. ReLU was chosen over other available functions such as sigmoid and tanh as it produces a sparser representation by converting all negative values to 0, reducing the number of firing neurons in each iteration and resulting in faster model convergence and a less complex model [16].

The output layer will consist of a single neuron using a linear activation function to return our IP prediction.

The Adam optimiser was chosen for use in the ANN. This is a variant of stochastic gradient descent which has been shown to have a favourable rate of convergence in convex problems when compared to other optimisers [17].

Cross-validation over a 10-fold time series split was used to obtain the optimal hyperparameter values on number of nodes in the hidden layer, epochs, and batch size for each feature selection. Once these hyperparameters were determined, a further 10-fold cross-validation was performed to identify an optimal learning rate for the chosen architecture. The learning rate determines the rate of the weight updates during gradient descent. If it is too low then we will only perform very small updates to the weights at each iteration and the model will be very slow to converge, whilst if the learning rate is too high, we may overshoot the global minimum resulting in a divergent loss function and model performance suffering.

The optimal hyperparameters for each ANN can be found in Table 4.

Features	Nodes in hidden layer	Epochs	Batch size	Learning Rate
All	100	500	128	0.01
F	25	10	256	0.001
MI	50	100	16	0.001

Table 4 -Optimal ANN hyperparameters for each feature selection

One drawback of using an ANN is that they are a black box. This means that we are unable to ascertain the direct impact of a particular feature to the nowcast performance. This may become an issue if nowcast performance starts to deteriorate or if a practitioner seeks to identify how changes in a particular feature will impact the nowcast.

8.6 Long Short-Term Memory Networks

The final model we use for nowcasting is the long short-term memory (LSTM) neural network. This model adds a new element to the predictions by allowing an intertemporal element to be introduced. As a result, the model can capture longer term dependencies within the time series.

An LSTM was chosen over a vanilla recurrent neural network (RNN) as this can suffer from the vanishing gradient problem. RNNs introduce an intertemporal component by augmenting the hidden layer of a network at time t, with the values of the hidden layer from the previous time step, t-1, thus introducing a form of memory. In theory, this recurrent element does not have a fixed context limit and can stretch back to the beginning of the time series. An additional set of weights, also trained during backpropagation, are introduced to determine the strength of this recurrent input. However, as each recurrent layer influences the loss in the period ahead of it, during training the hidden layers face repeated multiplications, often with very small numbers, which result in the gradients heading to zero and the recurrent impact diminishing [18].

LSTMs seek to rectify this issue by introducing gates and a cell state inside their neural unit which can help to manage the information that is deemed useful over time periods. The cell state contains the memory of the network and regulates the content over time as information is added or removed from this based on the gates. Each gate contains a feed forward layer and then a sigmoid activation function:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \qquad (9)$$

The range of output from the sigmoid function is 0 to 1 and tends to force small and large numbers to the respective ends of the range. As such, when the product of the sigmoid output is taken with the values being gated, information that is no longer relevant should have a value closer to zero and remove itself by multiplying by very small values, whilst useful information will be saved by multiplying by values close to 1.

The LSTM neural unit starts by deciding on the relevant information to keep from previous periods using the forget gate. The forget gate receives input from the previous hidden layer and current time series and computes their weighted sum. This is then fed into the sigmoid activation function and multiplied with the previous cell state to identify information no longer needed.

Next, the information from the previous hidden state and the current time series is calculated using their relevant weights as candidates for consideration in the current cell state. This uses the tanh activation function:

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \qquad (10)$$

This activation function is used to avoid the vanishing gradient problem between recurrent hidden layers by regulating values between -1 and 1.

Now, the input gate determines what information from the candidate values should be added to the cell state for use by future time periods. This takes weighted input from the current time series and the previous hidden state but uses a sigmoid activation function. When the product of the input gate and the candidate series is taken, irrelevant series are removed thanks to the sigmoid activation function of the input gate.

The new cell state is then calculated as a sum of the product of the forget gate and previous cell state and the product of the input gate and candidate series. As a result, irrelevant information is removed from the previous cell state with new information added from the current time series.

The final stage of the LSTM neural unit is to determine the output of the hidden layer. The output gate takes weighted input from the current time series and the previous hidden state but uses a sigmoid activation function to determine the values to pass into the hidden layer. The final hidden layer is the product between the output gate and the current cell state with values input through tanh to prevent the vanishing gradient problem.

The LSTM is trained using backpropagation, which determines the weights for each of the gates in the neural units and the weights between the hidden layers.

As a result, LSTMs ensure a clear intertemporal connection between neural units and overcome the vanishing gradient issue of vanilla RNNs.

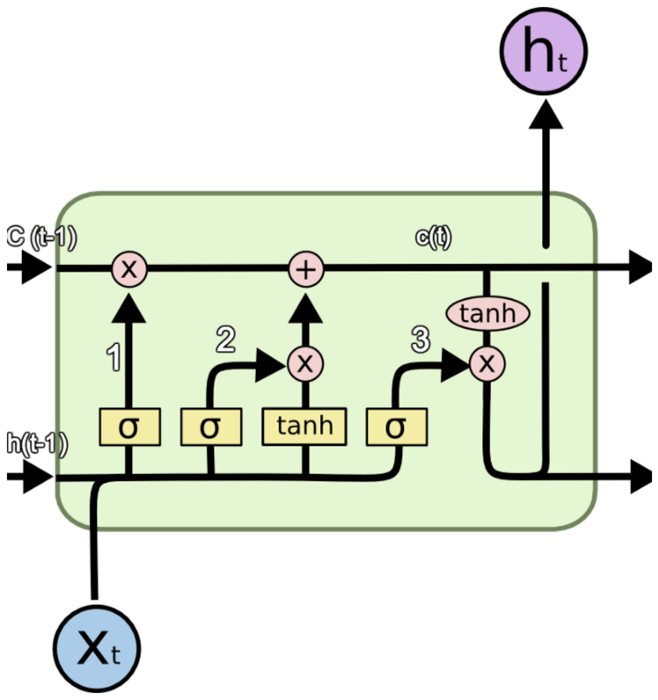


Figure 9 – An LSTM neural unit, 1 – forget gate, 2 – input gate, 3 – output gate. [19]

As with the ANNs, we want to create an LSTM architecture that is not prone to overfitting. As such, we only use one hidden LSTM layer to keep the model as simple as possible. Due to the increased computational demand of the LSTM model, we searched for the LSTM hyperparameters using trial and error instead of cross-validation.

The Keras implementation of an LSTM layer takes a sequence of previous observations as input to an output observation. Therefor we need to convert the input shape of our data to represent the lagged sequence of variables to output at each time step. The split_sequences function used to do this was adapted from the function of the same name in [20].

Choosing the length of sequence to use as an input is a key parameter for an LSTM. Several lag lengths were trialled, including 24 months, 13 months, 3 months and 1 month. It was found that shorter lag lengths were more effective, with longer lag lengths appearing to make the forecast smoother and less reactive to changes in economic data than when compared to the actual IP data. The optimal lag length was determined to be 1 month as this resulted in the lowest root mean squared errors of the models tested. This also mirrors the autocorrelation analysis performed on the log-differenced IP series in section 8.1, where only one lag of IP was found to be necessary.

As such, our LSTM architecture consists of an input layer taking sequences of one lag as input, followed by a single hidden LSTM layer using a ReLU activation function, and a final output layer containing a single node with a linear activation function which returns our prediction of IP. The full hyperparameters of the LSTM models are outlined in Table 5.

Features	**Neural units in LSTM layer**	**Epochs**	**Batch size**	**Learning Rate**
All	100	100	64	0.001
F	200	100	32	0.001
MI	100	100	64	0.001

Table 5 - LSTM hyperparameters for each feature selection

9 Results

9.1 Forecast evaluation procedure

Each trained model is used in the nowcast pipeline and produces daily nowcasts from 1st January 2017 out to 26th July 2021 – the last observation in the vintage of data used to tune the models. This period covers both an initial time span of relative economic stability and a significant economic shock from the coronavirus pandemic, allowing us to test the models in both environments.

As such, the models are initially trained on all the observations prior to January 1st 2017 and then estimate the higher frequency data available on January 1st 2017. The nowcasts are performed on a rolling basis to account for the higher frequency data released each day, with the training period shifting when a new observation of IP is released, according to their historical release dates [11].

It can be shown that for small growth rates, the log-difference between one period is equivalent to the growth rate [21]:

$$\frac{x_t - x_{t-1}}{x_{t-1}} \approx \ln(x_t) - \ln(x_{t-1}) \qquad (11)$$

As we are using the log-difference of the IP series as the target, we are effectively nowcasting the monthly growth rate of IP.

As well as evaluating the forecast accuracy on the nowcasted log-difference of IP, we will also evaluate it in level terms. As each nowcast is performed using all monthly observations before it only to predict one month ahead, we will reconstruct the level for analysis by growing the previous actual level forward with the growth rate nowcasted at the current period:

$$nowcast\ level_t = actual\ level_{t-1} * (1 + \hat{y}) \qquad (12)$$

As such, this prevents errors in the nowcast propagating through the entire forecast period. The level predictions of the AR(1) benchmark model is calculated in the same way to ensure comparability with the nowcasts.

The root mean squared error (RMSE) is used to assess the accuracy of the nowcasts:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T}(y_t - \hat{y}_t)^2}{T}} \qquad (13)$$

Where T is the total number of forecasts and y_t and $\hat{y}_t$ are the actual and forecast values of the target series. This tells us the square root of the average squared difference between our prediction and actual values. A lower value signifies better performance as it indicates that the predicted values are closer to the actual values. The RMSE values are in the units of the series being predicted leading to easier interpretation than other measures such as mean squared error.

Due to the significant run times for the ANN and LSTM models to nowcast over four and a half years' worth of observations, we opt to use a saved specification of each model trained on the first 80% of the full dataset to provide the nowcast instead of retraining the models at each vintage.

9.2 End of month nowcast performance

We first evaluate the model performance using nowcasts taken at the end of the month (EOM). In theory these should be more accurate than nowcasts taken at earlier points in the month, as they have the maximum amount of information from the higher frequency data available to inform the nowcast.

Growth, EOM nowcast			
Model	**All features**	**F features**	**MI features**
Ridge	0.02372	0.02028	0.02401
Lasso	0.01922	0.01973	0.02283
Elastic Net	0.01973	0.01979	0.02440
ANN	**0.01763**	**0.01408**	**0.02012**
LSTM	0.02124	0.01545	0.02129
AR(1) benchmark	0.02525		

Table 6 – RMSE of predicted log-difference (growth) IP series taken from nowcasts performed at the end of the month, bold results indicate optimal model for each feature selection

Level, EOM nowcast			
Model	**All features**	**F features**	**MI features**
Ridge	2.17210	1.84841	2.20061
Lasso	1.78381	1.79890	2.07084
Elastic Net	1.84020	1.80386	2.23501
ANN	**1.61289**	**1.33246**	**1.83202**
LSTM	1.92808	1.39396	1.93495
AR(1) benchmark	2.28128		

Table 7 – RMSE of predicted level IP series taken from nowcasts performed at the end of the month, bold results indicate optimal model for each feature selection

From tables 6 and 7 we can see that each of the models are successful in outperforming the benchmark AR(1) model, with the ANN producing the lowest RMSE across all feature selection methods. The ANN trained using the features from the f selection methodology obtains the most accurate overall nowcasts.

Across all models, the F regression feature selection usually offers the most accurate nowcast results. Only the elastic net and lasso models produce more accurate growth nowcasts when using all features, with only the lasso model producing more accurate level nowcasts when using all features.

Whilst it is somewhat of a surprise that the optimal feature selection only contains two features, it does make economic sense. The two variables selected with F regression, DFF (The effective federal funds rate) and DPRIME (the bank prime loan rate), can influence industrial production by impacting the level of bank lending to both consumers and businesses and thus both the demand for and supply of industrial products. Furthermore, interest rates having been used successfully as a subset of regressors for US IP in [22].

From observing charts A2-A16 in the appendix, it appears the main strength of the models using the F regression feature selection was more accurate nowcasting during the initial downturn at the start of the coronavirus pandemic in March 2020, as they typically nowcast stronger contractions in growth than other feature selection models.

Indeed, the optimal ANN model using F regression features appears to nowcast the downturn relatively accurately (Figure 10). This makes sense given the Fed reduced the effective federal funds rate to near zero at the start of the pandemic and has kept it relatively low ever since [23], which may explain why this model fails to capture the strong rebound in IP in the summer of 2020.

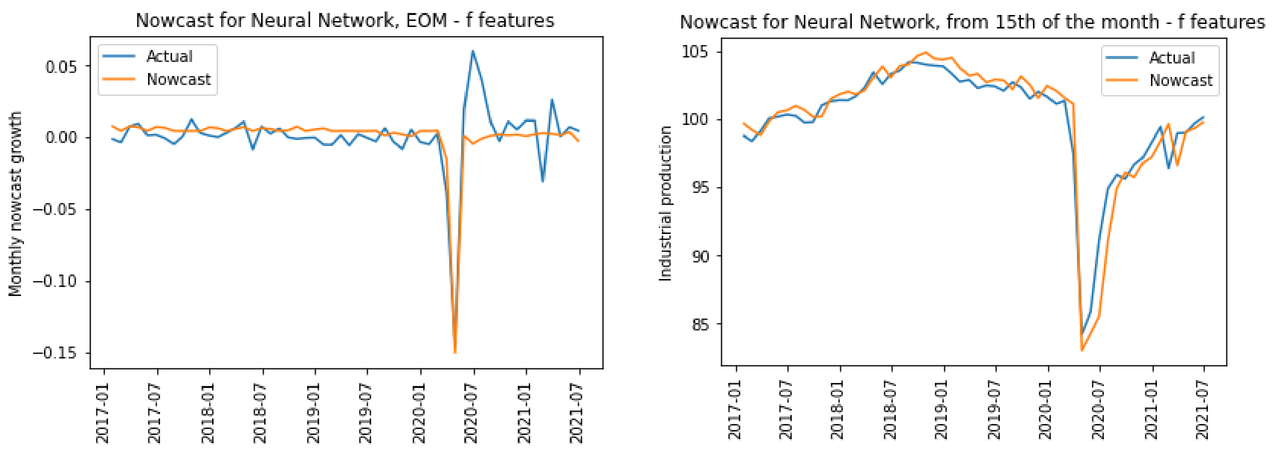


Figure 10 - Nowcast results – ANN, F features, End of month

One drawback of the F feature selection methodology appears to be relatively constant nowcasts of monthly IP growth from January 2017 to early 2020, and as such it might be worth using a model trained with all features to nowcast IP during periods of relative economic stability.

9.3 Middle of month nowcasting performance

As practitioners will wish to perform nowcasts from any point in the month, we also assess the RMSE of the nowcasts when performed on the 15th of each month.

Growth, 15th of month nowcast			
Model	**All features**	**F features**	**MI features**
Ridge	0.02378	0.02069	0.02406
Lasso	**0.01741**	0.02020	0.02329
Elastic Net	0.01743	0.02025	0.02449
ANN	0.01847	**0.01442**	0.02122
LSTM	0.02149	0.01613	**0.02109**
AR(1) benchmark	0.02525		

Table 8 – RMSE of predicted log-difference (growth) IP series taken from nowcasts performed on the 15th of the month, bold results indicate optimal model for each feature selection

Level, 15th of month nowcast			
Model	**All features**	**F features**	**MI features**
Ridge	2.17872	1.89020	2.20527
Lasso	1.57390	1.84578	2.11075
Elastic Net	**1.56986**	1.85028	2.24280
ANN	1.68535	**1.36692**	1.93183
LSTM	1.95145	1.46299	**1.91413**
AR(1) benchmark	2.28128		

Table 9 – RMSE of predicted level IP series taken from nowcasts performed on the 15th of the month, bold results indicate optimal model for each feature selection

Whilst model performance has deteriorated across almost all models due to less days of higher frequency data being available to inform the nowcast, all models remain successful in outperforming the benchmark AR(1) model.

At this point in the month, there is no model that offers the best performance across all features selection methodologies, but the ANN trained using the F regression features still offers the best overall nowcast accuracy.

The F regression feature selection continues to offer the most accurate nowcast results across both the growth and level predictions for almost all models, with the only exceptions being for level value nowcasts using lasso and elastic net regression trained with all features. From charts A17-A31 in the appendix, we see those models using the F regression features continue to capture the downturn at the start of the coronavirus pandemic, even when using fewer days of information to inform the nowcast.

9.4 Final comments

Whilst all models successfully outperform the autoregressive benchmark, the ANN trained using the F regression features is a standout model for nowcasting, particularly in periods of economic shock. In periods of relative economic stability, it may be worth considering all available features with the ANN, to obtain a more reactive nowcast.

Although the ANN models take significantly more time to run than the regularised linear regression models, it is worth it for the improved performance in the nowcast accuracy.

Whilst models trained using the MI feature selection variables outperform the benchmark, their performance is usually limited when compared to other models and thus should be avoided.

10 Conclusion

Overall, this project has been successful in identifying several machine learning models and feature selection methods for nowcasting US IP, which outperform the benchmark autoregressive model. The optimal model identified is the ANN using F regression feature selection. However, the performance of the models can likely be improved by adding more relevant higher frequency series, trialling different solutions for the ragged edge, and testing more models.

The higher frequency dataset constructed from the FRED database largely contains series relating to financial measures following removal of series due to short or incomplete time series. Whilst these variables still produced reasonable nowcasts, it may be useful to supplement the higher frequency data of the FRED with other sources to include series that relate more directly to industry or the consumer, such as daily electricity use by specific industries, road haulage, inventory stock levels or consumer and business confidence measures.

Furthermore, the approach to the ragged edge could be adapted to use more advanced solutions such as forecasting the higher frequency values out to the end of the month before aggregating them, instead of taking simple averages or scaling the values depending on the units of the series.

Finally, I was surprised that the LSTM model didn’t offer the best performance for the nowcast due to its recurrent components allowing information from previous time periods to influence the nowcast. Thus, given more time I would try more complicated specifications of LSTM and improve the hyperparameter selection process.

Bibliography

[1] Federal Reserve Bank of St. Louis, “FRED Economic Data,” Federal Reserve Bank of St. Louis, [Online]. Available: https://fred.stlouisfed.org/. [Accessed 2021 June 24].

[2] Board of Governors of the Federal Reserve System, “Industrial Production Explanatory Notes,” Board of Governors of the Federal Reserve System, 28 May 2021. [Online]. Available: https://www.federalreserve.gov/releases/g17/ipnotes.htm. [Accessed 17 August 2021].

[3] US Bureau of Economic Analysis, “Value added by Industry as a Percentage of Gross Domestic Product,” US Bureau of Economic Analysis, 24 June 2021. [Online]. Available: https://apps.bea.gov/iTable/iTable.cfm?reqid=150&step=2&isuri=1&categories=gdpxind. [Accessed 17 August 2021].

[4] Federal Reserve Bank of New York, “Nowcasting Report Methodology,” Federal Reserve Bank of New York, [Online]. Available: https://www.newyorkfed.org/research/policy/nowcast/methodology.html. [Accessed 18 August 2021].

[5] J. Loermann and B. Maas, “Nowcasting US GDP with artificial neural networks,” MPRA Paper 95459, University Library of Munich, Germany, 2019.

[6] A. Richardson, T. van Florenstein Mulder and T. Vehbi, “Nowcasting New Zealand GDP Using Machine Learning Algorithms,” CAMA Working Paper 47/2018, Centre for Applied Macroeconomic Analysis, Crawford School of Public Policy, The Australian National University, 2018.

[7] D. Hopp, “Economic Nowcasting with Long Short-Term Memory Artificial Neural Networks (LSTM),” United Nations Conference on Trade and Development, UNCTAD Research Paper No. 62, 2021.

[8] M. W. McCracken and S. Ng, “FRED-MD: A Monthly Database for Macroeconomic Research,” *Journal of Business & Economic Statistics,* vol. 34, no. 4, pp. 574-589, 2016.

[9] M. Meyhar, “fredapi,” 14 September 2014. [Online]. Available: https://github.com/mortada/fredapi. [Accessed 1 April 2021].

[10] Board of Governors of the Federal Reserve System, “Industrial Production and Capacity Utilization: The 2021 Annual Revision,” Board of Governors of the Federal Reserve System, 18 May 2021. [Online]. Available: https://www.federalreserve.gov/releases/g17/Revisions/Current/DefaultRev.htm. [Accessed 2 September 2021].

[11] Board of Governors of the Federal Reserve System, “Industrial Production Release Dates (hisotical and scheduled),” Board of Governors of the Federal Reserve System, [Online]. Available: https://www.federalreserve.gov/releases/g17/release_dates.htm. [Accessed 18 August 2021].

[12] scikit-learn, “sklearn.model_selection.TimeSeriesSplit,” scikit-learn, [Online]. Available: https://scikit-

learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html. [Accessed 8 August 2021].

[13] S. Shrivastava, “Cross Validation in Time Series,” Medium, 14 January 2020. [Online]. Available: https://medium.com/@soumyachess1496/cross-validation-in-time-series-566ae4981ce4. [Accessed 8 August 2021].

[14] scikit-learn, “sklearn.feature_selection.f_regression,” scikit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_regression.html. [Accessed 8 August 2021].

[15] scikit-learn, “sklearn.feature_selection.mutual_info_regression,” scikit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_regression.html. [Accessed 8 August 2021].

[16] C. Versloot, “ReLU, Sigmoid and Tanh: Today's most used activation functions,” Machine Curve, 4 September 2019. [Online]. Available: https://www.machinecurve.com/index.php/2019/09/04/relu-sigmoid-and-tanh-todays-most-used-activation-functions/. [Accessed 4 September 2021].

[17] D. P. Kingma and J. L. Ba, “Adam: A Method For Stochastic Optimization,” *arXiv:1412.6980,* 2017.

[18] D. Jurafsky and J. H. Martin, “Chapter 9: Deep Learning Architectures for Sequence Processing,” in *Speech and Language Processing*, 2021.

[19] C. Olah, “Understanding LSTM Networks,” 27 August 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed 6 September 2021].

[20] J. Brownlee, “How to Develop LSTM Models for Time Series Forecasting,” Machine Learning Mastery, 28 August 2020. [Online]. Available: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/. [Accessed 6 September 2021].

[21] L. Hendricks, “Growth Rates and Logarithms: A Refresher,” 10 November 2020. [Online]. Available: https://lhendricks.org/econ520/growth/growth_algebra_sl.pdf. [Accessed 15 September 2021].

[22] J. Moody, U. Levin and S. Rehfuss, “Predicting the U.S. Index of Industrial Production,” in *PASE '93: Parallel Applications in Statistics and Economics*, Zeist, 1993.

[23] J. E. Ihrig, G. Weinbach and S. A. Wolla, “How the Fed Has Responded to the COVID-19 Pandemic,” Federal Reserve Bank of St. Louis, 12 August 2020. [Online]. Available: https://www.stlouisfed.org/open-vault/2020/august/fed-response-covid19-pandemic. [Accessed 21 Sepetember 2021].

Acronym Glossary

Acronym	Definition
ADF	Augmented Dickey-Fuller test
ANN	Artificial Neural Network
API	Application Programming Interface
AR	Autoregressive model
EOM	End of Month
FRED	Federal Reserve Bank of St. Louis Economic Database
FRED MD	Federal Reserve Bank of St. Louis Economic Database, Monthly Dataset
GDP	Gross Domestic Prodcut
IP	Industrial Production
LSTM	Long Short-Term Memory
MI	Mutual Information
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network

Github repo contents

The files listed are as follows:

- Data_pull_no_vintages.ipynb
  - Accesses the FRED database via the API to download all higher frequency series
- IP_Benchmark_V3.ipynb
  - Creates the benchmark AR model of IP
- PIPELINE_V5_Linear.ipynb
  - Contains the hyperparameter tuning and rolling nowcasts for the linear regularised regression models
- PIPELINE_V5_NN.ipynb
  - Contains the hyperparameter tuning and rolling nowcasts for the neural network models
- PIPELINE_V5_LSTM.ipynb
  - Contains the hyperparameter tuning and rolling nowcasts for the LSTM models
- Project_library.ipynb
  - Contains the functions and dictionaries frequently used across all of the above notebooks
- Current.zip
  - Contains the vintage of the FRED MD database used to train the models for the report
- Daily_2021_7_26.zip
  - Contains the vintage of the daily series used to train the models for the report
- Weekly_2021_7_26.zip
  - Contains the vintage of the weekly series used to train the models for the report
- Ip_release_dates.csv
  - Contains historic IP release dates used for vintagisation
- Nn_all_final.h5
  - Saved model values of the optimal neural network for all features
- Nn_f_final.h5
  - Saved model values of the optimal neural network for F features
- NN_mi_final.h5
  - Saved model values of the optimal neural network for MI features
- Lstm_all_w1_v2.h5

    - Saved model values of the optimal LSTM network for all features
- Lstm_f_w1_v2.h5
    - Saved model values of the optimal LSTM network for F features
- Mi_lstm_w1.h5
    - Saved model values of the optimal LSTM network for MI features

To run the program, save to your local directory and unzip the data files, and run each of the IP_Benchmark_V3 and Pipeline_V5 files in turn.

If you want to update the higher frequency data first, run Data_pull_no_vintages.ipynb first and then update all path names to the latest data in the other files.

Appendix

Table A1 – Feature information

ID	Series	Frequency	Units	Aggregation	Differenced?	Feature Selection
INDPRO	US Industrial Production	Monthly	Index	#N/A	Yes	Target series
DEXCAUS	Canada / U.S. Foreign Exchange Rate	Daily	Canadian $ to 1 U.S. $	AVERAGE	Yes	Not selected
DEXDNUS	Denmark / U.S. Foreign Exchange Rate	Daily	Danish Kroner to 1 U.S. $	AVERAGE	Yes	MI
DEXINUS	India / U.S. Foreign Exchange Rate	Daily	Indian Rupees to 1 U.S. $	AVERAGE	Yes	MI
DEXJPUS	Japan / U.S. Foreign Exchange Rate	Daily	Japanese Yen to 1 U.S. $	AVERAGE	Yes	MI
DEXMAUS	Malaysia / U.S. Foreign Exchange Rate	Daily	Malaysian Ringgit to 1 U.S. $	AVERAGE	Yes	MI
DEXNOUS	Norway / U.S. Foreign Exchange Rate	Daily	Norwegian Kroner to 1 U.S. $	AVERAGE	Yes	MI
DEXSDUS	Sweden / U.S. Foreign Exchange Rate	Daily	Swedish Kronor to 1 U.S. $	AVERAGE	No	MI
DEXSFUS	South Africa / U.S. Foreign Exchange Rate	Daily	South African Rand to 1 U.S. $	AVERAGE	Yes	Not selected
DEXSLUS	Sri Lanka / U.S. Foreign Exchange Rate	Daily	Sri Lankan Rupees to 1 U.S. $	AVERAGE	Yes	Not selected
DEXSZUS	Switzerland / U.S. Foreign Exchange Rate	Daily	Swiss Francs to 1 U.S. $	AVERAGE	Yes	MI

DEXUSAL	U.S. / Australia Foreign Exchange Rate	Daily	U.S. $ to 1 Australian $	AVERAGE	Yes	MI
DEXUSNZ	U.S. / New Zealand Foreign Exchange Rate	Daily	U.S. $ to 1 New Zealand $	AVERAGE	No	MI
DEXUSUK	U.S. / U.K. Foreign Exchange Rate	Daily	U.S. $ to 1 British Pound	AVERAGE	No	MI
DFF	Effective Federal Funds Rate	Daily, 7-Day	%	AVERAGE	Yes	F
DGS1	1-Year Treasury Constant Maturity Rate	Daily	%	AVERAGE	Yes	MI
DGS10	10-Year Treasury Constant Maturity Rate	Daily	%	AVERAGE	Yes	MI
DGS2	2-Year Treasury Constant Maturity Rate	Daily	%	AVERAGE	Yes	MI
DGS20	20-Year Treasury Constant Maturity Rate	Daily	%	AVERAGE	Yes	Not selected
DGS3	3-Year Treasury Constant Maturity Rate	Daily	%	AVERAGE	Yes	MI
DGS30	30-Year Treasury Constant Maturity Rate	Daily	%	AVERAGE	Yes	Not selected
DGS5	5-Year Treasury Constant Maturity Rate	Daily	%	AVERAGE	Yes	MI
DGS7	7-Year Treasury Constant Maturity Rate	Daily	%	AVERAGE	Yes	MI
DPRIME	Bank Prime Loan Rate	Daily	%	AVERAGE	Yes	F, MI
DTB1YR	1-Year Treasury Bill:	Daily	%	AVERAGE	Yes	MI

	Secondary Market Rate					
DTB3	3-Month Treasury Bill: Secondary Market Rate	Daily	%	AVERAGE	Yes	Not selected
DTB6	6-Month Treasury Bill: Secondary Market Rate	Daily	%	AVERAGE	Yes	MI
T10Y2Y	10-Year Treasury Constant Maturity Minus 2-Year Treasury Constant Maturity	Daily	%	AVERAGE	#N/A	Dropped when ln() taken
T10YFF	10-Year Treasury Constant Maturity Minus Federal Funds Rate	Daily	%	AVERAGE	#N/A	Dropped when ln() taken
T1YFF	1-Year Treasury Constant Maturity Minus Federal Funds Rate	Daily	%	AVERAGE	#N/A	Dropped when ln() taken
T5YFF	5-Year Treasury Constant Maturity Minus Federal Funds Rate	Daily	%	AVERAGE	#N/A	Dropped when ln() taken
BAMLC C0A0CM TRIV	ICE BofA US Corporate Index Total Return Index Value	Daily, Close	Index	CLOSE	No	MI
BAMLC C1A013 YTRIV	ICE BofA 1-3 Year US Corporate Index Total Return Index Value	Daily, Close	Index	CLOSE	No	MI
BAMLC C2A035 YTRIV	ICE BofA 3-5 Year US Corporate Index Total	Daily, Close	Index	CLOSE	No	MI

	Return Index Value					
BAMLCC7A01015YTRIV	ICE BofA 10-15 Year US Corporate Index Total Return Index Value	Daily, Close	Index	CLOSE	Yes	MI
BAMLCC8A015PYTRIV	ICE BofA 15+ Year US Corporate Index Total Return Index Value	Daily, Close	Index	CLOSE	Yes	MI
NASDAQCOM	NASDAQ Composite Index	Daily, Close	Index Feb 5, 1971=100	CLOSE	Yes	MI
NIKKEI225	Nikkei Stock Average, Nikkei 225	Daily, Close	Index	CLOSE	Yes	MI
WILL5000IND	Wilshire 5000 Total Market Index	Daily, Close	Index	CLOSE	Yes	Not selected
WILL5000INDFC	Wilshire 5000 Total Market Full Cap Index	Daily, Close	Index	CLOSE	Yes	Not selected
WILL5000PR	Wilshire 5000 Price Index	Daily, Close	Index	CLOSE	Yes	Not selected
WILL5000PRFC	Wilshire 5000 Full Cap Price Index	Daily, Close	Index	CLOSE	Yes	Not selected
WILLLRGCAP	Wilshire US Large-Cap Total Market Index	Daily, Close	Index	CLOSE	Yes	MI
WILLLRGCAPGR	Wilshire US Large-Cap Growth Total Market Index	Daily, Close	Index	CLOSE	Yes	MI
WILLLRGCAPVAL	Wilshire US Large-Cap Value Total Market Index	Daily, Close	Index	CLOSE	Yes	MI
WILLMICROCAP	Wilshire US Micro-Cap Total Market Index	Daily, Close	Index	CLOSE	Yes	Not selected
WILLMIDCAP	Wilshire US Mid-Cap Total Market Index	Daily, Close	Index	CLOSE	Yes	MI

WILLMIDCAPGR	Wilshire US Mid-Cap Growth Total Market Index	Daily, Close	Index	CLOSE	Yes	Not selected
WILLMIDCAPVAL	Wilshire US Mid-Cap Value Total Market Index	Daily, Close	Index	CLOSE	Yes	Not selected
WILLREITIND	Wilshire US Real Estate Investment Trust Total Market Index (Wilshire US REIT)	Daily, Close	Index	CLOSE	Yes	Not selected
WILLRESIND	Wilshire US Real Estate Securities Total Market Index (Wilshire US RESI)	Daily, Close	Index	CLOSE	Yes	Not selected
WILLSMLCAP	Wilshire US Small-Cap Total Market Index	Daily, Close	Index	CLOSE	Yes	MI
WILLSMLCAPGR	Wilshire US Small-Cap Growth Total Market Index	Daily, Close	Index	CLOSE	Yes	MI
WILLSMLCAPVAL	Wilshire US Small-Cap Value Total Market Index	Daily, Close	Index	CLOSE	Yes	Not selected
ANFCI	Chicago Fed Adjusted National Financial Conditions Index	Weekly, Ending Friday	Index	AVERAGE	#N/A	Dropped when ln() taken
FF	Effective Federal Funds Rate	Weekly, Ending Wednesday	%	AVERAGE	Yes	MI
IURSA	Insured Unemployment Rate	Weekly, Ending Saturday	%	AVERAGE	No	MI
MORTGAGE30US	30-Year Fixed Rate Mortgage Average in the United States	Weekly, Ending Thursday	%	AVERAGE	Yes	MI

MORTPTS30US	Origination Fees and Discount Points for 30-Year Fixed Rate Mortgage in the United States	Weekly, Ending Thursday	%	AVERAGE	Yes	Not selected
NFCI	Chicago Fed National Financial Conditions Index	Weekly, Ending Friday	Index	AVERAGE	#N/A	Dropped when ln() taken
NFCICREDIT	Chicago Fed National Financial Conditions Credit Subindex	Weekly, Ending Friday	Index	AVERAGE	#N/A	Dropped when ln() taken
NFCILEVERAGE	Chicago Fed National Financial Conditions Leverage Subindex	Weekly, Ending Friday	Index	AVERAGE	#N/A	Dropped when ln() taken
NFCINONFINLEVERAGE	Chicago Fed National Financial Conditions Index Nonfinancial Leveral Subindex	Weekly, Ending Friday	Index	AVERAGE	#N/A	Dropped when ln() taken
NFCIRISK	Chicago Fed National Financial Conditions Risk Subindex	Weekly, Ending Friday	Index	AVERAGE	#N/A	Dropped when ln() taken
WAAA	Moody's Seasoned Aaa Corporate Bond Yield	Weekly, Ending Friday	%	AVERAGE	Yes	Not selected
WBAA	Moody's Seasoned Baa Corporate Bond Yield	Weekly, Ending Friday	%	AVERAGE	Yes	Not selected
WGS10YR	10-Year Treasury Constant Maturity Rate	Weekly, Ending Friday	%	AVERAGE	Yes	MI

WGS1YR	1-Year Treasury Constant Maturity Rate	Weekly, Ending Friday	%	AVERAGE	Yes	MI
WGS20YR	20-Year Treasury Constant Maturity Rate	Weekly, Ending Friday	%	AVERAGE	Yes	MI
WGS2YR	2-Year Treasury Constant Maturity Rate	Weekly, Ending Friday	%	AVERAGE	Yes	MI
WGS30YR	30-Year Treasury Constant Maturity Rate	Weekly, Ending Friday	%	AVERAGE	Yes	Not selected
WGS3YR	3-Year Treasury Constant Maturity Rate	Weekly, Ending Friday	%	AVERAGE	Yes	MI
WGS5YR	5-Year Treasury Constant Maturity Rate	Weekly, Ending Friday	%	AVERAGE	Yes	MI
WGS7YR	7-Year Treasury Constant Maturity Rate	Weekly, Ending Friday	%	AVERAGE	Yes	MI
WPRIME	Bank Prime Loan Rate	Weekly, Ending Wednesday	%	AVERAGE	Yes	MI
WTB1YR	1-Year Treasury Bill: Secondary Market Rate	Weekly, Ending Friday	%	AVERAGE	Yes	Not selected
WTB3MS	3-Month Treasury Bill: Secondary Market Rate	Weekly, Ending Friday	%	AVERAGE	Yes	MI
WTB6MS	6-Month Treasury Bill: Secondary Market Rate	Weekly, Ending Friday	%	AVERAGE	Yes	MI
AOLACBW027SBOG	Other Loans and Leases: All Other Loans and Leases, All Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
AOLDCBW027SBOG	Other Loans and Leases: All Other Loans and	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI

	Leases, Domestically Chartered Commercial Banks					
AOLFRI W027SB OG	Other Loans and Leases: All Other Loans and Leases, Foreign-Related Institutions	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
BC0DCB W027SB OG	Bank Credit, Domestically Chartered Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
BC0FRI W027SB OG	Bank Credit, Foreign-Related Institutions	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	Not selected
CASACB W027SB OG	Cash Assets, All Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	Not selected
CASDC BW027S BOG	Cash Assets, Domestically Chartered Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	Not selected
CASFRI W027SB OG	Cash Assets, Foreign-Related Institutions	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	Not selected
CC4WS A	4-Week Moving Average of Continued Claims (Insured Unemployment)	Weekly, Ending Saturday	Number	SUM	No	Not selected
CCSA	Continued Claims (Insured Unemployment)	Weekly, Ending Saturday	Number	SUM	No	MI
CILDCB W027SB OG	Commercial and Industrial Loans, Domestically Chartered	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI

	Commercial Banks					
CILFRIW027SBOG	Commercial and Industrial Loans, Foreign-Related Institutions	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	Not selected
CLSACBW027SBOG	Consumer Loans, All Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
CLSDCBW027SBOG	Consumer Loans, Domestically Chartered Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
COVEMP	Covered Employment	Weekly, Ending Saturday	Number	SUM	Yes	Not selected
DPSACBW027SBOG	Deposits, All Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	Not selected
DPSDCBW027SBOG	Deposits, Domestically Chartered Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
DPSFRIW027SBOG	Deposits, Foreign-Related Institutions	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
DPSSCBW027SBOG	Deposits, Small Domestically Chartered Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
H8B3053NCBA	Other Assets, All Commercial Banks	Weekly	Mil. of U.S. Dollars	SUM	Yes	MI
H8B3053NDMA	Other Assets, Domestically Chartered Commercial Banks	Weekly	Mil. of U.S. Dollars	SUM	Yes	MI
H8B3053NFRA	Other Assets, Foreign-Related Institutions	Weekly	Mil. of U.S. $	SUM	Yes	MI

H8B3094NCBA	Borrowings, All Commercial Banks	Weekly	Mil. of U.S. Dollars	SUM	Yes	MI
H8B3094NDMA	Borrowings, Domestically Chartered Commercial Banks	Weekly	Mil. of U.S. Dollars	SUM	Yes	MI
H8B3094NFRA	Borrowings, Foreign-Related Institutions	Weekly	Mil. of U.S. $	SUM	Yes	MI
H8B3095NCBA	Other Liabilities, All Commercial Banks	Weekly	Mil. of U.S. Dollars	SUM	Yes	MI
H8B3095NDMA	Other Liabilities, Domestically Chartered Commercial Banks	Weekly	Mil. of U.S. Dollars	SUM	Yes	MI
H8B3095NFRA	Other Liabilities, Foreign-Related Institutions	Weekly	Mil. of U.S. $	SUM	Yes	MI
IC4WSA	4-Week Moving Average of Initial Claims	Weekly, Ending Saturday	Number	SUM	No	MI
ICSA	Initial Claims	Weekly, Ending Saturday	Number	SUM	No	MI
LCBACBW027SBOG	Loans to Commercial Banks, All Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
LCBDCBW027SBOG	Loans to Commercial Banks, Domestically Chartered Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
LCBFRIW027SBOG	Loans to Commercial Banks, Foreign-Related Institutions	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	#N/A	Dropped when ln() taken

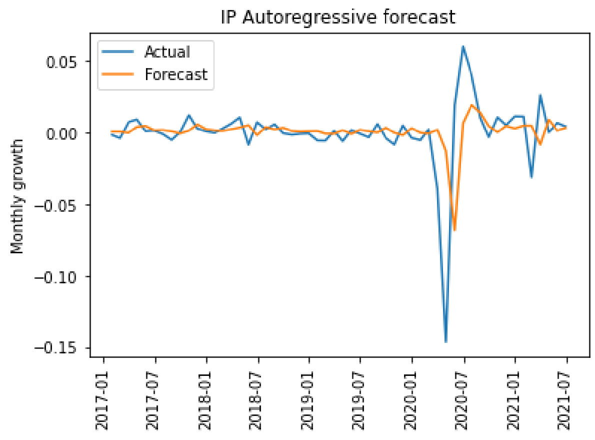
LLBDCBW027SBOG	Loans and Leases in Bank Credit, Domestically Chartered Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
LLBFRIW027SBOG	Loans and Leases in Bank Credit, Foreign-Related Institutions	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
LTDACBW027SBOG	Large Time Deposits, All Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
LTDDCBW027SBOG	Large Time Deposits, Domestically Chartered Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
LTDFRIW027SBOG	Large Time Deposits, Foreign-Related Institutions	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
NDFFRIW027SBOG	Net Due to Related Foreign Offices, Foreign-Related Institutions	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	#N/A	Dropped when ln() taken
ODSFRIW027SBOG	Other Deposits, Foreign-Related Institutions	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
OSEACBW027SBOG	Other Securities, All Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
OSEDCBW027SBOG	Other Securities, Domestically Chartered Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
OSEFRIW027SBOG	Other Securities, Foreign-	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI

	Related Institutions					
RALACBW027SBOG	Residual (Assets Less Liabilities), All Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
RALDCBW027SBOG	Residual (Assets Less Liabilities), Domestically Chartered Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
RALFRIW027SBOG	Residual (Assets Less Liabilities), Foreign-Related Institutions	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	#N/A	Dropped when ln() taken
RELACBW027SBOG	Real Estate Loans, All Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	No	MI
RELDCBW027SBOG	Real Estate Loans, Domestically Chartered Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
SBCACBW027SBOG	Securities in Bank Credit, All Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
SBCDCBW027SBOG	Securities in Bank Credit, Domestically Chartered Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
SBCFRIW027SBOG	Securities in Bank Credit, Foreign-Related Institutions	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	No	MI
TASACBW027SBOG	Treasury and Agency Securities, All Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI

TASDCBW027SBOG	Treasury and Agency Securities, Domestically Chartered Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
TASFRIW027SBOG	Treasury and Agency Securities, Foreign-Related Institutions	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
TLAACBW027SBOG	Total Assets, All Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
TLADCBW027SBOG	Total Assets, Domestically Chartered Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
TLAFRIW027SBOG	Total Assets, Foreign-Related Institutions	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
TLBACBW027SBOG	Total Liabilities, All Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
TLBDCBW027SBOG	Total Liabilities, Domestically Chartered Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
TLBFRIW027SBOG	Total Liabilities, Foreign-Related Institutions	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
TOTBKCR	Bank Credit, All Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
TOTCI	Commercial and Industrial Loans, All Commercial Banks	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI
TOTLL	Loans and Leases in Bank Credit, All	Weekly, Ending Wednesday	Bil. of U.S. $	SUM	Yes	MI

	Commercial Banks					

Chart A1 – IP autoregressive forecast results



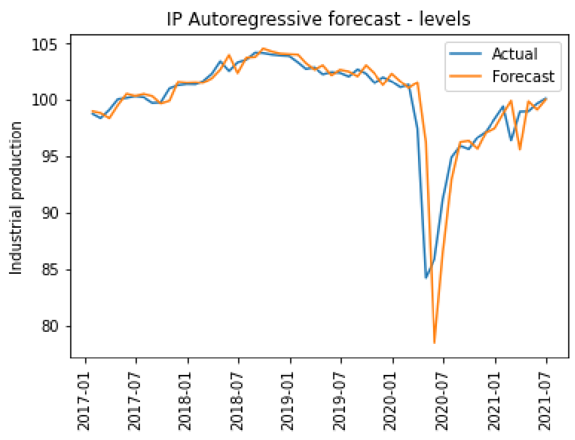
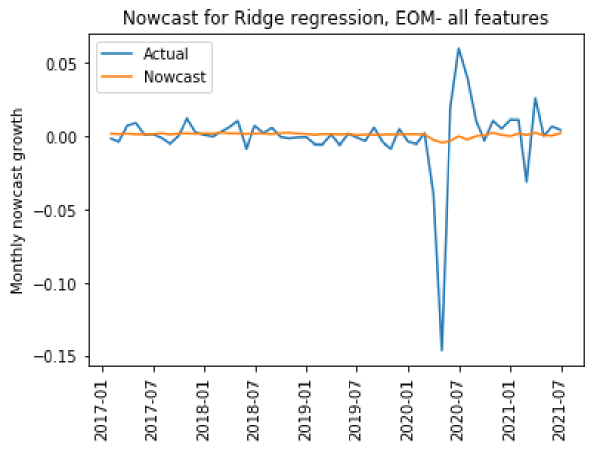


Chart A2 – Nowcast results – Ridge regression, All features, End of month



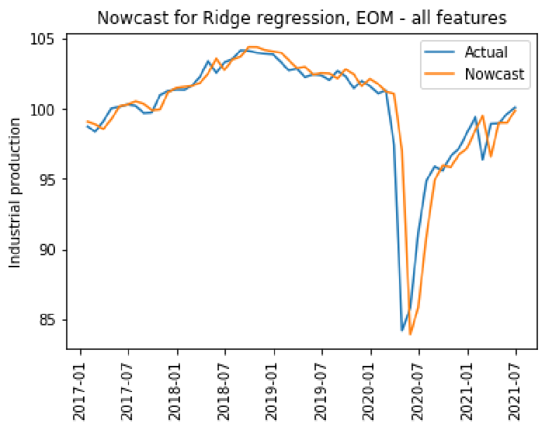
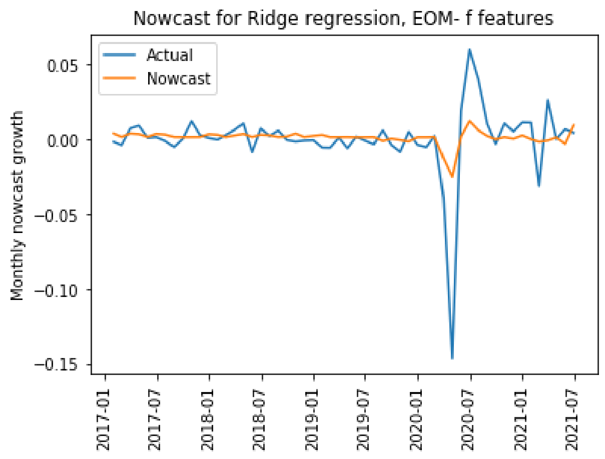


Chart A3 – Nowcast results – Ridge regression, F features, End of month



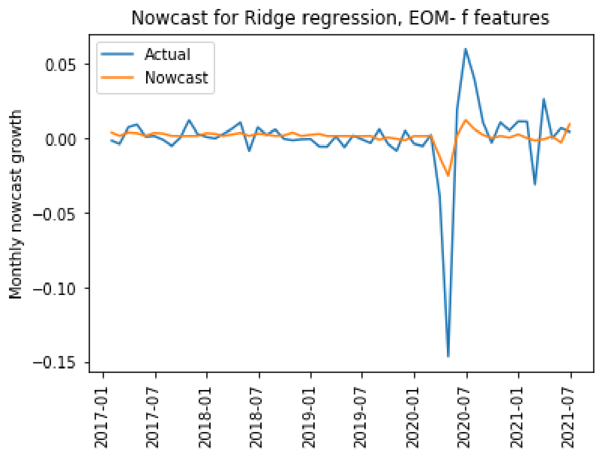


Chart A4 – Nowcast results – Ridge regression, MI features, End of month

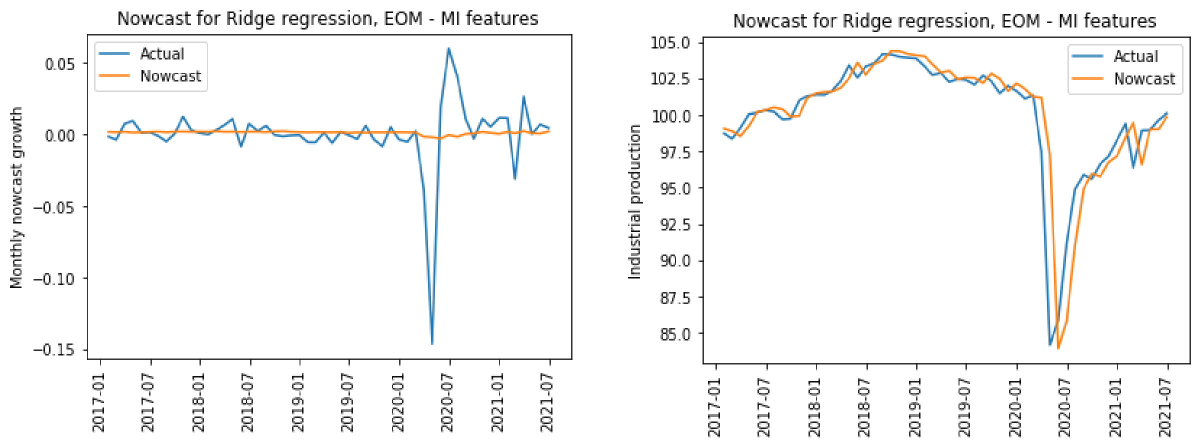


Chart A5 – Nowcast results – Lasso regression, All features, End of month

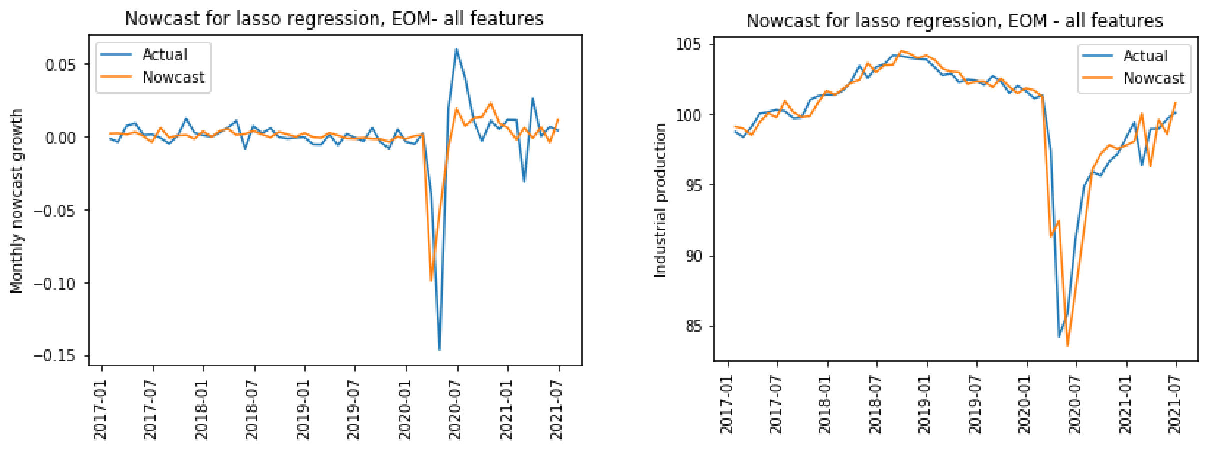


Chart A6 – Nowcast results – Lasso regression, F features, End of month

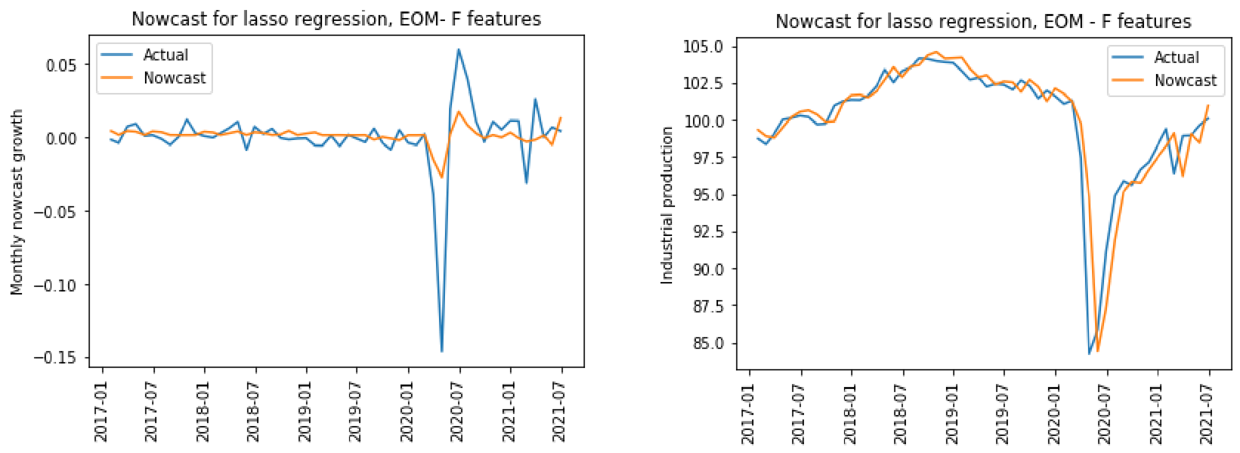
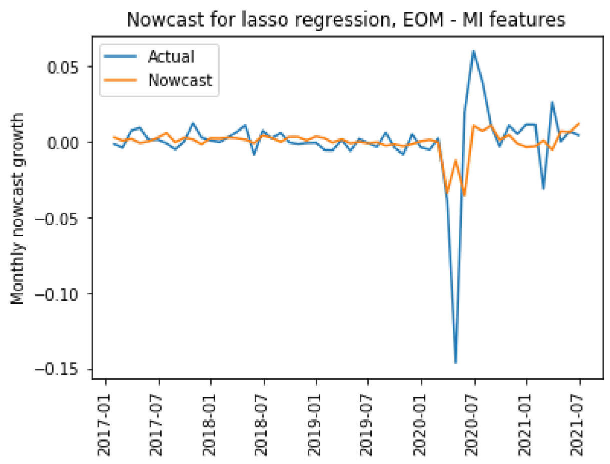


Chart A7 – Nowcast results – Lasso regression, MI features, End of month



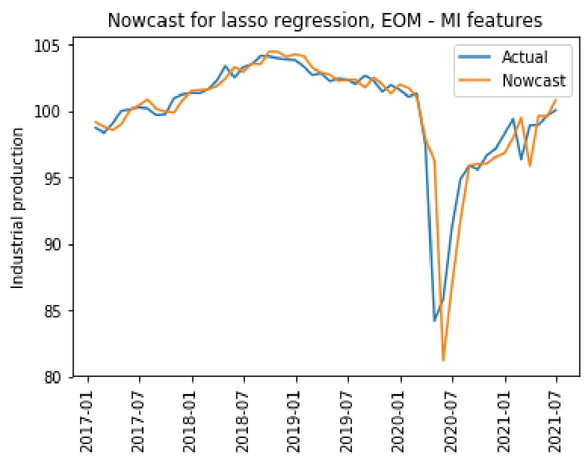
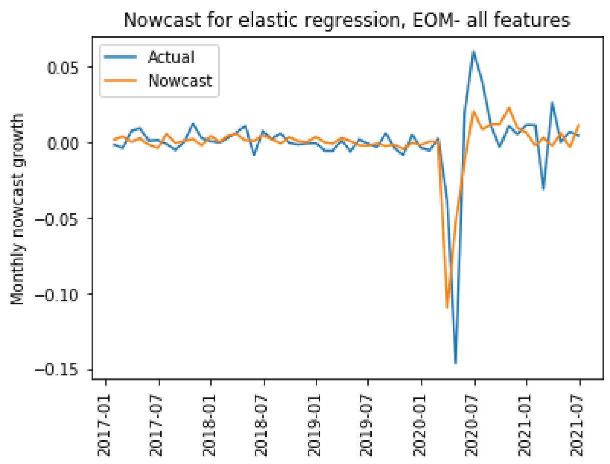


Chart A8 – Nowcast results – Elastic net regression, All features, End of month



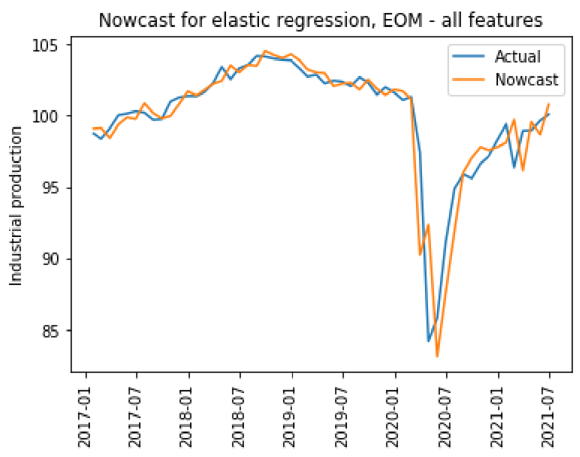
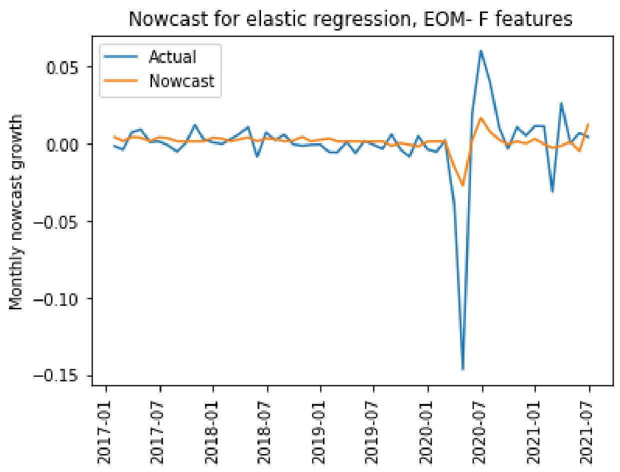


Chart A9 – Nowcast results – Elastic net regression, F features, End of month



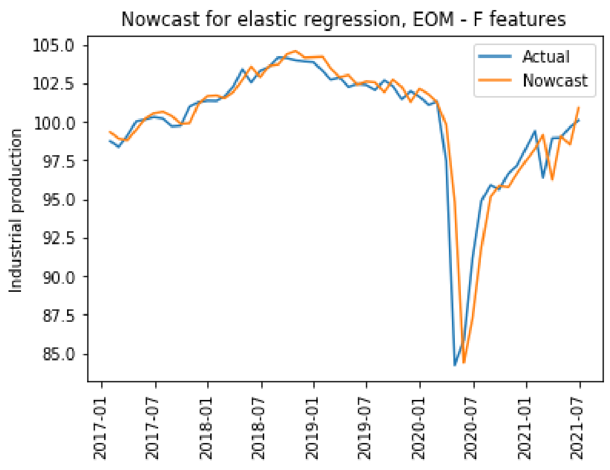


Chart A10 – Nowcast results – Elastic net regression, MI features, End of month

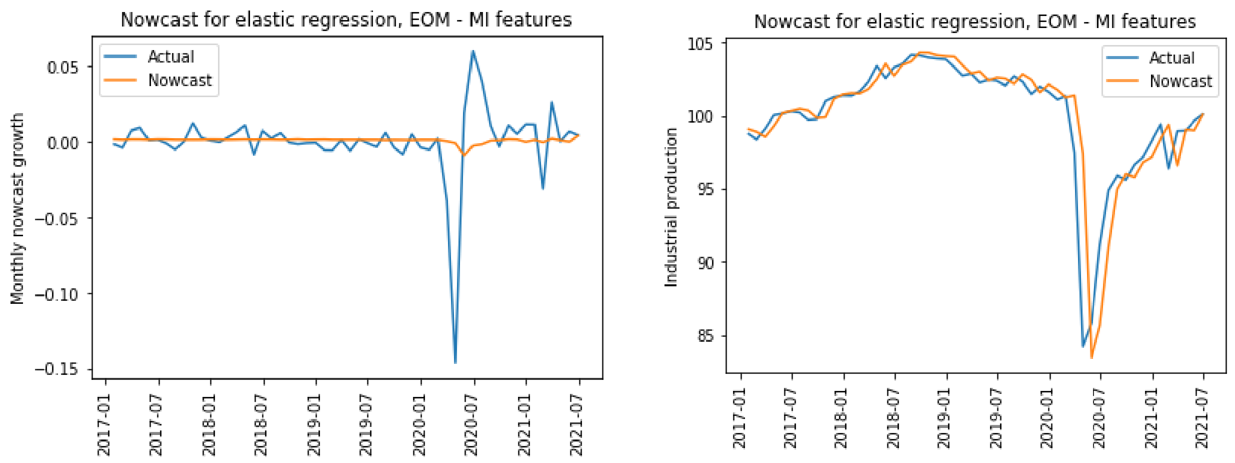


Chart A11 – Nowcast results – ANN, All features, End of month

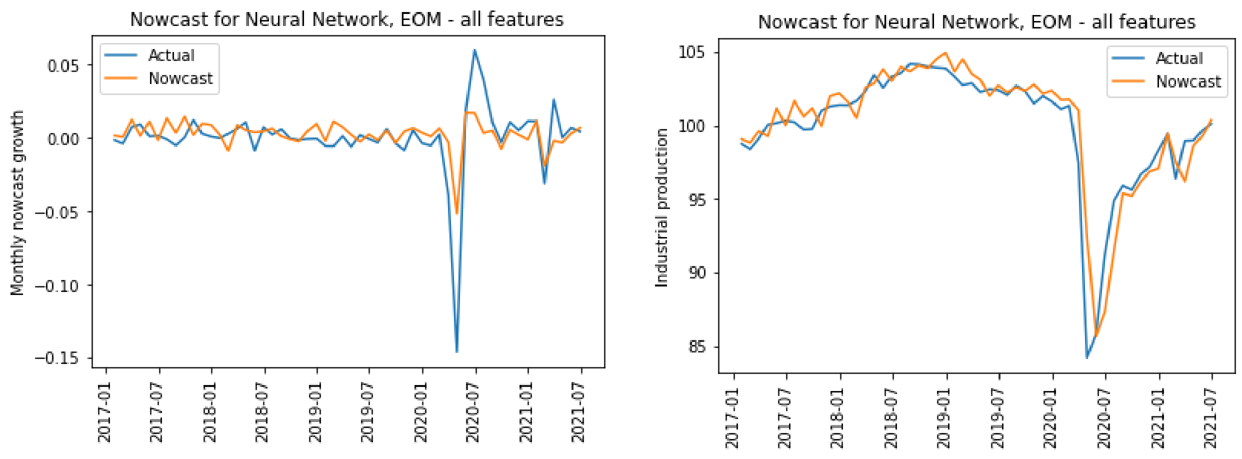


Chart A12 – Nowcast results – ANN, F features, End of month

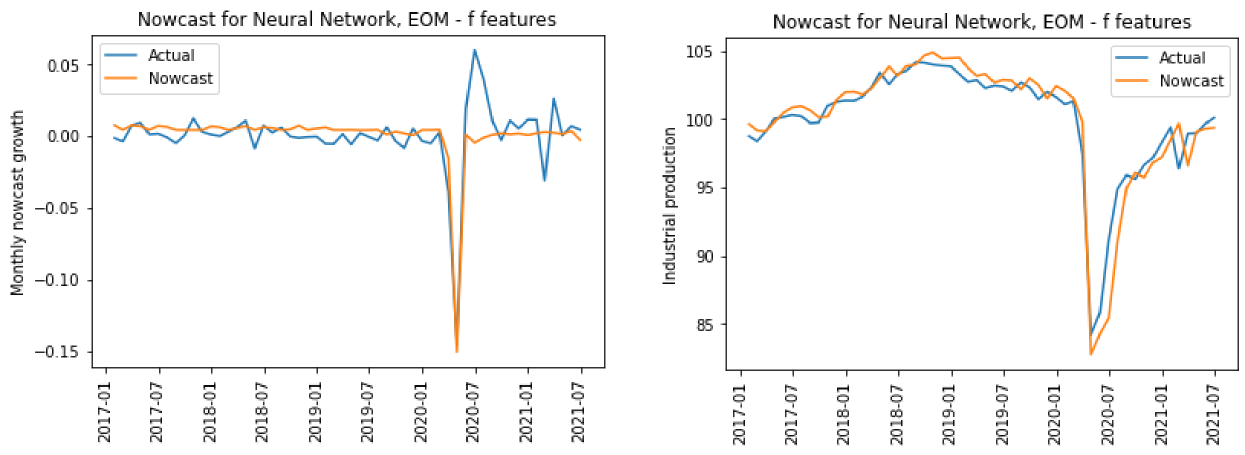
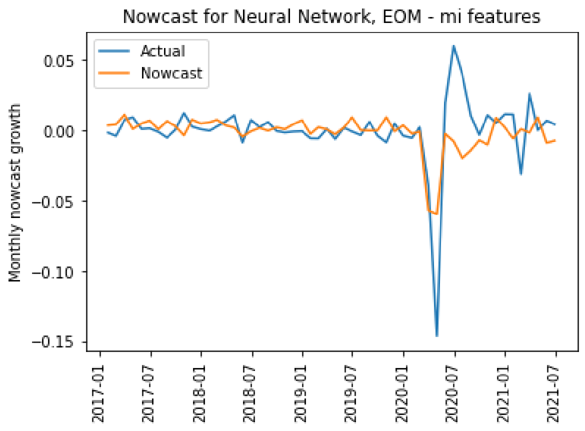


Chart A13– Nowcast results – ANN, MI features, End of month



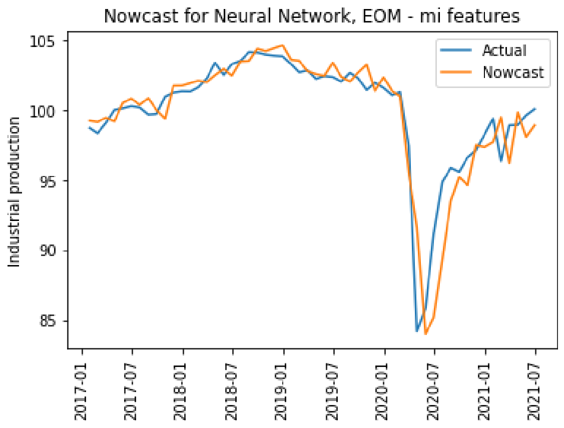
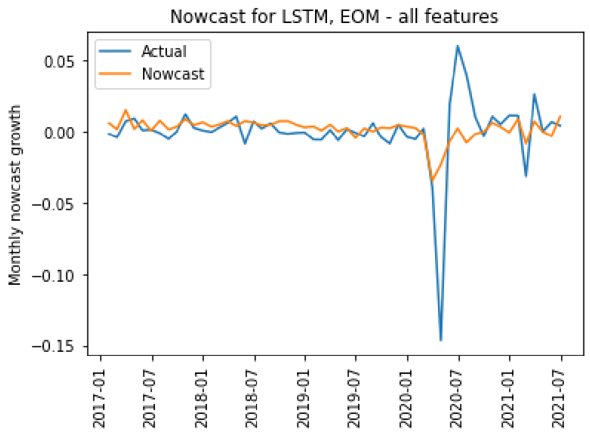


Chart A14 – Nowcast results – LSTM, All features, End of month



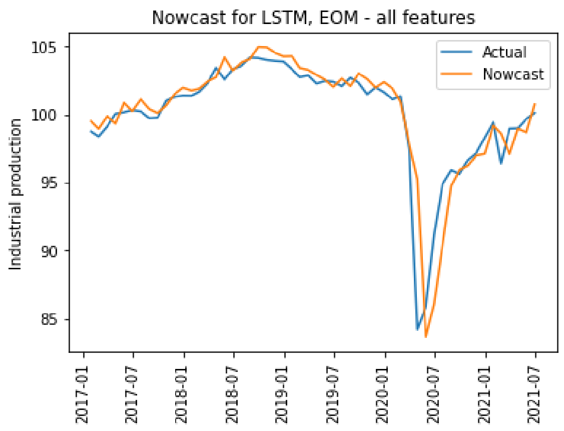
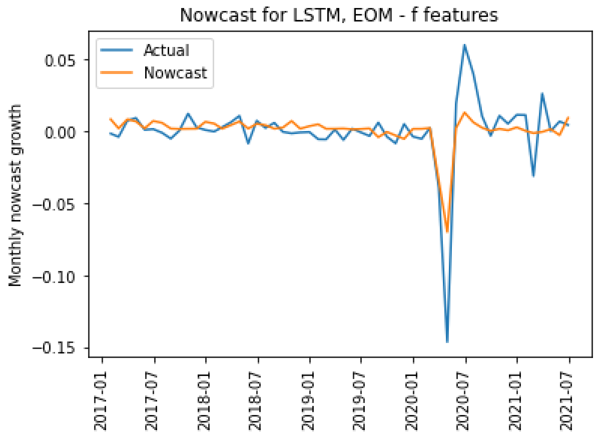


Chart A15 – Nowcast results – LSTM, F features, End of month



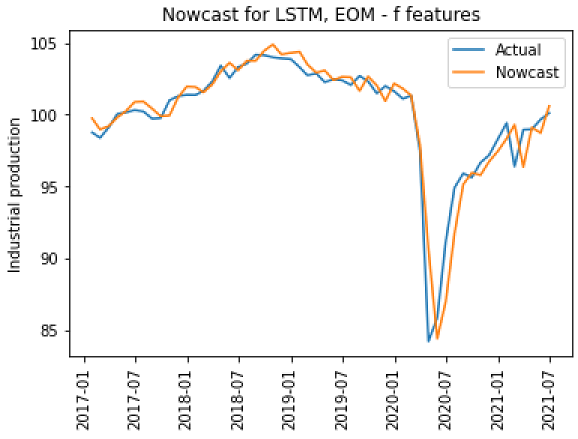


Chart A16– Nowcast results – LSTM, MI features, End of month

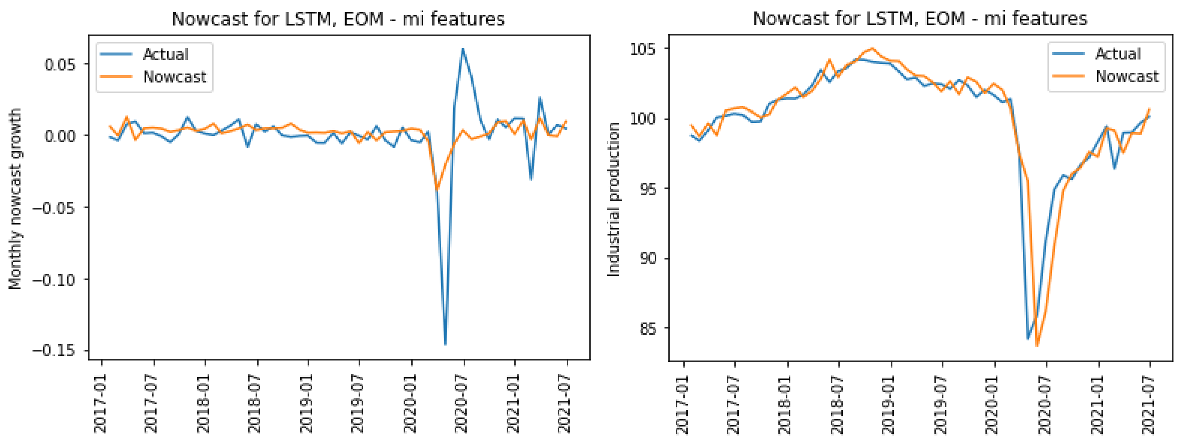


Chart A17 – Nowcast results – Ridge regression, All features, From 15th of month

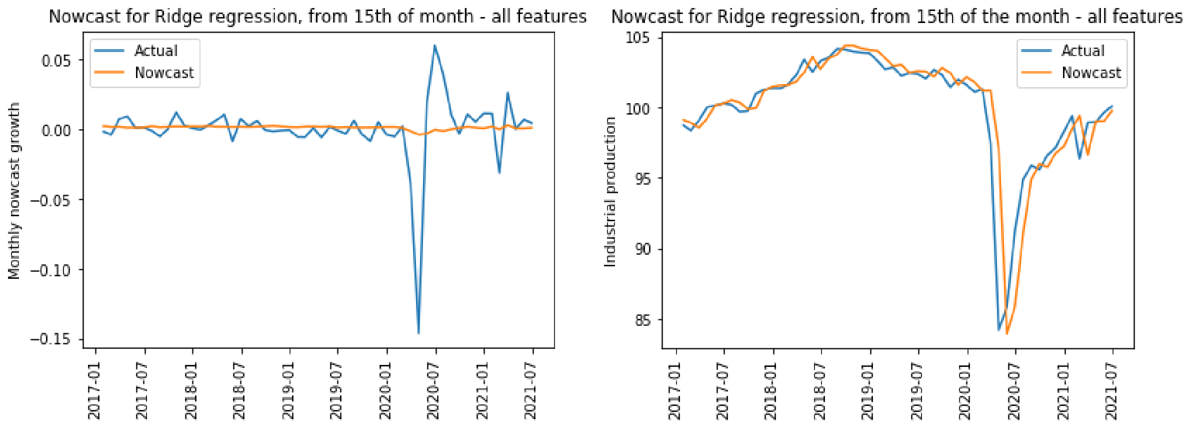


Chart A18 – Nowcast results – Ridge regression, F features, From 15th of month

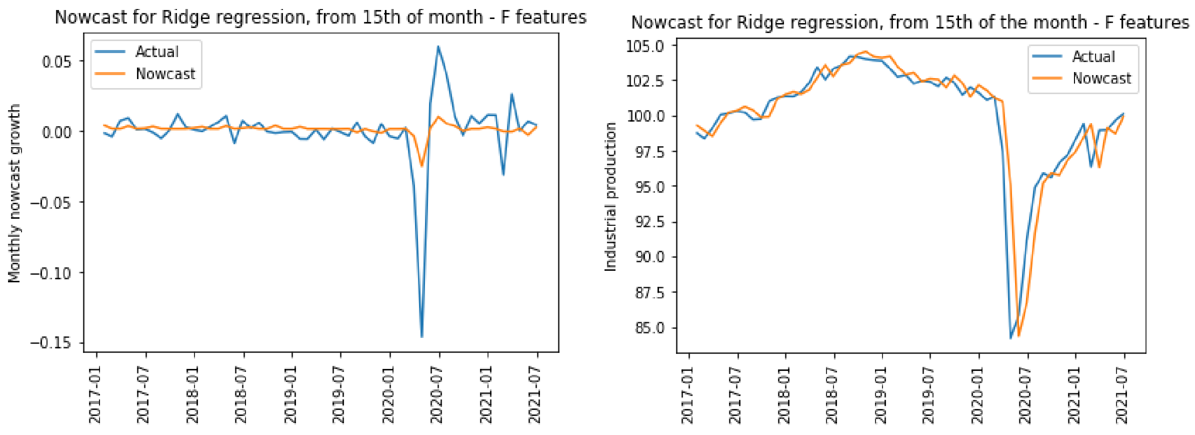
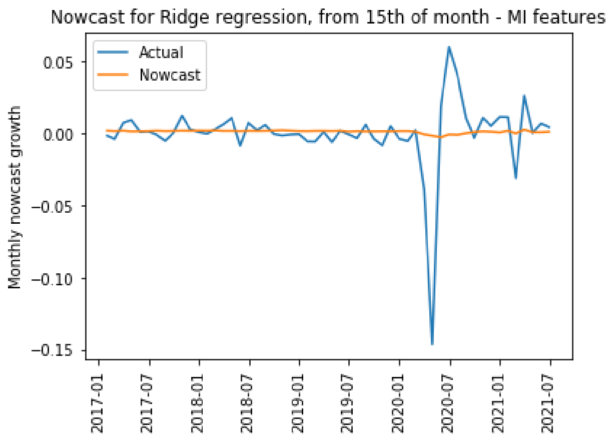


Chart A19 – Nowcast results – Ridge regression, MI features, From 15th of month



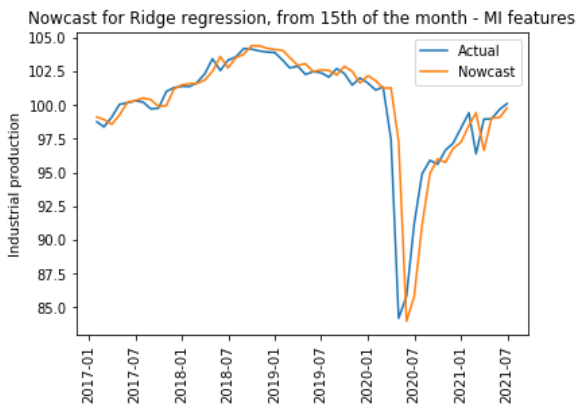
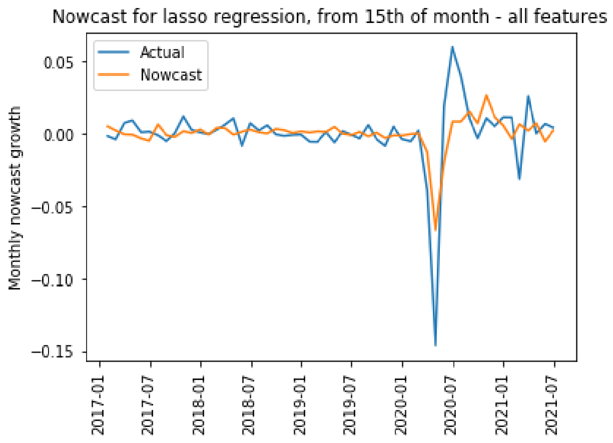


Chart A20 – Nowcast results – Lasso regression, All features, From 15th of month



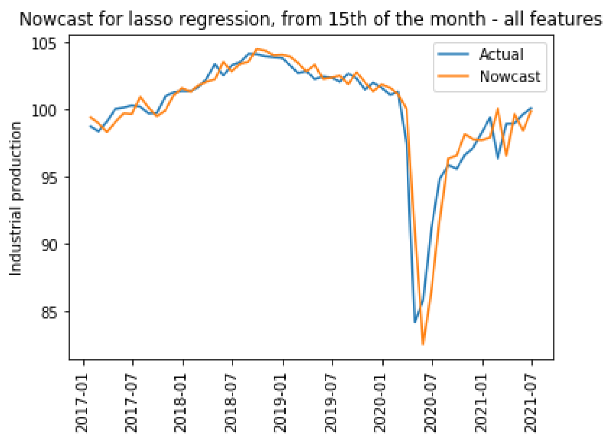
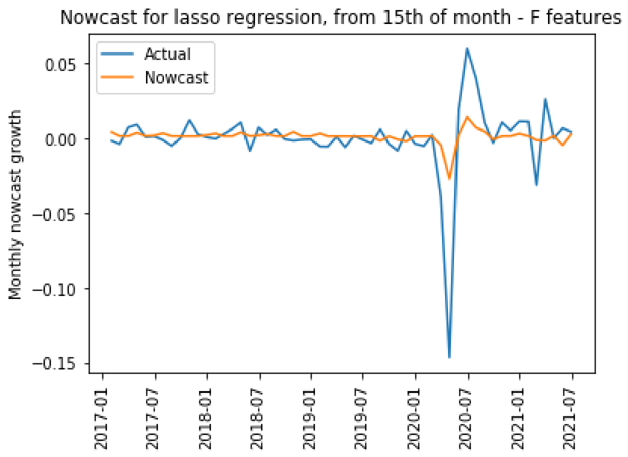


Chart A21 – Nowcast results – Lasso regression, F features, From 15th of month



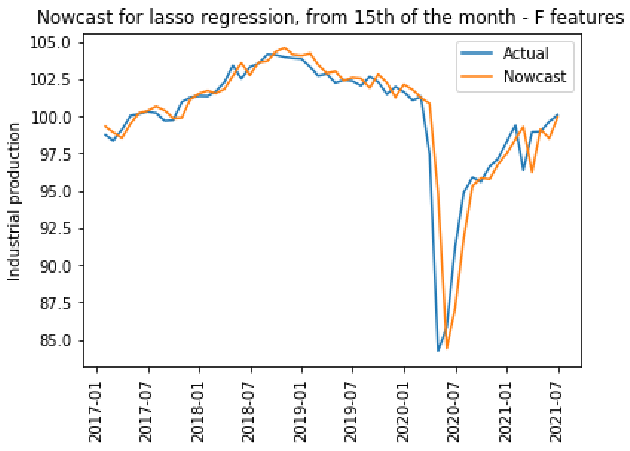


Chart A22 – Nowcast results – Lasso regression, MI features, From 15th of month

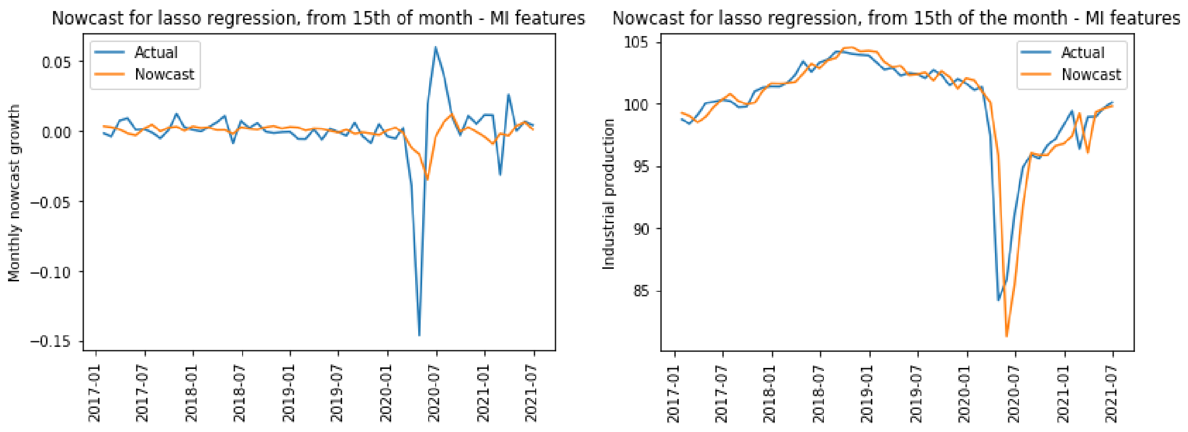


Chart A23 – Nowcast results – Elastic net regression, All features, From 15th of month

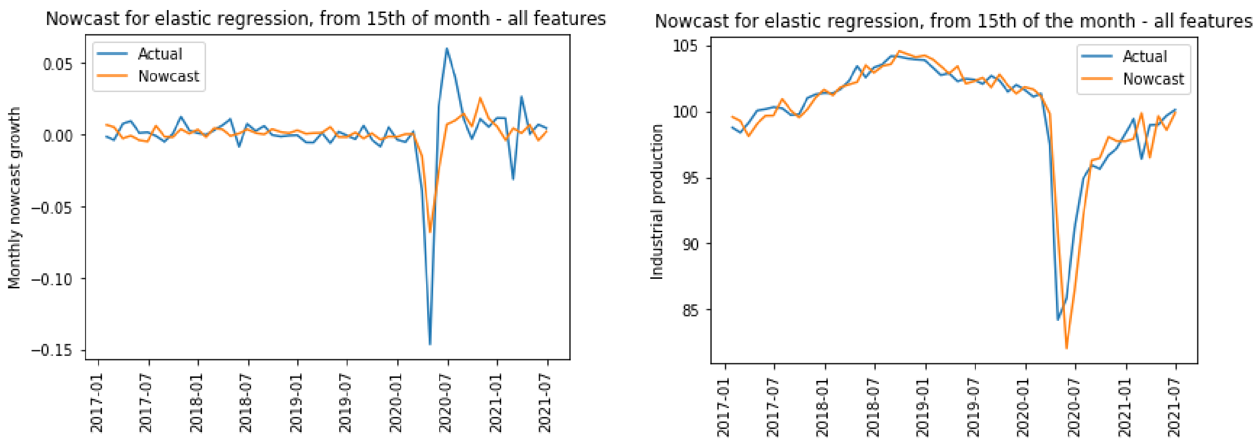


Chart A24 – Nowcast results – Elastic net regression, F features, From 15th of month

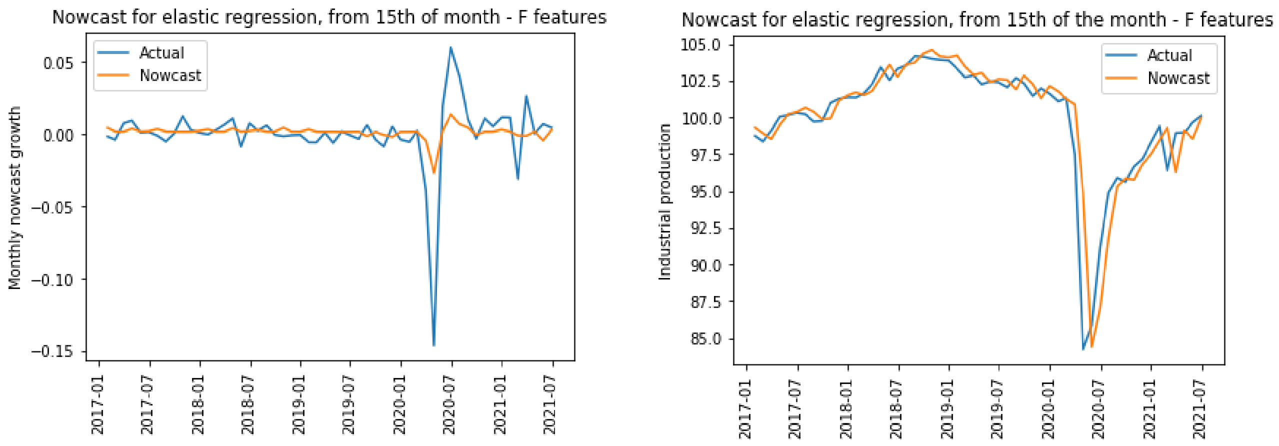


Chart A25 – Nowcast results – Elastic net regression, MI features, From 15th of month

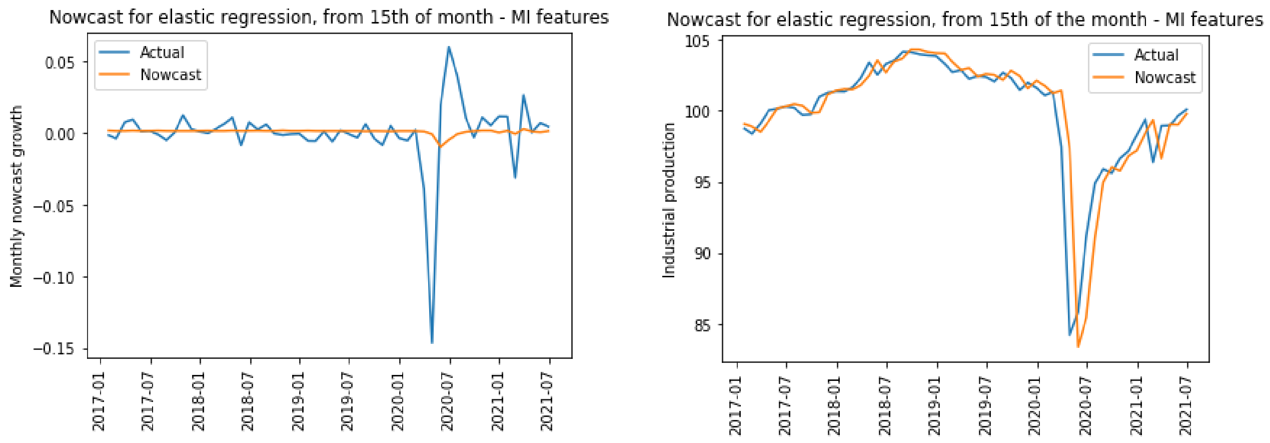


Chart A26 – Nowcast results – ANN, All features, From 15th of month

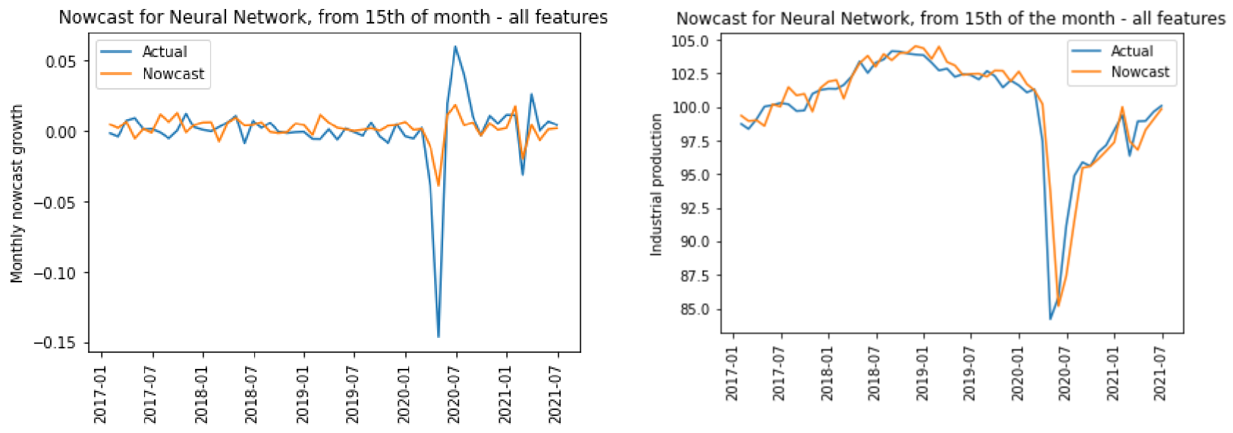


Chart A27 – Nowcast results – ANN, F features, From 15th of month

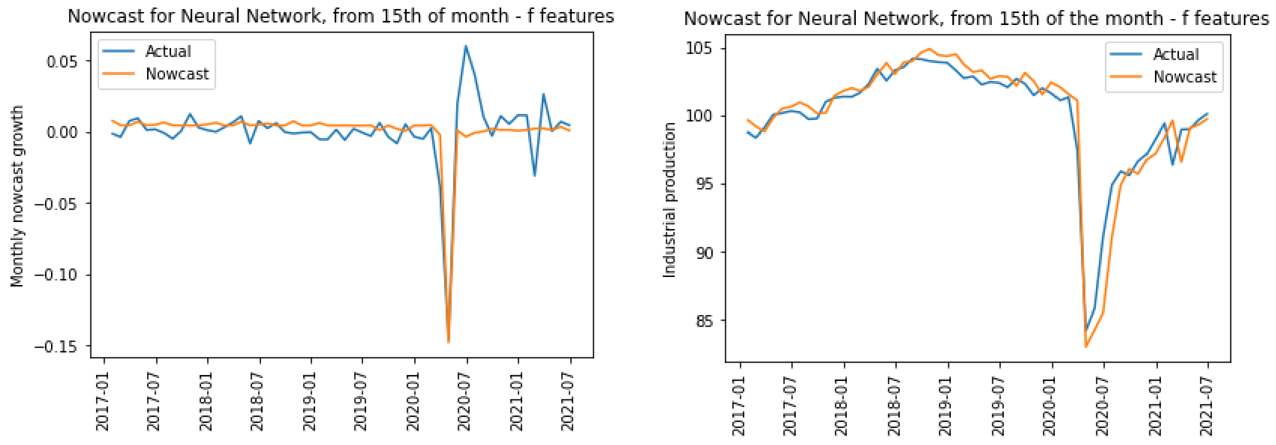
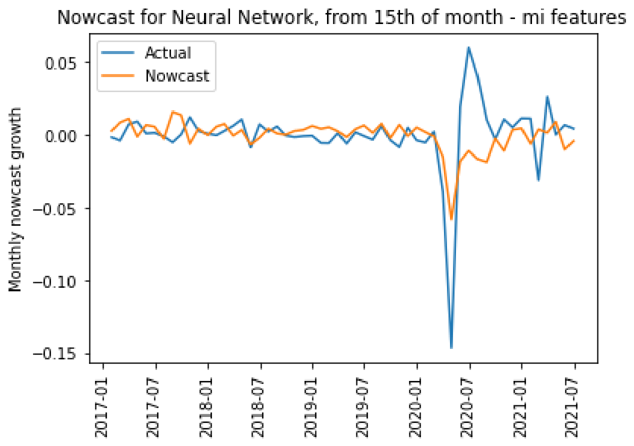


Chart A28– Nowcast results – ANN, MI features, From 15th of month



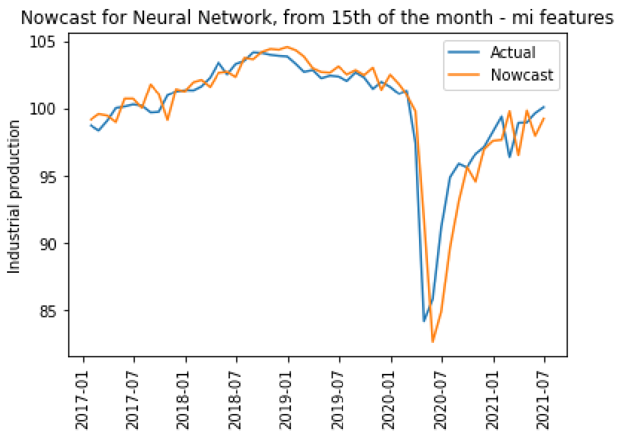
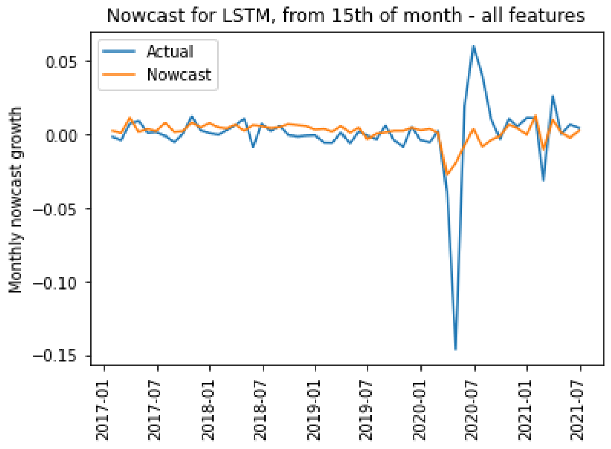


Chart A29 – Nowcast results – LSTM, All features, From 15th of month



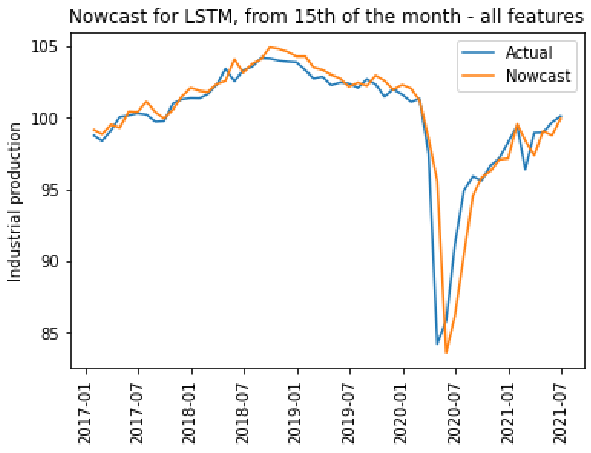
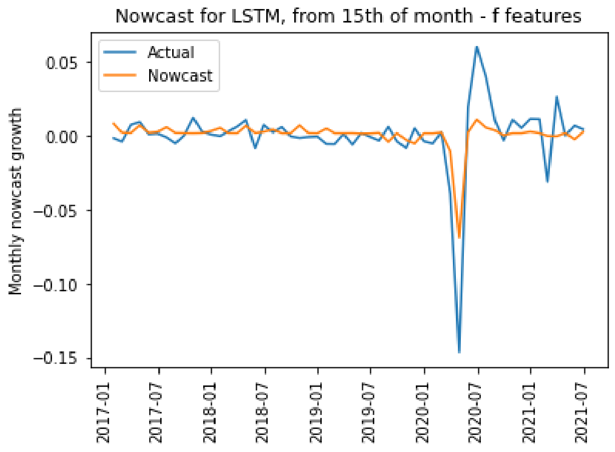


Chart A30 – Nowcast results – LSTM, F features, From 15th of month



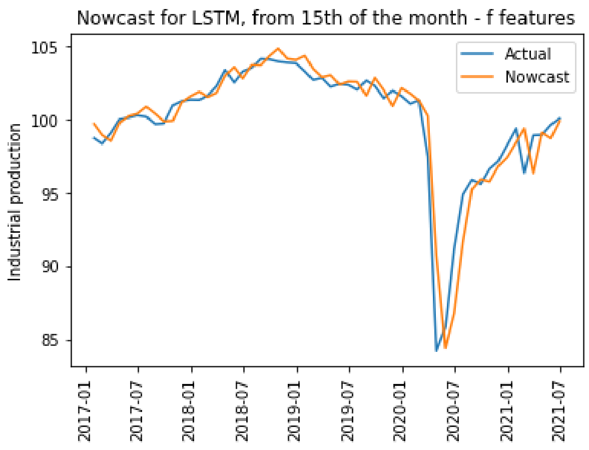
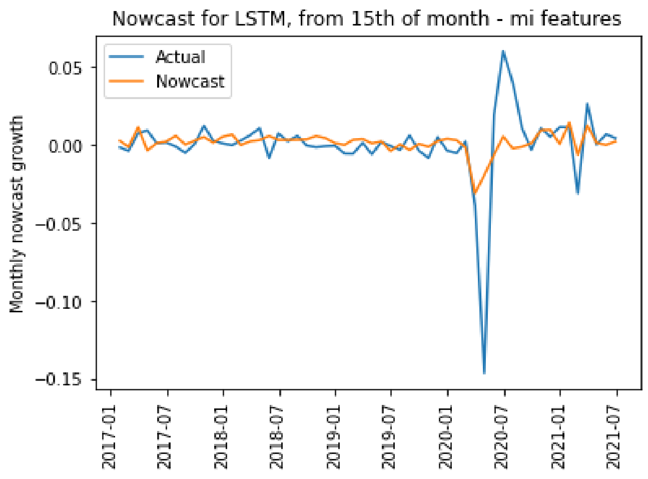
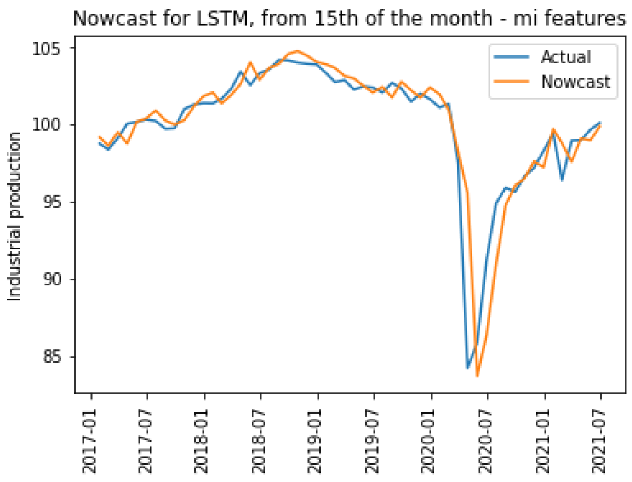


Chart A31 – Nowcast results – LSTM, MI features, From 15th of month


Nowcast for LSTM, from 15th of month - mi features
Actual
Nowcast
Monthly nowcast growth


Nowcast for LSTM, from 15th of the month - mi features
Actual
Nowcast
Industrial production