NOWCASTING U.S. INFLATION USING MIXED FREQUENCY REAL-TIME DATA

GUSTAF LUNDGREN, NILS WICKTOR

Master's thesis
2023:E38

LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Mathematical Statistics

CENTRUM SCIENTIARUM MATHEMATICARUM

Master's Theses in Mathematical Sciences 2023:E38
ISSN 1404-6342

LUTFMS-3478-2023

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lu.se/

Nowcasting U.S. inflation using mixed frequency real-time data

Lund University

Gustaf Lundgren
Nils Wicktor

May 2023

Abstract

Different models were developed with the aim of nowcasting inflation at a daily basis with high frequency variables, while using real-time data to avoid look ahead bias. Both popular machine learning models such as Random Forest and XGBoost, and more traditional models such as UMIDAS and Almon distributed lag models were used to make the nowcasts. The MIDAS framework was utilized as a way of handling predictors sampled at mixed frequencies and variations of LASSO were used to select the best variables and features for the model.

The main analysis considers the performance of the models compared to each other and a simple benchmark AR(1) model. It can be concluded that the ML models outperform all other models, with XGboost at the top. UMIDAS and Almon were slightly outperformed by the AR(1) model which could probably be explained by overparametrization and that the LASSO did not do a good enough job to remove enough features. Further, other topics related to the nowcasting of inflation was investigated. It was concluded that inflation has become harder to nowcast during the recent years. However the variables used have stayed relatively constant throughout time. The inclusion of higher frequency variables, such as daily, improved the nowcasts compared to the more traditional approach of only using monthly released macro variables.

Keywords: *Inflation, Machine Learning, Nowcasting, MIDAS, Almon distributed lag models, Real-Time data, Random Forest, XGBoost*

Acknowledgements

We would like to thank our academic supervisor Erik Lindström at the Centre for Mathematical Sciences at Lund University for his guidance and encouragement. We would also like to thank our supervisor Jonas Bergroth, and all other people at Erik Penser Bank for their helpful insights.

Definitions and acronyms

The following is a list of useful acronyms that will be used throughout the paper.

- **AGL** - Adaptive Group LASSO
- **AL** - Adaptive LASSO
- **ASGL** - Adaptive Sparse Group Lasso
- **GL** - Group Lasso
- **LASSO** - Least Absolute Shrinkage and Selection Operator
- **MIDAS** - Mixed Data Sampling
- **MoM** - Month-on-month
- **OLS** - Ordinary Least Squares
- **PCA** - Principal Component Analysis
- **YoY** - Year-on-year

The following are useful definitions and terms used throughout the paper.

- **Flow variable** - A variable whose value that referees to a period of time, e.g. Industrial production

- **Horizon** - The number of days until the next inflation number is released
- **Lag** - Value from a previous timepoint
- **Nowcasting** - Nowcasting is a contraction of now and forecasting, meaning forecasting the present or near future
- **Observation date** - A simplification of observation period, only referring to the first date in the observation period
- **Observation period** - Referring to the period for which the data applies
- **Release date** - When the data is released and made available
- **Stock variable** - A variable whose value that referees to a specific point in time, e.g. Oil price
- **Variable selection** - The act of reducing the number of variables by selecting the most important ones by some criterion

Contents

List of Figures

Chapter 1

Introduction

Accurately assessing the current economic situation is crucial for predicting the future and ensuring that the right actions are taken. One important measurement to keep track of is inflation. Central banks are tasked with ensuring price stability, and therefore rely on monitoring inflation to design policies and adjust yields to ensure stability. On the other hand, market actors use it to predict the markets' direction and to adjust their trading strategies.

A challenge when keeping track of inflation and the consumer price index is the publication lag. The delay in publishing means that a month's official inflation is not available until approximately two weeks after the month is over. If this information was available sooner, the right actions could be taken earlier, both from the perspective of both market actors and central banks.

During a target month, several data points containing valuable information regarding consumer prices are released. Some are released daily, others weekly, and some are monthly variables released on different dates spread throughout the month. By using the information in these data points, it is

possible to estimate the inflation in real-time by nowcasting. Nowcasting is a concatenation of *now* and *forecasting*, and describes the act of forecasting the present or near future.

There are a variety of approaches for dealing with the nowcasting challenge. The approach chosen in this master's thesis is a combination of variable selection using *Least Absolute Shrinkage and Selection Operator* (LASSO) and *Mixed Data Sampling* (MIDAS), in an attempt to take advantage of data released at different frequencies. The model will use a large set of predictive series and choose which are to be included. A main feature of the model is that it can deal with real-time data and can therefore be used daily to create new estimates.

1.1 Nowcasting

The problem of nowcasting is formulated as follows:

$$\hat{y}_{t|\tau} = E[y_t|\Omega_\tau], \tag{1.1}$$

which in words means the expected value of a variable at observation time t, given available information at evaluation date τ, denoted Ω_τ. Here τ is assumed to be close to t in time, in the sense that we are interested in short-term nowcasts rather than long-term equilibrium relations.

We aim to provide an accurate estimation of inflation, from the perspective of a market actor, by nowcasting the official release value. This is the value that the market watches and reacts to. Our nowcasting approach involves estimating the inflation until the official number is released, after which we repeat the process for the subsequent release. For instance, when we estimate the inflation of January 2023, we begin estimating it on January 12-*th* (when December's inflation is released) and stop estimating it on February 14-*th*

(when January's inflation is released). Technically it is not nowcasting when estimating January's inflation in February since it is something that has already happened, but just not been released. For consistency throughout the thesis, the process of estimating inflation between official releases will be denoted as nowcasting.

1.2 Consumer price index and inflation

Consumer price index (CPI) is an index which tracks the average price urban customers pay for a market basket of consumer goods and services. Measurements of prices are done monthly by the Bureau of Labor Statistics in the U.S. Measurements begin the first day of the month and end the last day of the month. The market basket consists of 12 categories, which are the following: Food & non-alcoholic beverages, Alcohol & tobacco, Clothing & footwear, Housing & household services, Furniture & household goods, Health, Transport, Communication, Recreation & culture, Education, Restaurants & hotels, Miscellaneous goods & services. Some prices are measured by conducting surveys, others by observing prices in stores. Changes in the CPI is therefore a representation of the change in cost of living in the U.S. The Bureau of Labor Statistics began measuring CPI in 1913 and the period between 1982 and 1984 is set to the index level of 100. (Bureau of Labor Statistics, 2023b)

Inflation is defined as the change in the CPI from the previous period, which can be previous month, previous quarter or previous year. The most common choice is to measure inflation on a yearly basis. For example, say that the CPI for January 2022 is 500 and the CPI for January 2023 is 550. This means that the inflation on a year-on-year basis in January 2023 is 10 percent.

Since some items in the CPI basket can be volatile, for example energy and food, there exists sub-indexes where these are excluded. There are also

sub-indexes for different population groups. CPI-U covers urban customer, which make up approximately 93 percent of the U.S. population. CPI-W on the other hand, covers urban wage earners and clerical workers, which make up 29 percent of the population. The CPI measurement of interest in this master's thesis is the CPI for all urban consumers and all items, since it is the broadest and the one most actors focus on. For more information about how CPI is measured and its sub-indexes, see Bureau of Labor Statistics (2023a).

1.3 Purpose

Inflation is, as mentioned before, a key variable for both central banks and market actors to keep track of as it can provide an overview of the current state of the economy. In recent times U.S. inflation has become both harder and easier to nowcast, depending on how you view it. Stock and Watson (2006) shows that the inflation is less volatile today than it was during the late 20th century, making it easier to forecast. But at the same time it is shown that commonly used models are inferior to the naïve 12-month average inflation, indicating inflation is becoming harder to capture in standard models such as the backward-looking Phillips curve. The Phillips curve was one of the first models used to model inflation, see e.g. Phelps (1967) for more information about the Phillips curve. In the last two years, the inflation in the U.S. and other parts of the world has risen sharply and affected the global economy. The U.S. inflation which in the beginning was commented on as transitory by leading experts proved to stick around, proving once more how hard it is to capture the inflation. (Tretina, 2022)

Our goal is to nowcast the inflation from the perspective of a market actor. This means estimating on a short horizon and focusing on predicting the next inflation number that is to be released. The model should be able to create nowcast at any given horizon and select variables to use from a large data set with real time data.

The MIDAS framework has previously mostly been used for predicting Gross Domestic Product (GDP), see Kitchen and Monaco (2003) or Kuzin et al. (2011) for examples. But the MIDAS framework has also been used for predicting inflation with monthly and daily variables as input. Often a lag structure with few parameters, like exponential Almon, is enforced on the lags to simplify estimation of the parameters. An alternative method of reducing how many parameters that need estimation is to use feature selection, for example sparse LASSO.

Our contribution to the previous work is combining variable selection using adaptive sparse group LASSO and the MIDAS framework with data of different frequencies and models. Further, the nowcasts will be produced on a daily basis which is uncommon in previous papers, where only a few horizons are usually selected.

To measure the effectiveness of our proposed solution to the nowcasting challenge, two key questions must be answered. These are presented below.

Are the models successful in estimating the release value of the U.S. inflation evaluated at *daily frequency*? This is evaluated in the model's performance relative to a simple AR model. It is also evaluated in observing how the model updates its estimate as more information is made available closer to the release date. By examining this we can also answer the question regarding how well standard regression models stand up against the two very popular machine learning models XGBoost (XGB) and Random Forest (RF). When investigating the model's performance, it is also interesting to evaluate if they perform differently depending on the horizon and how much data is available. It is also interesting to see the effect of tuning the hyperparameters for the machine learning models.

What variables are important when nowcasting inflation, and has this changed? By using algorithms to select variables it is possible to see which ones are frequently selected and thus more important for the now-

casts. The frequency of which the variables are selected will be compared over different periods to determine if this has changed over time. Another topic related to this which can be investigated is if the addition of higher frequency variables is necessary for the model. If higher frequency variables such as daily and weekly are not included after variable and feature selection is performed, there is no purpose in using the MIDAS framework since its sole purpose is to combine mixed frequency data. Therefore, the importance of daily and weekly variables is evaluated to see if the model structure could be simplified.

1.4 Previous work

1.4.1 Nowcasting inflation

Nowcasting is not a new concept, and has been used for a long time within areas such as meteorology (Mass et al., 2011). In more recent times however, the concept has been applied in economics (Giannone et al., 2008) to nowcast variables such as GDP or inflation. This has resulted in a wide range of approaches and models ranging from simple bridge equations, to more complex machine learning algorithms. GDP is the most common variable to nowcast. This is mainly due to the fact that it is released on a quarterly basis, allowing to use monthly released macroeconomic variables in the nowcasting approach. The topic of this thesis, nowcasting inflation, is also popular. However, as opposed to GDP, inflation is released on a monthly basis and the use of higher frequency variables is therefore more common (Modugno, 2013). Other attempts at nowcasting inflation have been made by Knotek II and Zaman (2022), Knotek and Zaman (2017), Clark et al. (2022), Xu et al. (2019) and Monteforte and Moretti (2013), along with many others.

1.4.2 Commonly used models for nowcasting

Below is a brief introduction of some models, beyond the already mentioned Philips curve, that are commonly used for nowcasting.

Bridge equations

Due to its simple estimation approach and low technical requirements, bridge equations have for a long time been widely used within policy organizations, primarily central banks (Schumacher, 2014). The idea behind the bridge equations is to use a few high frequency predictive variables to predict a lower frequency variable through single equation regressions. The high frequency variables are aggregated to the frequency of the predicted variable through a set of transformations and can, after being forecasted, be used to predict the variable of interest. See Rünstler and Sédillot (2003) or Baffigi et al. (2004) for more information on the use of bridge equations in nowcasting.

MIDAS with lag structure

The downside of using bridge equations is the large amount of parameters that need to be estimated (Dauphin et al., 2022). The MIDAS approach address this issue by using a non-linear polynomial structure and also handles time series sampled with a varying frequency, see e.g. Ghysels et al. (2006).

Factor models

There are also many models that are built upon the state-space representation, where the DFM is the most popular one. In one of the first models, Stock and Watson (1989) extracted a single common factor from a set of variables such as employment and industrial production. The models were soon able to handle a larger number of variables and factor models are now widely used to nowcast economic variables. Giannone et al. (2008) created a model to nowcast GDP, which has now been refined and implemented by

various central banks.

Machine learning

With the recent developments, machine learning has become more popular among economists. The area of nowcasting is no exception, and several techniques utilizing machine learning has emerged. For example, Dauphin et al. (2022) show that ML models work well to nowcast GDP during times of high volatility. The fast development is partly due to new data becoming available through for example Google trends. Choi and Varian (2012) showed that by using specific keywords in Google trends, it is possible to nowcast economic conditions.

1.5 Model overview

The model and overall nowcasting approach will be briefly presented below.

The models that will be used in this master's thesis all utilize the MIDAS framework to handle data but use different techniques to form estimates. The first model is MIDAS with Almon lag structure, the second is unrestricted MIDAS, the third RF and the last XGB. The two first are classical regression models commonly used with the MIDAS framework while the latter two are machine learning models.

To begin, it is necessary to gather a data set of predictive time series that accurately reflect the various aspects of the economy using relevant economic variables. After the data set is gathered, it is time to model. The modelling is done in four steps. The first step is to perform frequency alignment on the data and arrange it according to the MIDAS framework. See Chapter 4 for details about frequency alignment and the MIDAS framework. The second step is to conduct variable selection to reduce the number of variables in the model. Chapter 5.1 will discuss this in more detail. Then the third step

is to perform feature selection on the remaining variables. See Chapter 5.2 for details regarding this process. The fourth and final step is to run and evaluate the models daily on an expanding window, and this will be covered in Chapter 7.

1.6 Structure of paper

This master's thesis is structured as following. The next Chapter will introduce all the necessary theory for the models used and variable and feature reduction. Chapter 3 will present the data set and details regarding transformations. In Chapter 4 the methodology behind frequency alignment will be in explained. Chapter 5 explains the procedure used for selecting variables and features. Chapter 6 explains how the models are used while Chapter 7 covers the estimation approach. Chapter 8 presents the results and Chapter 9 concludes.

Chapter 2

Theory

This Chapter presents the theory behind the models that will be used further in the paper.

2.1 K-Fold Cross validation

Cross validation is a procedure that can be used to select the best hyper-parameters for a model (Brownlee, 2018). K is an important parameter to specify, and it refers to the number of groups that the data is split up into prior to evaluation. When a value for K is selected, it is often referred to in the model name, for example 5-fold cross validation. The general procedure is as follows:

1. The data set is randomly shuffled
2. The data set is split up into K groups
3. For every group the following is done:
   (a) Set the data in the group as test data
   (b) Set the data in the remaining groups as train data

(c) Fit a model on the training data and evaluate it on the test data

(d) Retain the score of the model

4. Take average of the scores.

Using 5-folds cross validation, it is possible to evaluate many models with different combinations of hyperparameters and eventually select the one with the best score or lowest error. It is important that every sample is put into one group, and stays in that group throughout the process. Thus, each sample is used once to evaluate the model and K-1 times to train the model. The fact that all samples are used to both train and evaluate the model often leads to models with less bias, making it a popular method.

2.2 Ordinary least squares

Suppose a linear regression model:

$$Y = X\beta + \epsilon, \tag{2.1}$$

where the variable of interest, Y, is to be explained by p covariates $X = (X_1, \ldots, X_p)^T$. The β vector can be estimated by the OLS methods by solving:

$$\hat{\beta}^{OLS} = \min_{\beta} \left\{ \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \right\}. \tag{2.2}$$

It can be shown that the best linear unbiased estimator can be formed as following:

$$\hat{\beta}^{OLS} = (X^T X)^{-1} X^T Y \tag{2.3}$$

Equation 2.3 is the so-called Normal Equation, see Eric Ghysels (2018) for more details.

2.3 Least Absolute Shrinkage and Selection Operator

The original Least Absolute Shrinkage and Selection Operator (LASSO) was first introduced in 1996 by (Tibshirani, 1996), with the aim to improve upon the OLS estimates which are obtained by minimizing the residual squared error. Two of the alternative techniques are the subset selection and ridge regression, both have some drawbacks. Subset selection is a discrete process and either keeps or drops the regressors. Therefore small changes in the data can lead to very different models, and thus high variability (Tibshirani, 1996). Ridge regression (Hoerl and Kennard, 1970) is instead a continuous process and shrinks coefficients, making it more stable. However, since no coefficients are set to zero, the resulting model is harder to interpret, as well as risking keeping irrelevant regressors in the model. LASSO tries to retain the good features of both subset selection and ridge regression by shrinking some parameters and setting some to zero. Further, LASSO is more efficient and thus able to handle problems of higher dimension, which can be a big problem with the previously mentioned methods. Below, the theory behind the original, as well as some variations of LASSO will be presented.

2.3.1 LASSO

It is common for linear regression models, such as Equation 2.1, to include some covariates that are more or less irrelevant. If this is the case, the β vector is sparse and the model could be improved by finding the important covariates, and thus discarding the irrelevant ones. One common way to do this is to impose restrictions on the number of included variables by

adding some constraint on the regular OLS problem (2.2) introduced earlier by Tibshirani (1996).

The constraint could be formulated in several ways. LASSO proposes a L_1 penalization in Equation 2.2, aiming to perform variable selection and reduce the dimension of the problem. By adding the L_1 penalty, the OLS is transformed, keeping the convex properties, to the following optimization problem:

$$\begin{aligned} &\hat{\beta}^{LASSO} = \min_{\beta} \left\{ \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \right\}, \\ &s.t. \sum_{j=1}^{p} ||\beta_j||_1 \leq t, \end{aligned} \tag{2.4}$$

where t is a tuning parameter. Finally, Equation 2.4 can be rewritten as:

$$\hat{\beta}^{LASSO} = \min_{\beta} \left\{ ||y - X\beta||_2^2 + \lambda ||\beta||_1 \right\}. \tag{2.5}$$

The sparsity of the solution of the β vector is decided by the shrinkage parameter ($\lambda \geq 0$). For greater values of λ, the β coefficients are more penalized leading to a greater number of elements being shrunken to zero and thus a more sparse solution. If λ is set to 0, the original OLS solution is returned.

2.3.2 Group LASSO

In some regression problems the predictors have a natural grouped structure. Sometimes the variables within a group all originate from the same variable, in the form of for example a transformation or time lag. Other examples of group structures are groups of genetic pathways in the field of genetics or groups of technical indicators in finance (Méndez Civieta et al., 2021). For these problems, variable selection typically amounts to selecting important

groups of variables rather than individual variables.

Yuan and Lin (2006) propose a variant of the LASSO, group LASSO (GL), to deal with regression problems where the predictors have a grouped structure. Suppose there are m groups. The new β-vector then becomes $\beta = \{\beta^{(1)}, \beta^{(2)}, \ldots, \beta^{(m)}\}$ where $\beta^{(l)}$ is the part of β corresponding to the *l-th* group. The same is done for X and $X^{(l)}$ represents the submatrix of X with columns corresponding to group l. The optimization problem becomes:

$$\hat{\beta}^{GL} = \min_{\beta} \left\{ ||y - \sum_{l=1}^{m} X^{(l)}\beta^{(l)}||_2^2 + \lambda \sum_{l=1}^{m} \sqrt{p_l}||\beta^{(l)}||_2 \right\}, \tag{2.6}$$

with p_l being the number of weights in $\beta^{(l)}$, or equivalent to the number of variables in the *l-th* group. GL thus works similarly to LASSO, but while LASSO penalizes every individual predictor, GL penalizes entire groups. This results in group sparse solutions, either keeping or setting entire groups of variables to zero.

2.3.3 Sparse Group LASSO

GL does not introduce sparsity within the groups. To do this, Friedman et al. (2010) propose the Sparse Group LASSO (SGL), defined as a linear combination between LASSO and GL. This results in solutions that are sparse both between and within groups. The penalization problem is the following:

$$\hat{\beta}^{SGL} = \min_{\beta} \left\{ ||y - \sum_{l=1}^{m} X^{(l)}\beta^{(l)}||_2^2 + \alpha\lambda||\beta||_1 + (1-\alpha)\lambda \sum_{l=1}^{m} \sqrt{p_l}||\beta^{(l)}||_2 \right\}, \tag{2.7}$$

where $\beta = \{\beta^{(1)}, \beta^{(2)}, \ldots, \beta^{(m)}\}$ is the entire parameter vector. For $\alpha = 1$ the problem reduces to that of LASSO and for $\alpha = 0$, it reduces to GL.

2.3.4 Adaptive LASSO

All previously presented versions of the LASSO apply a constant penalization rate. Zou (2006) shows that this results in biased estimates, which in turn leads to reduced quality of the variable selection as well as increased forecasting error. To solve this problem and to improve upon the original versions of the LASSO, he introduced the Adaptive LASSO (AL). Simply put, the idea behind AL is to introduce additional weights on the penalization. AL fulfills the oracle property, meaning it has the probability of 1 to find the correct model.

Adaptive Sparse Group LASSO

Based on the original paper (Zou, 2006) on AL, the adaptive idea has been applied to the other versions of LASSO. Mendez-Civieta et al. (2021) introduced the Adaptive Sparse Group LASSO (ASGL) defined as follows:

$$\hat{\beta}^{ASGL} = \min_{\beta} \left\{ ||y - \sum_{l=1}^{m} X^{(l)}\beta^{(l)}||_2^2 + \alpha\lambda \sum_{j=1}^{p} \tilde{\omega}_j ||\beta||_1 + (1-\alpha)\lambda \sum_{l=1}^{m} \sqrt{p_l}\tilde{v}_l ||\beta^{(l)}||_2 \right\}, \tag{2.8}$$

where $\tilde{\omega}$ and $\tilde{v}$ are predefined weight vectors. It should also be mentioned that AL and Adaptive Group LASSO (AGL) can be defined similarly by adding weights to the previously defined LASSO and GL respectively.

With ASGL the weights can be set to adjust how variables or groups of variables should be penalized before running the optimization. Therefore, the weights for important variables (or groups of variables) should be small to reduce the penalization and the opposite for less important variables.

2.3.5 Weight calculation for Adaptive LASSO

Since the weights $\tilde{\omega}$ and $\tilde{v}$ are to be set before the optimization, it is necessary to investigate how they should be specified for best results.

Zou (2006) originally proposed to define the weights $\tilde{\omega}$ based on the results of a non penalized model according to:

$$\tilde{\omega_i} = \frac{1}{|\tilde{\beta_i}^{\gamma}|}, \tag{2.9}$$

where $\tilde{\beta}$ is the solution received from unpenalized linear regression and γ is a non-negative constant. By looking at Equation 2.9 it is clear that a larger β coefficient, meaning that the variable is more important, results in a smaller weight which results in less penalization. The opposite goes for less important variables with smaller β, leading to an increased weight as well as heavier penalization.

The original weight definition (2.9) works fine in lower dimensions when the number of samples is greater than the number or features ($n > p$). However, when dealing with regression models of higher dimension ($p \gg n$) it is not feasible to solve an unpenalized model, making Equation 2.9 hard to use. Mendez-Civieta et al. (2021) introduce other ways of calculating the weights for AL for higher dimensional frameworks. Further, Mendez-Civieta et al. (2021) finds that out of all proposed methods, *PCA based on a subset of components* yielded the best results and therefore suggests using it when faced with a higher dimensional problem. PCA stands for Principal Component Analysis.

PCA based on a subset of component

The matrix of principal components Q is defined in a way such that the first principal component has the largest variance and each succeeding one having

the maximum possible variance under the constraint that it is orthogonal to the preceding components. Mendez-Civieta et al. (2021) propose taking advantage of the low dimensional framework defined by the PCA scores $Z = XQ$. Further consider the submatrix $Q_d = (q_1, \ldots, q_d)^t$ where the d can be selected to explain a certain amount of the variability. Finally solve the unpenalized model:

$$\tilde{\beta} = \min_{\beta} \left\{ \frac{1}{n} \sum_{i=n}^{n} \rho_\tau (y_i - z_i^t \beta) \right\}, \tag{2.10}$$

where $Z_d = XQ_d$.

This solution, $\tilde{\beta}$, is then used to estimate the higher dimensional solution, $\hat{\beta} = Q_d \tilde{\beta}$, which is then used to compute the adaptive weights as follows,

$$\tilde{\omega} = \frac{1}{|\hat{\beta}_j|^{\gamma_1}} \text{ and } \tilde{v} = \frac{1}{||\hat{\beta}^{(l)}||_2^{\gamma_2}}, \tag{2.11}$$

where $\hat{\beta}^{(l)}$ is the vector of components related to the *l-th* group and $\hat{\beta}_j$ is the *j-th* component of β respectively. γ_1 and γ_2 are constants which needs to be selected.

2.4 Distributed lag models

Distributed lag models is a class of models where the target variable is a sum of lags of the exogenous variable.

$$y_t = a + w_0 x_t + w_1 x_{t-1} + ... w_n x_{t-n} e_t \tag{2.12}$$

Here y_t is the target variable, x_t is the exogenous variable at time t and w_0 its associated parameter, often called weight in this context. The weights can easily be estimated with ordinary least squares (OLS). If one has n lags

included in the model *n+1* parameters must be estimated as it also has an intercept *a*. This can in some cases lead to numerous parameters needing to be estimated, which can be troublesome if the amount of data is limited.

Since lagged variables often have strong multicollinearity, it may not be necessary to estimate one weight per lag. Instead, one can use the assumption of multicollinearity to perform a structured estimation using finite distributed lag structures. This leads to that the number of parameters requiring estimation is reduced. Distributed lag structures enforce a parameterized structure upon the lags. These structures often only depend on two parameters and by estimating them, weights for all lags are found. Two commonly used finite distributed lag structures used in economics are the Exponential almon lag and the Beta lag. These are often used together with the MIDAS framework where multiple lags of variables are included, see Eric Ghysels (2018).

2.4.1 Almon lag

The most common lag structure is the Exponential almon lag, which is a modification of the original almon lag introduced by Shirley Almon. Here the lag distribution is fitted using two parameters. It has been proved many times that the exponential almon lag works well in econometrics and is a common choice, see Bokati and Kreinovich (2022).

$$w(k,\theta) = \frac{\exp(\theta_1 k + ... + \theta_Q k^Q)}{\sum_{k=0}^{K} \exp(\theta_1 k + ... + \theta_Q k^Q)} \tag{2.13}$$

Here k denotes index of lag. See Almon (1965) for a full explanation of the Almon lag.

2.4.2 Beta lag

The other common choice is the Beta lag, where the Beta distribution is used

$$w(k,\theta) = \frac{f(\frac{k}{K},\theta_1;\theta_2)}{\sum_{k=0}^{K} f(\frac{k}{K},\theta_1;\theta_2)} \tag{2.14}$$

$$f(x,a,b) = \frac{x^{a-1}(1-x)^{b-1}\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \tag{2.15}$$

where $\Gamma(a) = \int_0^\infty e^{-x}x^{a-1}dx$ is the Gamma function.

A common specification of the Beta polynomial is setting $\theta_1 = 1$ and estimating θ_2 with the requirement that it must be larger than one, see Eric Ghysels (2018) for more details regarding the Beta lag.

2.5 Machine Learning models

2.5.1 Decision tree

Decision trees are a popular machine learning technique used for solving classification and regression problems. They work by recursively partitioning the input space based on a set of decision rules, which are derived from the training data. A popular version is Classification and Regression Tree (CART) introduced by Breiman (1984). It uses Gini's impurity index to decide when to split the data.

The process of building a decision tree begins with selecting a feature that best splits the data based on some criterion, such as maximizing information gain or minimizing impurity. This feature is then used to partition the data into two subsets, with each subset containing only those samples that satisfy the decision rule associated with that feature.

The process is repeated for each subset, and a new feature is selected to

partition the data into smaller subsets. This process continues until some stopping criterion is met, such as reaching a maximum depth, achieving a minimum number of samples per leaf node, or achieving a minimum reduction in impurity.

Once the decision tree is constructed, it can be used to make predictions by traversing the tree from the root node to a leaf node. At each internal node, the decision rule associated with the corresponding feature is evaluated, and the traversal proceeds to the child node that satisfies the rule. This process continues until a leaf node is reached, which corresponds to a predicted class or value.

Decision trees have several advantages, such as being interpretable, easy to understand, and able to handle non-linear relationships between the features and target variable. However, they can also suffer from overfitting, where the tree becomes too complex and memorizes the training data rather than learning the underlying patterns. To mitigate this, techniques such as pruning, regularization, and ensemble methods are commonly used. (Sun and Zhou, 2018)

2.5.2 Random forest

RF is an algorithm which builds on the decision tree algorithm by combining multiple decisions trees in a clever way to lower the risk of overfitting. The idea is to generate multiple decision trees trained on a randomly selected subset of features and aggregate the result. The name forest comes from using multiple trees. This method was developed by Breiman (2001)

The first step when creating a RF model is to create new data sets with the same number of samples as the original data set. The new data sets are generated by random sampling with replacement from the original samples, bootstrapping the data. Then a random subset of features is selected for each decision tree and it is on these features each tree learns its decision nodes.

When the forest is used to predict, the new sample is passed through each tree and the result is aggregated to form the prediction. See Figure 2.1 for an illustration of how a new sample is passed through each tree in the forest and then aggregated into a final result.

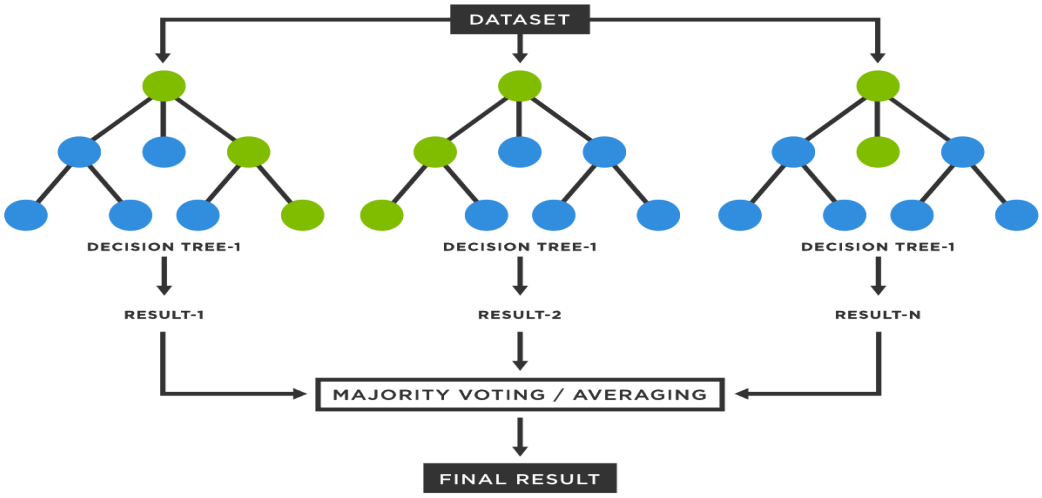

Figure 2.1: Illustration of RF and how it works. Source: Tibco software inc.

When training a RF, it is important to specify a couple of hyperparameters: number of trees in the forest, maximum number of features used in the decision trees, maximum depth of the decision trees, number of samples to use in the decision trees, the minimum number of samples to split an internal node and lastly the minimum number of samples required to be at a leaf node.

The number of trees to use depends on the size of the dataset, but there often exists an upper limit after which adding more trees has a diminishing effect. The number of trees often ranges from hundreds to thousands. Both the number of features used in the trees and the maximum depth of these trees are important to avoid overfitting. Too many features used, and the model might overfit the data since it becomes too complex. The same goes for maximum depth: the deeper the tree, the more complex and specific it

gets. The number of samples to use in the trees can lead to overfitting if set too high or under-fitting if set too low, since it affects how many samples each tree can model on. The minimum number of samples required to split a node also affects overfitting. A low value can lead to excessive splitting of the tree and make it more complex than necessary. Finally, the minimum number of samples required to be at a leaf node controls the complexity of the tree. A high number forces the tree to be small and under-fit, while a low number of minimum samples can lead to overfitting. For more details regarding parameters and tuning, see Sharoon (2020).

2.5.3 XGBoost

XGB is an algorithm which also builds on decision tree's. The goal here is also to use multiple tree's to increase performance and lower the risk of overfitting. For all details regarding XGB, see Chen and Guestrin (2016).

In XGB decision trees are built additive in series, using the residual from the other tree's. The predicted output is formed using K additive functions.

$$\hat{y}_i = \phi(\boldsymbol{x}_i) = \sum_{k=1}^{K} f_k(\boldsymbol{x}_i) \tag{2.16}$$

where f_k are the decision trees.

Let L denote the *regularized* objective. The goal is then to minimize L.

$$\begin{aligned} L(\phi) &= \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \\ \text{where } \Omega(f) &= \gamma T + \frac{1}{2}\lambda ||w||^2 \end{aligned} \tag{2.17}$$

Here l is a differential convex loss function that measures the distance between the prediction and the true value. Ω is a term that penalizes the

complexity of the tree structure. T is the number of leaf in the tree, w is the leaf weights and γ and λ are coefficients.

Since the equation above includes functions as parameters, it can not be optimized using traditional methods and is instead trained in an additive manner. Let $\hat{y}_i^{(t)}$ be the prediction at the t-th iteration of the i-th instance. The goal is to add f_t to minimize the equation below.

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(\boldsymbol{x}_i)) + \Omega(f_t) \tag{2.18}$$

This means that the f_t that improves Equation 2.17 is added. By using a second order approximation it is easy to optimize the objective in a general setting.

$$L^{(t)} \approx \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(\boldsymbol{x}_i) + \frac{1}{2} h_i f_t^2(\boldsymbol{x}_i)] + \Omega(f_t) \tag{2.19}$$

where $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ and $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

Similar to RF, XGB also requires specifying hyperparameters. These are: number of trees to use, maximum depth of the decision trees, subsample ratio of observations to use per iteration, subsample ratio of features to use per iteration and finally the learning rate. Learning rate controls how much the model adapts is response to new information when training.

For details regarding number of trees to use and maximum depth, see Chapter 2.5.2 as it uses the same hyperparameters. The subsample ratio is an important hyperparameter in XGB and is typically set to a value between 0.5 and 1, although it can be as low as 0. Lower values prevent over-fitting and makes the model more conservative, but too low of a value can lead to under-fitting. The same is true for the subsample rate of features. Valid values for the learning rate range from 0 to 1, but they typically end up at

around 0.01 to 0.2. A high learning rate might lead to over-fitting. It's important to find the right balance between the learning rate and the number of trees to get the best performance from the XGB model. For details about the parameters and tuning, see Brownlee (2016) and Prashant (2023),

2.6 Auto Regressive models

2.6.1 AR(p)

The autoregressive (AR) model is a model where the output variable depends on lags of the variable itself and a stochastic term. An AR model of order p is defined as follows:

$$y_t = \phi_1 y_{t-1} + ... + \phi_p y_{t-p} + \epsilon_t, \tag{2.20}$$

where ϵ_t is white noise, meaning it has a mean of zero, constant variance and is uncorrelated to the innovations of y_t.

By adding a constant it is possible to rewrite Equation 2.20 as:

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \epsilon_t \tag{2.21}$$

In order to use the model for forecasting the parameters ϕ_i and c need to be estimated, which can be done by using a maximum likelihood estimator. For more in-depth details, see Lindgren et al. (2013).

Chapter 3

Data

3.1 Data set

The data set used in this master's thesis consists of 100 time series with varying frequencies. The set contains a variety of macroeconomic variables such as: measurements of money and credits, measurements of economic activity but also commodity prices and different yields.

The starting point for the data set is the commonly used database FRED-MD. FRED-MD is a database with macroeconomic monthly variables and is provided by Federal Reserve of St. Louis, see Michael W. McCracken (2015) for details. This data set was chosen due to it being a recognized starting point in many articles conducting big data analysis and because it contains a mix of variables representing many aspects of the economy in the U.S. FRED-MD collects its data from the database Archival Federal Reserve Economic Data (ALFRED). ALFRED provides an even broader spectrum of data series than FRED-MD and with varying frequencies. All series in FRED-MD available at a higher frequency from ALFRED is replaced with its higher frequency counterpart, since we want to use data of varying fre-

quencies. Higher frequencies in this case is weekly or daily. In addition to the series from FRED-MD and ALFRED, daily commodity prices are collected from Bloomberg. These are included since it has been showed that daily and weekly financial data can improve short- to medium-term forecasts. (Andreou et al., 2013) (Pettenuzzo et al., 2016) (Adrian et al., 2019)

The monthly and weekly variables are so-called *flow variables*, whose value refer to a period of time, for example January. The other kind of variable to exists are *stock variables* and refer to a value associated with a specific point in time. For example, Industrial production is a measure of an entire month's industrial production and is therefore a flow variable. Oil price on the other hand refers to the price at a specific time and is therefore a stock variable. Due to the definition of a stock variable, its value is available at the time of the observation date. The same does not hold for flow variables; their value is often available a couple of days after the observation period ends. This difference between observation period and publication date is called publication delay, and is important to keep track of when dealing with flow variables.

3.2 Dealing with dates

For every value in the data set, two dates were gathered: the observation period and the publication date. The observation period represents the period of time to which the data point applies, and can be a range of time such as a week or a month. To maintain consistency in our notation, we denote the observation period with a single date, which we refer to as the *observation date*. For stock variables, the observation date and the publication date are the same. However, for the flow variables which are weekly or monthly, the publication date differs from the observation date. For weekly data, the observation date is set to the Monday of the referenced week, while for monthly data, the observation date is set to the first day of the referenced

month. For example, the U.S. initial claims for unemployment insurance is released weekly, and the data point for initial claims for the sixth week of 2023 has an observation date of February 6, 2023, since it is the Monday of that week.

3.3 Real time data

Using the publication date associated with every value it is possible to create data sets for each evaluation date. These data sets only include data available up until the evaluation, allowing us to only model on data that is available at the date of evaluation.

Macroeconomic variables are often subject to revisions as time goes, and more measurements become available. This introduces the concept *vintage*, which refer to the latest revisions at a certain date. Since data series can undergo re-indexing or changes in how measurements are done, even values far back in the data set may differ between different vintages.

We use the values from the most current vintage, e.i. the latest revision of the data. Primarily due to limited availability of vintage data, but also to avoid having to deal with re-indexing of data and changes in measurement over time. Revisions are often small between vintages and therefore the most current vintage is used to get coherent data with long history. The alternative to using the latest vintage is to use the vintage available at the specific time point. This would lead to different data sets depending on vintage since past values are often revised multiple times. As discussed by Monteforte and Moretti (2010), the difference of using vintage data or the latest revision might be minimal.

3.4 Missing data

3.4.1 Daily data

The daily data set contains missing values on weekends, holidays, and on some random dates. Since exchanges are closed on weekends, all weekends are removed from the data set, since no information is gained from these missing values.

For the remaining missing values occurring on weekdays, there are several ways to deal with them, such as forward filling, interpolation of the adjacent data points, or curve-fitting on past values. Since the data is real-time, it is not possible to use future data points for interpolation without introducing look-ahead bias. To avoid this, the forward fill method is used to fill the missing values. Forward filling means to replace missing values with the latest available value, and is chosen for its simplicity over the more complex curve-fitting method.

3.4.2 Weekly and monthly data

Similarly, the weekly and monthly data also has missing values, probably due to errors in reporting or re-indexing, and are forward filled with the same reasoning as the daily data.

Some monthly values lack release dates, as records are not kept for all variables and with varying history. This is handled by finding an average publication date for every series and applying the same delay to those values that lack a publication date.

Due to the varying publication delay for flow variables, there is a jagged edge in the data panel as not all variables are released at the same time. This can be a problem in other models that cannot handle jagged edges with missing values. However, using the MIDAS framework, this is not a problem since it

only uses the available values at the date of evaluation. It does not require all variables to have values at a certain date and instead uses the latest available values for each variable at that date.

3.5 Transformations and standardization

After all missing values and missing release dates are removed or replaced, the data panel needs to undergo a sequence of transformations to ensure stationary. While it is most common for inflation to be reported as yearly percentage change, see the official reporting at Bureau of Labor Statistics (2023a), monthly percentage change is also used from time to time. Both transformations yield sufficiently stationary processes, and therefore both were tested in modeling to determine which one to use. For the rest of the monthly, weekly and daily variable's percentage change from previous value is used in an attempt to capture short term movement.

To find what kind of transformation to use on the inflation series, a test was conducted. Two models were constructed using the methodology described in Chapter 7, which means variables and feature selection using ASGL. To evaluate the performance on out of sample data, XGB with default hyper-parameters was used. The choice landed on XGB due to it being a fast and reliable model. Monthly percentage change yielded the lowest RMSE, meaning it performs better and is therefore selected as the transformation to use going forward. For details about RMSE, see Chapter 7.2 and for details about the performance with the different transformations, see Table 8.1

After transformations are performed, the data set is also standardized, ensuring zero mean and unit variance. This is achieved by removing the mean and dividing with the standard deviation for each data series.

Chapter 4

Frequency alignment

Using higher frequency variables to forecast lower frequency variables is not always an easy task, and there are several ways to do it. This paper utilizes the MIDAS framework, which was first presented by Ghysels et al. (2006), and has since been one of the most prominent approaches to handle mixed frequency data. The first step when estimating a MIDAS regression model is to make sure that all variables observed at different frequencies and different lag lengths are aligned. Ghysels et al. (2016) denotes this the frequency alignment process. After the frequency alignment, Ghysels et al. (2006) presents further steps to finalize the model. These steps will be covered later in Chapter 6, together with the other models applied. In this paper, the MIDAS frequency alignment step is combined with not only the MIDAS regression model, but also other models. The frequency alignment step is therefore presented separately in this Chapter, as well as other alternatives to handle mixed frequency data.

4.1 The MIDAS framework as a feature engineering process

For simplification, the frequency alignment step will be demonstrated with examples in matrix notation. Suppose we want to explain the behavior of y_t, observed monthly, with another variable $x_t^{(1)}$ which is observed at a higher frequency, weekly. The model assumes that each month has 4 weeks and the frequency is thus set to $m = 4$ to illustrate how many observations of $x_t^{(1)}$ that are available for each observation of y_t. In practice, months do not have exactly 4 weeks and one can think of the model as simply taking a fixed set of weekly lags. This basically means that instead of including the previous month of weekly lags, a fixed amount of the most recent available lags are included. Assuming further that the weekly data in the past 2 months has explanatory power. This implies that for each month t, y_t is modeled as a linear combination of the observations of $x_t^{(1)}$ during month t and $t-1$. More specifically, $x_{4t}^{(1)}$, $x_{4t-1}^{(1)}$, $x_{4t-2}^{(1)}$ and $x_{4t-3}^{(1)}$ are observed during month t and $x_{4(t-1)}^{(1)} = x_{4t-4}^{(1)}$, $x_{4(t-1)-1}^{(1)} = x_{4t-5}^{(1)}$, $x_{4(t-1)-2}^{(1)} = x_{4t-6}^{(1)}$ and $x_{4(t-1)-3}^{(1)} = x_{4t-7}^{(1)}$ are observed in the previous month $t-1$. In matrix notation the frequency aligned MIDAS model for this example can be written as:

$$\begin{bmatrix} y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_8^{(1)} & \dots & x_1^{(1)} \\ \vdots & \vdots & \vdots \\ x_{4n}^{(1)} & \dots & x_{4n-7}^{(1)} \end{bmatrix} \begin{bmatrix} \beta_1^{(1)} \\ \vdots \\ \beta_8^{(1)} \end{bmatrix} + \begin{bmatrix} \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}. \tag{4.1}$$

It can now be seen that the high frequency variable $x_t^{(1)}$ has been transformed into a lower frequency vector $[x_{4n}^{(1)}, \dots, x_{4n-7}^{(1)}]^T$. Thus, all variables are observed at the same frequency making it possible to accommodate variables sampled at different frequencies and transform the problem into a normal time series regression. In essence, this is what the frequency alignment is about.

This process can be repeated for all other variables that are to be included in the model. Suppose another variable $x_t^{(2)}$ observed at daily frequency, is added to the model and that only values from the previous month are considered to have explanatory power. For month t, y_t is now modeled as a linear combination of the variables $x_{4t}^{(1)}$, $x_{4t-1}^{(1)}$, $x_{4t-2}^{(1)}$, $x_{4t-3}^{(1)}$ and $x_{30n}^{(2)}$, ..., $x_{30n-29}^{(2)}$ observed in month t, and the variables $x_{4(t-1)}^{(1)}$, $x_{4(t-1)-1}^{(1)}$, $x_{4(t-1)-2}^{(1)}$, $x_{4(t-1)-3}^{(1)}$ observed in month t–1.

$$\begin{bmatrix} y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_8^{(1)} & \dots & x_1^{(1)} \\ \vdots & \vdots & \vdots \\ x_{4n}^{(1)} & \dots & x_{4n-7}^{(1)} \end{bmatrix} \begin{bmatrix} \beta_1^{(1)} \\ \vdots \\ \beta_8^{(1)} \end{bmatrix} + \begin{bmatrix} x_{60}^{(2)} & \dots & x_{31}^{(2)} \\ \vdots & \vdots & \vdots \\ x_{30n}^{(2)} & \dots & x_{30n-29}^{(2)} \end{bmatrix} \begin{bmatrix} \beta_1^{(2)} \\ \vdots \\ \beta_{30}^{(2)} \end{bmatrix} + \begin{bmatrix} \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix} \tag{4.2}$$

The second example illustrates the flexibility of the frequency alignment process. First, it shows how variables with 2 different frequencies (weekly and daily) easily are incorporated into the framework and used to describe a variable with a third frequency (monthly). Second, it shows the flexibility in terms of the number of lags that are included for each variable with the weekly variable including the past 2 months and the daily variable only including the past month. As earlier mentioned, the number of weeks and days every month varies and in practice the model takes a fixed number of lags for each variable instead of a time period.

In the general case, a high frequency variable x_t is frequency aligned by transforming it to the low frequency vector $[x_{tm_i}^{(i)}, x_{tm_i-1}^{(i)}, \dots, x_{tm_i-l}^{(i)}]^T$. The model can then be written in matrix notation as:

$$\begin{bmatrix} y_l \\ \vdots \\ y_n \end{bmatrix} = \Sigma_{i=0}^{k} \mathbf{X}^{(i)} \begin{bmatrix} \beta_0^{(i)} \\ \vdots \\ \beta_l^{(i)} \end{bmatrix} + \begin{bmatrix} \epsilon_l \\ \vdots \\ \epsilon_n \end{bmatrix} \tag{4.3}$$

where

$$\mathbf{X}^{(i)} = \begin{bmatrix} x^{(i)}_{um_i} & x^{(i)}_{um_i-1} & \dots & x^{(i)}_{um_i-l} \\ x^{(i)}_{(u+1)m_i} & x^{(i)}_{(u+1)m_i-1} & \dots & x^{(i)}_{(u+1)m_i-l} \\ \vdots & \vdots & \dots & \vdots \\ x^{(i)}_{tm_i} & x^{(i)}_{tm_i-1} & \dots & x^{(i)}_{tm_i-l} \\ \vdots & \vdots & \dots & \vdots \\ x^{(i)}_{(n-1)m_i} & x^{(i)}_{(n-1)m_i-1} & \dots & x^{(i)}_{(n-1)m_i-l} \\ x^{(i)}_{nm_i} & x^{(i)}_{nm_i-1} & \dots & x^{(i)}_{nm_i-l} \end{bmatrix}$$

and u is the smallest integer so that the following holds:

$$um_i - l > 0,$$

$$u > p.$$

4.2 Keeping track of lags

The model will be used on real time data, meaning it will only use data that is available at the time of the estimation. This means that the latest daily, weekly and monthly variables are used when forming the estimate. Depending on how many lags the model uses and at what day the model is evaluated at, the variables might belong to the target month, the month before, the month after or a combination. This is not always something desired. Values that have an observation date past the target month should not have any effect on the inflation, since all measurements for the target month are done. For example, when estimating the inflation for January 2023 commodity prices with observation dates in February should not have any effect, neither should unemployment numbers for February. Here it is important that the observation date should be in the target month while there is no restriction on publication date. See Chapter 3.2 for definitions of

publication date and observation date.

Therefore a limit is set on lags to only include values with observation dates in the target month or the month before. For example, say that the evaluation date is February 9, 2023, which means that January's inflation is being estimated. Then only variables with observation date in January or December are included. The last lag of daily variables will have observation date January 31.

4.3 Variables and features

Throughout this paper, the terms variables and features will be used a lot, and might require a brief explanation to improve understanding. Specific data series, such as the unemployment, are referred to as variables. After the feature engineering process, each variable is transformed into a lower frequency vector. This vector contains time lags of the variable and each time lag will be referred to as a feature. In the previous Chapter 4.1, the variable would be x_t^i, while the low frequency vector, containing the features, would be $[x_{4n}^{(i)}, \ldots, x_{4n-7}^{(i)}]^T$.

4.4 Alternative approaches

There are several ways to handle mixed frequency data. One common way to solve the problem is to pre-filter the data which in essence means that the frequency of the high frequency data is transformed to that of the low frequency one. The most straightforward way of pre-filtering the data is to aggregate the high frequency data and take their average (Asimakopoulos et al., 2013).

Pre-filtering is a very simplistic approach and fails to exploit all useful information that may be contained in the high frequency data. Therefore, to

avoid pre-filtering, numerous alternative methods have been developed with MIDAS, bridge equations and factor models being some of the most common ones. See (Foroni and Marcellino, 2013) for more detailed information on approaches to handle mixed frequency data.

Another method for dealing with mixed frequency is to model at multiple frequencies and then use temporal aggregation to combine them. Here, long-term forecasts are combined with short-term forecasts to utilize information carried in the time series at the different frequencies. For an example of how this approach can be used, see Nystrup et al. (2021).

Chapter 5

Dimensionality reduction

Today, there is a lot of data available, which has made high dimensional frameworks, in which the number of variables p exceeds the number of observations n, more common. This increases the need to select the best variables in a systematic manner to be able to create the best model. This issue is often referred to as *the curse of dimensionality* (Venkat, 2018).

In this paper, frequency alignment using the MIDAS framework, explained further in Chapter 4, is used to incorporate data with mixed frequencies. When doing so, each time lag of each variable will be presented to the model as a unique feature, which can lead to a high dimensional framework. For example, if 10 variables, each with 30 time lags are to be used, the model is presented with 300 features. The dimensionality problem in this paper is handled by first removing redundant variables, and then using two different techniques to reduce the number of features from the remaining variables. How this is done will be the focus of this Chapter.

5.1 Variable selection

The purpose of this Chapter is to describe the initial variable selection process, which aims to reduce the number of variables in the original data set while retaining the most explanatory ones.

To achieve this, the variable selection method must work well with the MIDAS framework and recognize groups of features that belong to the same variable. Additionally, it should be a sparse method, which can set coefficients to zero, as a single lag may be important while the others may not contribute much. For daily variables with multiple lags, this is especially critical. This paper uses LASSO (2.3) for variable selection. Combining the MIDAS framework with some variant of LASSO has been shown to work well before, see the papers by Mogliani and Simoni (2021a) or Babii et al. (2022).

5.1.1 Selection of LASSO variant

Since there are several variants of LASSO, an important part of performing the best possible variable selection and thus achieving the best result, is to select the best variant for the problem at hand. The original LASSO (2.3.1) can sometimes have problems with estimation bias and a lack of selection consistency (Mogliani and Simoni, 2021b). The adaptive versions of LASSO (2.3.4) mitigates this problem, making it more attractive. To apply the adaptive versions of LASSO, it is crucial to calculate the weights properly. Mendez-Civieta et al. (2021) propose using the method *PCA based on a subset of components*, further described in Chapter 2.3.5, and this is also the proposed method in this paper.

Mogliani and Simoni (2021b) further argues that the group penalty, described in Chapter 2.3.2, works well in combination with the MIDAS framework. This is mainly because one can assign all lags of a particular variable to a single group. Shrinkage can then be performed over the entire variable.

The fact that the lags within a variable by construction have a very high correlation can cause problems if a version of LASSO without group structure were to be applied. For example, the estimator would likely randomly select one or very few of the lags from each variable and shrink the remaining (potentially relevant) ones to zero, since they are all highly correlated. For more theoretical details, please see the papers by Efron et al. (2004) and Zou and Hastie (2005) that discusses similar problems.

For these reasons, the proposed method for variable selection in this paper is ASGL (2.3.4). AGL would also be a valid alternative. However, since the sparse version is a linear combination of AGL and AL, AGL can be obtained simply by setting α to 0 in Equation 2.8. Further, since the hyperparameters are to be optimized (more on this in Chapter 5.1.2), the sparse version seems to be the more flexible version out of the two and was thus selected.

5.1.2 Selection of ASGL hyperparameters

The selection of the optimal hyperparameters is very important since small changes can have a big impact on what variables are selected.

ASGL takes four hyperparameters, two of which (α and λ) are found in Equation 2.8 and the remaining two (γ_1 and γ_2) are used to calculate the adaptive weights in Equation 2.11. As described earlier, ASGL is a linear combination between AGL and AL. The degree to which each version will be included is decided by α and setting it to 0 yields AGL while setting it to 1 yields AL. Further, λ decides the amount of penalization and γ_1 and γ_2 are non-zero constants.

To select the optimal values for the hyperparameters, 5-fold cross validation, described in Chapter 2.1, was used. While this method works well, it relies on the researcher initiate a reasonable grid to optimize on, not missing any potentially better values for any parameter. Further, the process is quite time-consuming and initiating a grid with too many values would not be

feasible with the available computing power for this project. To solve this, the cross validation process was divided into steps, where the goal of each step was to narrow down the range of the parameters. First, an equally distanced coarse grid was defined, making sure most potential values for each hyperparameter were inside the limits of the grid. The values for the initial grid were determined according to recommendations by Méndez Civieta et al. (2021). By optimizing over this grid, initial estimates of the parameters were found. By then adjusting the grid to center around the initial estimates and increasing the resolution, the parameters found will be closer and closer to the optimal ones. This process was repeated until the estimations converged.

5.1.3 How variables are selected

The idea behind selecting variables with ASGL is very simple. After finding the optimal hyperparameter, the model is run and a vector of parameters is obtained. Then all variables with at least one of its time lags not shrunken to zero, are kept in the model.

5.2 Feature reduction

Depending on the amount of time lags that are to be incorporated, each variable can result in many features for the model to handle. Therefore, an important part of reducing the dimension of the problem is to reduce the number of features from the remaining variables after the variable selection step. Two alternative methods, ASGL and Almon lag, are proposed to handle this.

Due to high risk of over-fitting when using multiple lags where there might be high correlation between lags, feature selection is conducted. Even though machine learning models work well in high dimensional settings, it is often useful to remove redundant features.

5.2.1 ASGL for feature reduction

ASGL can be used to select the most desired features and reduce the dimension of the problem. The process is very similar to the one described in Chapter 5.1.3. However, instead of then just removing the variables of which all parameter values are shrunken to 0, all features with a parameter value of 0 were removed. This way not only variables but also individual features, or time lags, are removed.

5.2.2 Almon lag for feature reduction

An alternative to using ASGL for feature reduction is to use a distributed lag model. As mentioned in Chapter 2.4 the lag structures often only depend on one or two parameters, and therefore reduces the number of parameters that need estimation. It does not reduce the number of features but accomplishes the same goal, reduce the number of parameters that need estimation. The choice falls on using Almon lag over Beta lag due to the Almon lag's proven effectiveness.

5.3 Alternative methods

5.3.1 Principal Component Analysis

PCA is a commonly used technique to reduce the dimensionality of data while preserving its underlying information. PCA achieves this by identifying the directions, or principal components, in which the data varies the most. These principal components are determined by calculating the eigenvectors and eigenvalues of the covariance matrix. Transforming the original data to a new coordinate system using the eigenvalues and eigenvectors results in factors that represent the information in the original data in an optimal way. For further explanation and details, see Abdi and Williams (2010).

However, PCA requires a balanced data panel with values for each variable at every time point, which is not always feasible in real-time datasets due to publication lags. One solution is to use forward-filling or prediction methods to fill in missing values to get a balanced data panel. While this could work, PCA was cast aside as variable selection tool due to poor synergy with the MIDAS framework. PCA reduces variables to factors, instead of selecting factors and would have required filling out missing values.

5.3.2 Manual selection

Another approach to variable selection is to manually choose the variables to include in the model, allowing for more control over the degree of reduction. It has been shown that a model with carefully selected variables can perform well, see Bańbura et al. (2013). However, manually selecting variables can be challenging, as two selected variables may have a high correlation, rendering one of them redundant and making the choice suboptimal. Since the dataset used in this study is large and the manual selection process can be complex, this method of variable selection was not pursued.

Chapter 6

Models

Five different models will be compared in this thesis, with one of them being an **AR(1)** model used as benchmark. The other four use feature selection in someway to reduce dimensionality and to remove excess features. Three models: **UMIDAS**, **XGB** and **RF** use ASGL as feature reduction method while the **Almon** model will use an Almon lag structure as feature reduction method.

6.1 AR(1)

The autoregressive (AR) model is a commonly used model and will be used as reference model in this master's thesis. Since the AR model does not use any exogenous variables, there is no need for neither variable nor feature selection. The choice falls on an AR(1) since the model will only be used as benchmark and a model of higher order is deemed unnecessary for this purpose.

When using the model for forecasting the parameters are first trained on the training data set. Then these parameters are used to form an estimate on

the test data set. When examining the model estimates are created using the following equation:

$$\hat{y}_t = \hat{c} + \sum_{i=1}^{p} \hat{\phi}_t y_{t-i}, \tag{6.1}$$

where y_{t-i} are known values.

6.2 MIDAS - Almon

With the Almon lag model feature reduction is part of the estimation step of the model. See Chapter 2.4.1 for details.

To use the Almon model, one must first estimate $\hat{\theta}$ by solving a minimization problem. Using the Almon lag structure with weights w as described in Chapter 2.4.1, the minimization problem takes the following form:

$$\min_{t \in n} y_t - \sum_{k=1}^{K} w_t(k, \theta_t) x_t^{(k)}, \tag{6.2}$$

where n is the number of observations, K is the number of lags and $\theta_t = (\theta^1, \theta^2)$.

The Almon lag structure takes two parameters per variable θ^1, θ^2. By solving the non-linear regression problem using the Levenberg-Marquardt algorithm, optimal θ^1, θ^2 can be found for each variable. For more information regarding the implementation and theory of the Levenberg-Marquardt algorithm, see Moré (1978).

After finding $\hat{\theta}$ all that is left is to create estimates using the following equation:

$$\hat{y_t} = \sum_{k=1}^{K} w_{t-1}(k, \hat{\theta}_t) x_{t-1}^{(k)} \tag{6.3}$$

The steps for using the model is as following:

- Initiate Almon weights on the lags for each variable
- Optimize θ^1, θ^2 for each variable
- Create estimates using the found parameters

6.3 ASGL - UMIDAS

The most simple parameter estimation for a model built with the MIDAS framework is the unrestricted-MIDAS (UMIDAS) approach. Here one parameter is estimated for every lag for every variable, compared to Almon where a structure is enforced on the lags.

After the number of feature as been reduced, all that remains is to solve a normal multivariate regression with ordinary least squares.

$$Y = X\beta + \epsilon, \tag{6.4}$$

Here Y is a column vector with the inflation values, β a column vector with unknown coefficients and X a matrix containing the remaining features. To repeat, these features are the selected lags of the selected variables which remain after variable selection and feature selection. See Chapter 2.2 for details of estimating $\hat{\beta}$.

The steps for using the model is as following:

- Find optimal hyperparameters for ASGL with 5-fold cross validation

and perform feature selection with these parameters

- Find $\hat{\beta}$ using OLS on Equation 6.3
- Create estimates using the found parameters

6.4 ASGL - XGBoost

To use XGB to its full potential, it is important to optimize its hyperparameters. Poorly chosen hyperparameters can lead to suboptimal performance and over-fitting. Therefore 5-fold cross validation is performed to select parameter values.

The steps for using the model is as following:

- Find optimal hyperparameters for ASGL with 5-fold cross validation and perform feature selection with these parameters
- Find optimal hyperparameters for XGB using 5-fold cross validation
- Create estimates using the found parameters

6.5 ASGL - Random forest

The steps for RF are exactly the same as for XGB, except hyperparameters for RF are optimized instead of XGB.

Chapter 7

Estimation approach

7.1 Selection of included lags

When using the MIDAS framework to build a model, a crucial decision is how many lags to include. Including too few lags might result in a model that fails to capture the dynamics of the target variable, while including too many lags may lead to an overly complex model that is prone to over-fitting. Although variable and feature selection can help mitigate the risk of over-fitting, including as many lags as possible is not efficient from both computational and logical perspectives.

To strike a balance between using enough lags and maintaining computational efficiency, we focus on lags that are close in time. For monthly variables, we include 12 lags; for weekly variables, we also include 12 lags; and for daily variables, we include 31 lags. The daily limit of 31 lags is set to avoid incorporating values from other months that should have no impact on inflation. This selection of lags strikes a good balance between using enough lags and keeping computation time within a reasonable span.

7.2 Nowcasting strategy

First, the data was prepared according to Chapter 3 and frequency aligned as described in Chapter 4. Then, the nowcasting strategy employed for this paper can be described in the following five steps:

1. Splitting data into periods
2. Variable selection
3. Hyper parameter tuning
4. Model estimations and out of sample nowcasts
5. Model evaluation

A more in depth description of each step can be found below.

1. Data preparation

First, the data was standardized and transformed according to Chapter 3 and then frequency aligned as described in Chapter 4.

The data set was then split into two periods to investigate how they differ and to re-estimate parameters. The first period spans **2014-01-01 – 2019-01-01** and the second period spans **2019-01-01 – 2023-01-01**. In both cases data from 1992-02-01 until the start of the period is used as training data while the period itself is used as out of sample data.

2. Variable selection

The ASGL approach (described in Chapter 5.1) was used to remove redundant variables as a first step to reduce the dimensionality of the data set. For this step to work properly, the data was standardized in accordance with the original paper (Tibshirani, 1996). Then, 5-fold cross validation was performed to find the optimal values for all hyperparameters. The ASGL was

then run with these parameters and a vector of parameters was obtained. Since the purpose of this step is to select variables, and not select single time lags, the value of the parameters are not important, and only if they were either 0 or greater than 0. Basically, all variables with at least one time lag with a parameter value not shrunken to 0 by the ASGL, was kept since this means that there might be valuable information in any of the variables time lags. Thus, only variables with all time lags deemed irrelevant by ASGL were completely removed for the rest of the project.

3. Hyperparameter tuning

Based on the training data, the optimal hyperparameter values were decided using 5-fold cross validation. Theoretically, this step could be repeated continuously as more data gets available. For example, after every out of sample nowcast of the inflation, one could potentially improve upon the previously selected hyperparameters before nowcasting the following month, by including the new data and performing another cross validation. However, this would be very time-consuming and the hyperparameter values were thus only selected twice throughout the estimation process. First, based on the training data and then once again right before performing out of sample nowcasts on the second out of sample period. This is also the reason for why the out of sample period was split into two separate ones.

The hyperparameters that were tuned in this step are the ones required for feature reduction, explained in Chapter 5.2.1, as well as the ones required for the machine learning models RF (6.5) and XGB (6.4). It is also worth mentioning that for each horizon, separate hyperparameters were selected. More on this later in Chapter 7.3.

4. Model estimations and out of sample nowcasts

To estimate the models described in Chapter 6, a growing window approach was deployed. Basically, this means that all models were initially fit with all data from the training period. These fit models are then used to nowcast the inflation of the first month of the first out of sample period. All models are then fitted once again, this time including the data from the month previously nowcasted. This means that all models are continuously fitted on the most recent data available at the time of the nowcast. This process is repeated until the end of the first out of sample period. At this point, the hyperparameters were re estimated (according to step 3 above) and the process continues until the end of the second out of sample period.

5. Model evaluation

To compare models, root-mean-square error (RMSE) is used, defined as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{m=1}^{M}\sum_{d=1}^{D}(y_m - \hat{y}_{m|d})^2}, \tag{7.1}$$

where N is the number of days in the test period, $m = 1, ..., M$ are the months of the test period, $d = 1, ..., D_m$ are the days in month m. The residual of the current-month nowcasted at a specific day is therefore $y_d - \hat{y}_{m|d}$.

The model forms a nowcast on a daily basis and it is therefore possible to evaluate it daily. To answer the questions asked in the purpose, we focus on evaluating two things, how RMSE changes as more information becomes available (the horizon lowers) and how RMSE compares between the two periods. For details about horizons see the next Chapter. Since each day is associated with a horizon, it is possible to select all days with the same horizon and compare RMSE. RMSE is also compared between the two periods by selecting all nowcasts with dates in the specific period.

7.3 Horizons

Simply put, the horizon (h) is the number of days until the release of the inflation, at the day of the nowcast. If for example, the nowcast of the January inflation is being conducted on the 29^{th} of January and the actual value is due to be released on the 15^{th} of February, the horizon would be $h = 17$.

To be able to predict at different horizons, the models need to be trained on different data sets. For example, depending on the horizon, different data may be released and available. This means that every model presented in Chapter 6 needs to trained separately for every horizon to be able to perform daily predictions. This further means that many of the other steps presented earlier needs to be repeated for every horizon in order to produce the best results. For example, the hyperparameters were tuned separately for every horizon. Looking at the ASGL feature selection, it could be the case that, at different horizons, different lags of the variables are needed and it makes a lot of sense to repeat this step for every horizon.

Chapter 8

Results & Discussion

In this Chapter the results from the experiment will be reported. If nothing else is mentioned, the results refer to the entire out of sample period, starting in 2014-01-01 and continuing until 2023-01-01. Further, the plots including results for specific horizons will only be displayed for horizons up to and including 30. The reason for this is that nowcasts at horizons above 30 are not occurring on a regular basis, since it is less common for it to be more than 30 days between two inflation releases. All results are transformed to year-on-year percentage change for easier comparisons with other papers and official reporting. Results are presented in percentage points.

8.1 Model performance overview

During the process of writing this report, many different variations of the models have been tested to find the final ones. In Table 8.1, all models as well as their specifications are presented. Below is a brief description of what the columns describing the models means.

- **Model** - Describes what type of model being used.

- **Transformation** - What transformation that were used for the yearly variables. Either year-on-year percentage difference or month-on-month percentage difference. See Chapter 3.5.
- **Feature selection** - What method, if any, was used for feature selection. Either Almon lag (5.2.2) or ASGL (5.2.1).
- **Optimized** - If the hyperparameters were tuned or if the standard values were used for the ML models.
- **Variables** - What variables that were included when creating the model.

Table 8.1 presents a selection of model configurations and their RMSE. Some results that are worth noting are: Yearly percentage change transformation on the inflation series has a RMSE that is double that of monthly percentage change. Optimizing hyperparameters for XGB and RF lowers the RMSE in both cases, but more for XGB.

Table 8.1: *RMSE for several models that have been evaluated. Model name is derived from the specification used. The models in bold letters are the main models of this thesis*

Name	Model	Transformation	Feature-selection	Optimized	Variables	RMSE
AR	AR	MoM	ASGl	-	All	0.379
XGB-default-YoY	XGB	YoY	ASGl	No	All	0.881
XGB-default	XGB	MoM	ASGl	Yes	All	0.304
XGB-optimized	XGB	MoM	ASGl	Yes	All	0.269
XGB-reduced	XGB	MoM	ASGl	Yes	Monthly	0.349
RF-default	RF	MoM	ASGl	No	All	0.298
RF-optimized	RF	MoM	ASGl	Yes	All	0.295
UMIDAS	UMIDAS	MoM	ASGl	-	All	0.472
Almon-ASGL	ALMON	MoM	ASGl & Almon	-	All	0.442
Almon	ALMON	MoM	Almon	-	All	0.745

The models marked in bold in Table 8.1 will be examined in more detail and compared to each other. Going forward, these will be simply referred to as the XGB, RF, UMIDAS and Almon models. The first intention was to include *Almon*, instead of *Almon - ASGL*, but since the performance of *Almon-ASGL* was much better, it was selected instead. The other models will often be used as benchmarks to investigate and explain different topics. Further investigations regarding optimization of the hyperparameters for XGB and RF will be done in Chapter 8.2.4.

8.2 Performance of models

8.2.1 In depth performance of models

A key objective of this thesis is to evaluate if the models are successful in estimating the release value of the U.S. inflation evaluated at daily frequency. To make this evaluation easier, they are put in relation to the simple AR model. Hence, the results will focus on comparing the different model's performance relative to the performance of the AR model.

Below follows plots of the true inflation and the models nowcasts at daily resolution (Figure 8.1-8.3), a scatterplot of the true value compared to the nowcasts (Figure 8.4) at horizon one and a table showing each model's relative RMSE to the AR model (Table 8.2).

Due to having multiple models, the result is presented in three separate plots for easier viewing. The first plot shows only AR, XGB and RF, while the other shows AR, UMIDAS and Almon. The AR model is included in both to act as a guide to simplify comparisons. The last plot shows all models, and shows only nowcasts made between 2019 and 2020 for a more detailed view of how the models compare.

Figure 8.1: *Nowcasts of the AR, XGB and RF models compared to the actual year-on-year percentage change over the two periods.*

Figure 8.2: *Nowcasts of the AR, UMIDAS and Almon models compared to the actual year-on-year percentage change over the two periods.*

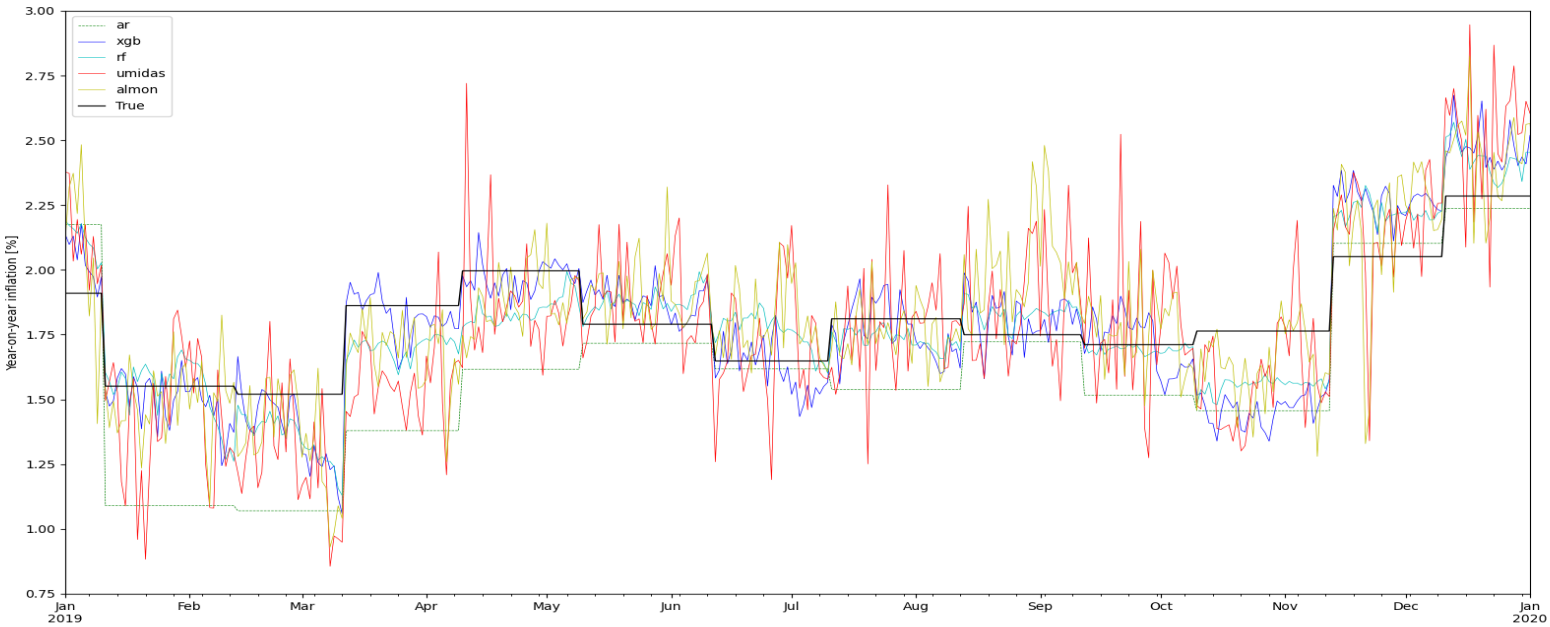

Figure 8.3: *Nowcasts of all models compared to the actual year-on-year percentage change over the two periods. Zoomed in on nowcasts made between 2019 and 2020.*

From comparing Figure 8.1 and 8.2 it can be seen that XGB and RF do a better job at nowcasting the inflation compared to UMIDAS and Almon. Overall, all models follow the inflation but UMIDAS and Almon have higher variance and deviates a lot at some points while XGB and RF tends to stay closer. Neither of the models are able to keep up when the inflation starts to rise heavily in 2021 and also have a hard time when the inflation begins to fall in the middle of 2022.

Looking at Figure 8.3 it is easier to see how the models compare to the AR model. The AR model is slow moving and updates only when a new value is released. The UMIDAS and Almon models behave similar and are fast

moving, but this comes with the cost of high variance. The XGB and RF models are also faster moving, but does not tend to overshoot as much. An explanation for the model's tendency to vary from day to day is that each day corresponds to a different horizon and for every horizon the data available differs since feature selection is done on a per-horizon basis. From the plots it is hard to see if the models improve closer to the release of the next inflation value, but more on this in Chapter 8.2.2.

Figure 8.4 shows performance of the models as a scatter plot where the x-axis is the actual yearly percentage change in inflation and the y-axis is the nowcasted value. The plot only includes the nowcast made the day before the release of the next inflation value. This means that the models have as much information as possible available.

Looking at Figure 8.4, is can be observed that models do a good job of nowcasting on horizon one, with an even spread of too high and too low nowcasts. Each of the models performs the best when the actual value is around 2%, which is also the most common true year-on-year percentage change inflation. Looking at the bottom left and top right, it is seen that the models tend to systematically underestimate the actual value when it is either very low or very high.

Table 8.2: *Relative RMSE for the models compared to the benchmark AR model.*

Modell	**RMSE**	**Relative RMSE**
AR	0.379	1.00
XGB	**0.269**	**0.71**
RF	0.295	0.78
UMIDAS	0.472	1.25
Almon	0.442	1.17

To summarize the relative RMSE between models is presented in Table 8.2. There it can be seen that the best model is XGB with a relative RMSE of

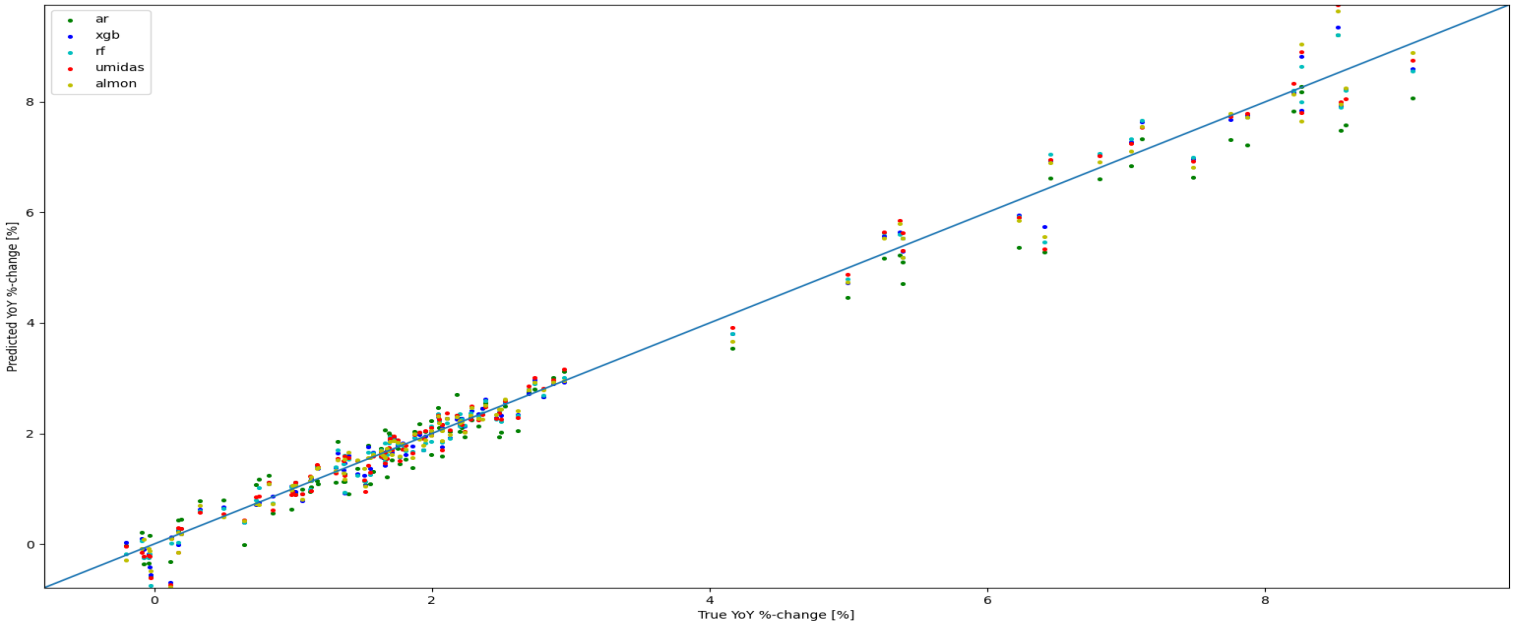

Figure 8.4: *Scatter plot of nowcasts (y-axis) made at the day before the release of the next inflation value, horizon 1, against actual inflation value (x-axis). The diagonal line represents a correct estimation.*

0.71 compared to the AR model. UMIDAS has the highest relative RMSE and is 25% worse than the AR model. An possible explanation for UMIDAS and Almon's poor performance is that they are over parameterized and might have benefitted from a lower number of features. This theory is supported by the fact that Almon performs better than UMIDAS and Almon has fewer parameters.

Overall XGB and RF are successful in estimating the release value of U.S. inflation evaluated at daily frequency, with XGB being the best model.

8.2.2 How does the performance vary on different horizons?

Since this thesis is using real time data, more data gets available for the models when the nowcasting date is approaching the release date of the inflation. It is thus expected that the models improve over time. Further, since the models are evaluated on a daily basis, it is possible to investigate if this is actually the case. This is shown in Figure 8.5, where the RMSE for every specific horizon is shown.

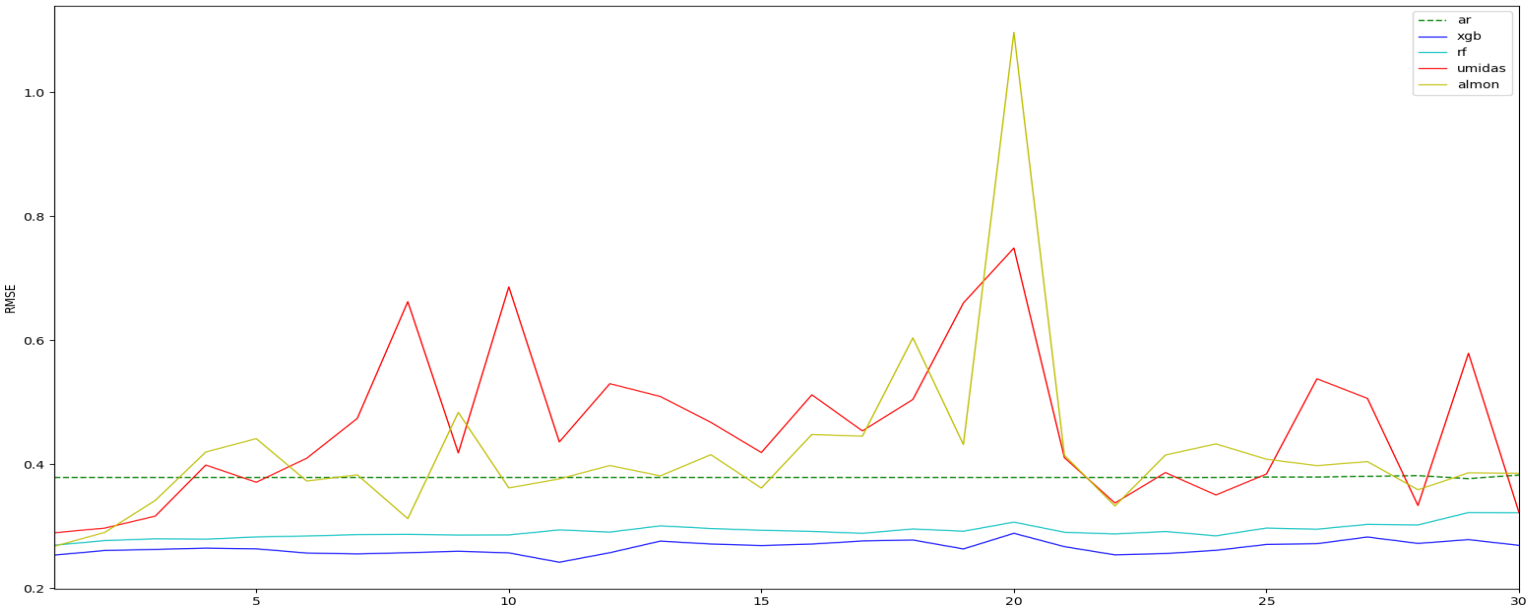

Figure 8.5: *RMSE for horizons one to 30 for the models, showing how the models' accuracy changes over time*

The ML (XGB, RF) model's RMSE increase only slightly as the horizon increases. The slight increase supports the expected result that the model

should improve, the more it gets available. However, since the increase is quite small, there is probably room for improvement in terms of how new data is incorporated.

While the RMSE of the econometric models (UMIDAS, Almon) is changing quite significantly at different horizons, there seems to be a more pronounced difference between higher and lower horizons compared to the ML models. Further, it is also interesting to see how some horizons perform very poorly. This could possibly partly be explained by the fact that these models seem to be slightly overparametrized. Another explanation could be that at each horizon, a separate ASGL feature selection is performed, meaning that different features could be selected and for some horizons this selection process could have been less successful. Looking at for example horizon 20, all models including the ML models perform slightly worse. However the performance decrease on UMIDAS and Almon is much larger and the ML models seem to be able to handle this much better.

Finally, the AR model exhibits a constant RMSE throughout all horizons. This is expected since each nowcasting period is between two releases of the inflation. Since the AR model does not incorporate any high frequency data, this means that it has the same data available throughout all horizons and will thus make the same nowcast.

To get an overview, the average RMSE of horizon 25-30 are compared with the average of horizon 1-5. This result is presented in Table 8.3 where it is clear that all models improve. Even the ML models which seemed to be constant throughout the period in Figure 8.5 now proves to actually improve between 5-10%. UMIDAS improves the most with 27%. However, UMIDAS is also the most volatile model and only looking at the improvement between the first and last 5 days, does not display the slightly worse performance on the horizons 10-20 that can be seen in Figure 8.5.

Table 8.3: *Average RMSE as well as improvement for nowcasts at horizon 1-5 compared with horizon 25-30*

Model	**Improvement**
RF	10%
XGB	5%
Almon	9%
UMIDAS	27%

8.2.3 Has US inflation become harder to nowcast?

As can be seen in Figure 8.1 and 8.2, the US inflation has been increasing a lot during the past years, 2021-. It is therefore interesting to investigate if the inflation has become harder to nowcast during these past years compared to the more stable period between 2014 and 2019. To do this, the RMSE between two periods are compared for four models. The result is displayed in Table 8.4.

Table 8.4: *RMSE for models during the period 2014-2019 as well as 2019-2023.*

Modell	**RMSE 2014-2019**	**Rank**	**RMSE 2019-2023**	**Rank**
AR	0.252	3	0.491	3
XGB	0.194	1	0.338	1
RF	0.220	2	0.371	2
UMIDAS	0.328	5	0.602	5
Almon	0.315	4	0.558	4

It is clear that the second period, between 2019 and 2023, is much harder for all models to nowcast. Since all models agree, it can be concluded that inflation has become harder to nowcast. It can also be seen that the models keeps their mutual rank even as the period changes.

Table 8.5: *RMSE for ML models with optimized as well as standard hyperparameter values as well as the percentage improvement*

Model	Standard (RMSE)	Optimized (RMSE)	Improvement
XGBoost	0.304	0.269	12%
Random Forest	0.298	0.295	1%

8.2.4 Does hyperparameter tuning improve the ML models

Today, many ML models are very accessible and relatively easy to implement using available packages. However, the models can still be improved by for example finding better hyperparameters and thus create a model more suited for the data at hand. This process can sometimes be time consuming, since there are often both many parameters, and many values for each parameter to try which results in a lot of different combinations to evaluate. For example, when implementing RF using **scikit-learn** in Python, there are 17 hyperparameters that can potentially be tuned. The standard values are carefully selected and often work well. It is thus interesting to see what potential improvement that may arise when actually tuning the parameters of the models, compared to using the standard ones.

Both XGB and RF were evaluated using its standard hyperparemeter values, as well as the resulting values after being tuned by cross validation. For more information about the standard values used, see the documentation for RF (Scikit-learn, 2023) and XGB (XGBoost, 2023). The results for can be seen in Table 8.5.

The result vary a lot between the two models. With an improvement of only around 1%, RF does not seem to improve a lot when optimizing the hyperparameters. Instead, XGB improves 12% which can be considered a lot. While this result indicates that hyperparameter tuning is probably not worth doing for RF and very important for XGB, there are many factors that

affect and should be considered.

First, the hyperparameter tuning is dependent on the researcher to select the best hyperparemeters to tune and to find a grid suitable for the problem, and is thus very sensitive to human error. It could for example be the case that some important hyperparameters were not tuned which could explain the scarse increase for RF.

A potential explanation for the result is the fact that RF has been around for a longer time and is more established. This means that it has been tested more and the standard parameters could be selected in a better way. Scikit-learn has also developed ways to select the standard parameters to be suited to the data set at hand, instead of always using the same value. For example, the important hyperparameter *max_depth* is selected in a dynamic way for RF (Scikit-learn, 2023) while it is defaulted to 6 in XGB (XGBoost, 2023).

Finally, the result is specific to the problem of nowcasting inflation and could be different for other data sets. However, the results should still give some guidance and inspiration to what could work for future problems.

8.3 What variables are important when nowcasting inflation?

Another interesting topic to investigate is what variables that are actually included in the final model, and thus considered relevant by ASGL. A variable is considered included if any of its time lags were included as a feature. Since a separate ASGL was performed for each horizon, features are selected once for every horizon. It is therefore possible to look at the results for all horizons and see what variables are selected. While horizons over 30 are less common, the ASGL was performed for horizons up until 43 to include all special cases. The maximum number of times a variable can be included is thus 43. The top

10 occurring variables for all horizons are displayed in Table 8.6 for 2014-2019 and 8.7 for 2019-2023.

Table 8.6: *Ten most used variables and how many horizons they are used in for the period between 2014 and 2019.*

Code	**Occurrences**
CPI-U: All Items in U.S.	43
Conventional Gasoline Prices	43
Cass Freight Index: Shipments	41
U.S. National Home Price Index	41
Dow Jones Index	40
WTI oil	39
CPI: Final Finished Goods	36
CPI: Metals and Metal Products	32
Corn Price	30
Federal Funds Effective Rate	30

Table 8.7: *Ten most used variables and how many horizons they are used in for the period between 2019 and 2023.*

Code	**Occurrences**
CPI-U: All Items in U.S.	43
Conventional Gasoline Prices	43
Cass Freight Index: Shipments	42
U.S. National Home Price Index	39
WTI oil	39
Dow Jones Index	38
CPI: Metals and Metal Products	37
Corn Price	34
CPI-U: All Items Less Food	31
Bank Prime Loan Rate	30

Both the inflation itself and the gasoline price were selected by ASGL at all horizons. Inflation is of course a good predictor of itself and this result was very expected. Further, other sub-indexes of inflation are included, which also makes sense. Then both oil and gasoline prices are included as well and these are often used in models for nowcasting inflation. U.S. Home price index is also used in both periods, and this is logical since cost of living makes up a large proportion of the CPI. More surprising results are the inclusion of corn price and shipment index. Many commodity prices were available for the model to select from, so the choice of corn was unexpected. Shipment index was not a variable we expected to have high explanatory power but seems to been helpful for the models. Overall the selection of varaibles were expected and it seems like ASGL does a good job. For an overview of all variables, see Appendix A.

Looking at Table 8.6 and 8.7 it seems like the same variables are important for nowcasting inflation indepent of which period is the focus.

8.3.1 Is the inclusion of high frequency variables valuable for the model?

Another question to answer is if the addition of higher frequency variables is necessary for the model. Necessary for the model can be interpreted in two ways: are higher frequency variables used in the model (and thus considered relevant according to ASGL) and do they affect the performance. If higher frequency are unnecessary, the model can be simplified by not using the MIDAS framework. The following two figures showcase the total number of variables used per horizon, but also the mutual split between different frequencies.

As can be seen in Figures 8.6 and 8.7 neither period utilizes a large number of weekly variables, often only one but sometimes none and two weekly variables are used for a single horizon in period two. Looking at daily variables, the story is different. The first five horizons use few daily variables, but after this point almost as many daily and monthly variables are used. Here it is worth remembering that the information contained in daily variables in lag one to around 15 is the same, since only daily variables from the target month is used. Another interesting observation is that few variables are used in the first horizons to then grow and reach a maximum and then decline afterward. By looking at Figure 8.6 and 8.7, one can draw the conclusion that higher frequency variables are of use to the model, allegedly the weekly variables may be omitted since so few are used.

A test is conducted to determine the higher frequency variables role in the models' performance. Two models are used for this test: the first being XGB with default hyperparameters using the full dataset and the second being XGB with default hyperparameters using a dataset with only monthly variables. The RMSE for the models can be seen in Table 8.8

As can be seen in Table 8.8, the addition of higher frequency variables in-

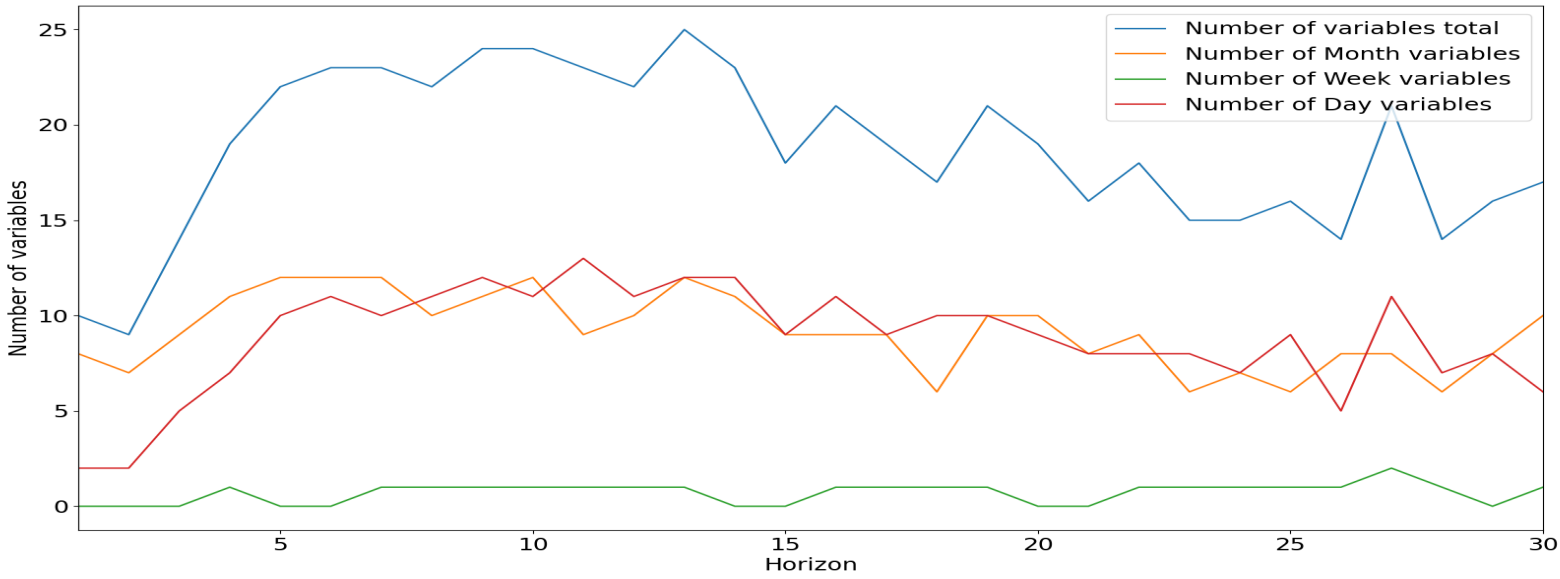

Figure 8.6: *Plot showing the total number of variables used on each horizon and the mutual split between frequencies for the first period.*

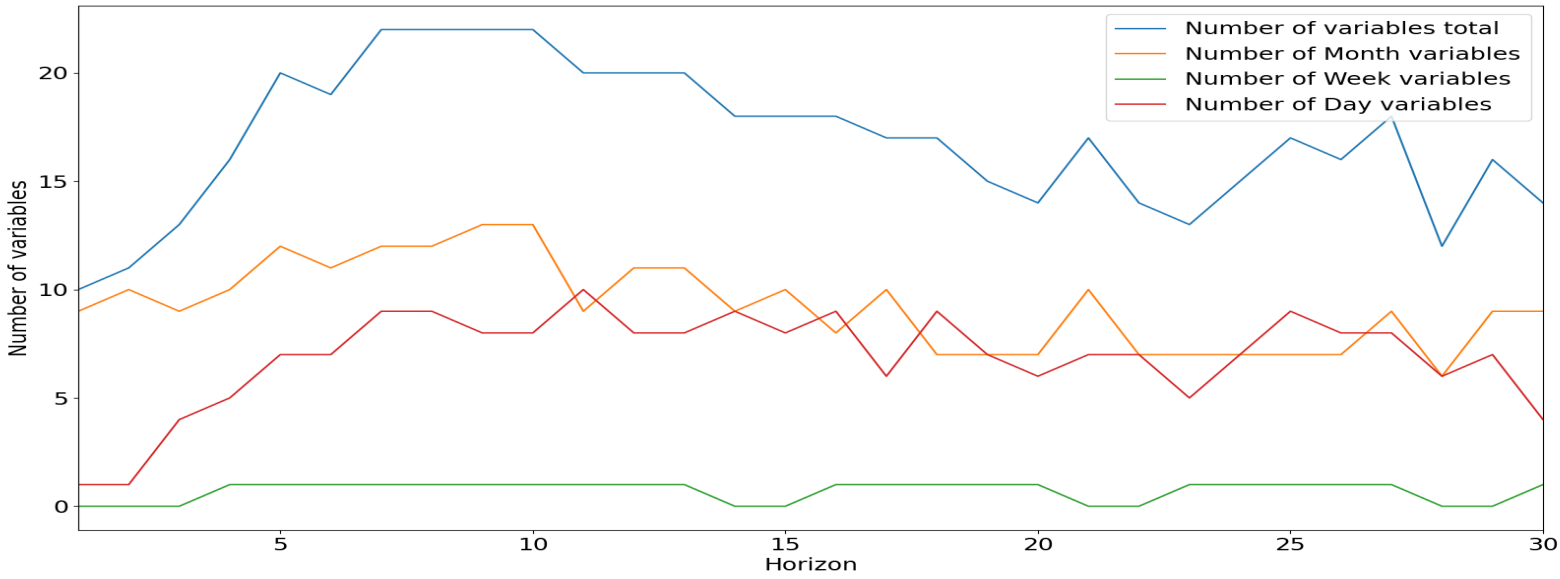

Figure 8.7: Plot showing the total number of variables used on each horizon and the mutual split between frequencies for the second period.

Table 8.8: *RMSE for 2 XGB models, where one is able to use all variables available, and the other is limited to variables released at monthly frequency.*

Variables available	RMSE
All	0.304
Only monthly	0.349

creases the model's performance with around 13%. This result combined with what is shown in Figure 8.6 and 8.7 helps us draw the conclusion that the addition of higher frequency variables is necessary for the model's performance. If it is worth implementing the MIDAS framework to reach these extra percent in performance is left to the reader to decide upon.

Chapter 9

Conclusion

In this section the main results and conclusions of Chapter 8 above will be summarized. It also attempts to answer the questions posed in Chapter 1.3 regarding the model's performance and used variables. Finally, some suggestions on further research and improvements will be made.

9.1 Performance of models evaluated at daily frequency

Based on the results of this thesis, the ML models outperforms the AR model, while the econometric models do not. Both machine learning models accomplices to produce a RMSE that is lower for both periods, with XGB being the better model. XGB produces a RMSE that is 29% lower than that of the AR model while RF produces a RMSE that is 22% lower. For UMIDAS and Almon the RMSE is 25% respectively 17% higher than for the AR model. Visual inspection of the models in Figure 8.1 and 8.2 shows how the machine learning models have a lower variance compared to UMIDAS and Almon. Nevertheless, none of the models are able to keep up with the

rapid increase beginning in 2021. A possible explanation of UMIDAS and Almons shortcomings is that they are overparameterized and would have benefitted from using fewer features.

When comparing XGB and RF with and without optimized hyperparameters in Table 8.5, it is observed that optimization is more important for XGB than RF. XGB improves its RMSE by 12% after optimization, while it is only an 1% increase for RF.

The third part of evaluating the model's success is their ability to incorporate real-time data releases and improve their estimation as more data is available. If the models are successful in incorporating real-data and using it, the model's RMSE should increase as the horizon increases and have its lowest value for horizon 1. This behavior is not clearly showcased when examining the models. Only RF showcases a clear linear increase in RMSE as the horizon increases. XGB has a linear increasing trend, but at specific horizons the RMSE is lower than compared to previous horizons. RMSE for Almon and UMIDAS changes a lot between horizons, but a linear trend can be observed for horizons between 1 and 20. UMIDAS performs almost the same at horizon 1 and 30. All models except XGB has its lowest RMSE at horizon 1, for XGB it is horizon 11.

In a simplified explanation, it can be said that the models do an ok job incorporating real-time data and improving nowcast as more data becomes available, since a majority of the models have their lowest RMSE at horizon 1 and some linear trends can be observed.

The results of this thesis suggest that variable and feature selection with ASGL followed by estimation with XGB is a good approach for nowcasting inflation in the U.S. It outperforms the benchmark AR model and the other models.

9.2 Selected variables

Another importance area of investigation for this thesis is also which data series are deemed valuable for the model and if it has changed over time. As seen in Table 8.6 and 8.7 the top ten used variables are almost the same for both periods, suggesting that some variables are always important when nowcasting inflation. A possible explanation for this is that inflation is a measurement of changes in prices, and the selected variables all have an important connection to prices in the U.S. For example, Pasaogullari and Waiwood (2014) shows that oil prices have predictive importance on short-term movements in inflation.

Regarding the importance of higher frequency variables, it is shown that at least the daily variables bear high importance. As seen in Figure 8.6 and 8.7 almost as many daily variables as monthly variables are used for most horizons. Higher frequency variables also play a role in obtaining lower RMSE, as seen in 8.8 where a model limited to only monthly data performs worse than one with both monthly and higher frequency data.

To summarize, using the MIDAS framework to incorporate higher frequency data and allowing the use of real time data is important for the model's success. It is also shown that the variables deemed important for nowcasting inflation are almost constant throughout time.

9.3 Further research

When writing this thesis, multiple ideas have come up that would be interesting to investigate but were not persuaded due to them being out of scope or due to limitations in time. Some are possible improvement on the models, while others are stand-alone ideas.

- Construct a web-scraper that collects daily price data from large retail

stores and websites with housing costs. Something along the lines of Cavallo and Rigobon (2016) and see if this data would help the nowcasts. In general, finding new data that might improve the nowcast could be interesting since most models use standard economic variables.

- Investigate if daily variable selection instead of doing it once per period would have improved performance. This was not conducted by us due to limitations in time and computational power. By using an implementation of ASGL supporting *warm start* it would be feasible.
- Test if the performance of the Almon and UMIDAS could be improved through careful selection of predictive series and using fewer features in an attempt to avoid overparameterization. An idea would be to use the most selected variables found in this thesis.
- Investigate if the models could be improved by creating separate models for some of the input variables as well. For example, if a variable is not released at a given date, a separate model could give an estimate of that variable which could then be used in the inflation nowcast.

Appendix A

Data

A.1 Data set

Table A.1: The complete data set used in the predictive models. Including code, description, frequency and source for each series

Index	Code	Description	Freq	Source
1	T1YFF	1-Year Treasury Minus Federal Funds Rate	Day	A
2	T10YFF	10-Year Treasury Minus Federal Funds Rate	Day	A
3	T5YFF	5-Year Treasury Minus Federal Funds Rate	Day	A
4	DFF	Federal Funds Effective Rate	Day	A
5	DGS1	Market Yield on U.S. Treasury Securities at 1-Year	Day	A
6	DGS10	Market Yield on U.S. Treasury Securities at 10-Year	Day	A
7	DGS5	Market Yield on U.S. Treasury Securities at 5-Year	Day	A
8	DGS3MO	Market Yield on U.S. Treasury Securities at 3-Month	Day	A
9	DAAA	Moody's Seasoned Aaa Corporate Bond Yield	Day	A
10	DBAA	Moody's Seasoned Baa Corporate Bond Yield	Day	A
11	JO1 Comdty	Orange juice	Day	B
12	LC1 Comdty	Kettle	Day	B

Table A.1: The complete data set used in the predictive models. Including code, description, frequency and source for each series

Index	Code	Description	Freq	Source
13	DJITR Index	Dow Jones Index	Day	B
14	KC1 Comdty	Coffe	Day	B
15	HG1 Comdty	Copper	Day	B
16	C 1 Comdty	Corn	Day	B
17	CCMP Index	Nasdaq Index	Day	B
18	SPXT Index	S&P 500 Index	Day	B
19	LB1 Comdty	Lumber	Day	B
20	CL1 Comdty	WTI oil	Day	B
21	DGASUSGULF	Conventional Gasoline Prices: U.S. Gulf Coast, Regular	Day	A
22	CUUR0000SEHA	CPI-U: Rent of Primary Residence in U.S. City Average	Month	A
23	CSUSHPINSA & S	P/Case-Shiller U.S. National Home Price Index	Month	A
24	TB3SMFFM	3-Month Treasury Bill Minus Federal Funds Rate	Month	A
25	TB6SMFFM	6-Month Treasury Bill Minus Federal Funds Rate	Month	A
26	USCONS	All Employees, Construction	Month	A
27	DMANEMP	All Employees, Durable Goods	Month	A
28	USFIRE	All Employees, Financial Activities	Month	A
29	USGOOD	All Employees, Goods-Producing	Month	A
30	USGOVT	All Employees, Government	Month	A
31	MANEMP	All Employees, Manufacturing	Month	A
32	CES1021000001	All Employees, Mining	Month	A
33	NDMANEMP	All Employees, Nondurable Goods	Month	A
34	USTRADE	All Employees, Retail Trade	Month	A
35	SRVPRD	All Employees, Service-Providing	Month	A
36	PAYEMS	All Employees, Total Nonfarm	Month	A
37	USTPU	All Employees, Trade, Transportation, and Utilities	Month	A
38	USWTRADE	All Employees, Wholesale Trade	Month	A
39	CES2000000008	Avg Hourly Earnings of Production, Construction	Month	A

Table A.1: The complete data set used in the predictive models. Including code, description, frequency and source for each series

Index	Code	Description	Freq	Source
40	CES3000000008	Avg Hourly Earnings of Production, Manufacturing	Month	A
41	AWHMAN	Avg Weekly Hours of Production, Manufacturing	Month	A
42	AWOTMAN	Avg Weekly Overtime Hours of Production, Manufacturing	Month	A
43	UEMPMEAN	Avg Weeks Unemployed	Month	A
44	CUMFNS	Capacity Utilization: Manufacturing (SIC)	Month	A
45	CLF16OV	Civilian Labor Force Level	Month	A
46	BUSLOANS	Commercial and Industrial Loans, All Commercial Banks	Month	A
47	DTCOLNVHFNM	Consumer Motor Vehicle Loans Owned by Finance Companies	Month	A
48	CPIAUCNS	CPI-U: All Items in U.S. City Average	Month	A
49	CPIULFSL	CPI-U: All Items Less Food in U.S. City Average	Month	A
50	CUSR0000SA0L5	CPI-U: All Items Less Medical Care in U.S. City Average	Month	A
51	CUSR0000SA0L2	CP-U: All Items Less Shelter in U.S. City Average	Month	A
52	CPIAPPSL	CPI-U: Apparel in U.S. City Average	Month	A
53	CUSR0000SAC	CPI-U: Commodities in U.S. City Average	Month	A
54	CUSR0000SAD	CPI-U: Durables in U.S. City Average	Month	A
55	CPIMEDSL	CPI-U: Medical Care in U.S. City Average	Month	A
56	CUSR0000SAS	CPI-U: Services in U.S. City Average	Month	A
57	CPITRNSL	CPI-U: Transportation in U.S. City Average	Month	A
58	CE16OV	Employment Level	Month	A
59	FEDFUNDS	Federal Funds Effective Rate	Month	A
60	INDPRO	Industrial Production: Total Index	Month	A
61	M2SL	M2	Month	A
62	ACOGNO	Manufacturers' New Orders: Consumer Goods	Month	A
63	DGORDER	Manufacturers' New Orders: Durable Goods	Month	A
64	ANDENO	Manufacturers' New Orders: Nondefense Capital Goods	Month	A
65	AMDMUO	Manufacturers' Unfilled Orders: Durable Goods	Month	A
66	BOGMBASE	Monetary Base; Total	Month	A

Table A.1: The complete data set used in the predictive models. Including code, description, frequency and source for each series

Index	Code	Description	Freq	Source
67	AAAFFM	Moody's Aaa Corporate Bond Minus Federal Funds Rate	Month	A
68	PERMIT	New Privately-Owned Housing Units Authorized: Total Units	Month	A
69	HOUST	New Privately-Owned Housing Units Started: Total Units	Month	A
70	NONREVSL	Nonrevolving Consumer Credit Owned and Securitized	Month	A
71	PCEPI	Personal Consumption Expenditures: Chain-type Price Index	Month	A
72	DDURRG3M086SBEA	Personal consumption expenditures: Durable goods	Month	A
73	DNDGRG3M086SBEA	Personal consumption expenditures: Nondurable goods	Month	A
74	DSERRG3M086SBEA	Personal consumption expenditures: Services	Month	A
75	WPSFD49207	CPI by Commodity: Final Finished Goods	Month	A
76	WPSFD49502	CPI by Commodity: Final Personal Consumption	Month	A
77	WPSID61	CPI by Commodity: Intermediate Processed Goods	Month	A
78	WPSID62	CPI by Commodity: Intermediate Unprocessed Goods	Month	A
79	PPICMM	CPI by Commodity: Metals and Metal Products	Month	A
80	REALLN	Real Estate Loans, All Commercial Banks	Month	A
81	M2REAL	Real M2 Money Stock	Month	A
82	DPCERA3M086SBEA	Real personal consumption expenditures	Month	A
83	RPI	Real Personal Income	Month	A
84	W875RX1	Real personal income excluding current transfer receipts	Month	A
85	BUSINV	Total Business Inventories	Month	A
86	ISRATIO	Total Business: Inventories to Sales Ratio	Month	A
87	DTCTHFNM	Total Consumer Loans and Leases	Month	A
88	UNRATE	Unemployment Rate	Month	A
89	UMCSENT	University of Michigan: Consumer Sentiment	Month	A
90	STICKCPI	Sticky Price Consumer Price Index	Month	A
91	FRGSHPUSM649NCIS	Cass Freight Index: Shipments	Month	A
92	WTB3MS	3-Month Treasury Bill Secondary Market Rate, Discount	Week	A
93	WTB6MS	6-Month Treasury Bill Secondary Market Rate, Discount	Week	A

Table A.1: The complete data set used in the predictive models. Including code, description, frequency and source for each series

Index	Code	Description	Freq	Source
94	WPRIME	Bank Prime Loan Rate	Week	A
95	ICSA	Initial Claims Jobs	Week	A
96	SBCACBW027SBOG	Securities in Bank Credit, All Commercial Banks	Week	A
97	MORTGAGE30US	30-Year Fixed Rate Mortgage Average in the United States	Week	A
98	DFII10	Market Yield 10-Year, Inflation-Indexed	Day	A
99	DFII30	Market Yield 30-Year, Inflation-Indexed	Day	A
100	DFII5	Market Yield 5-Year, Inflation-Indexed	Day	A

A.1.1 Source

A = *ALFRED* (Archival FRED provided by FED St. Louis).

B = *Bloomberg*

Bibliography

Abdi, H. and Williams, L. J. (2010). Principal component analysis. *WIREs Computational Statistics*, 2(4):433–459.

Adrian, T., Boyarchenko, N., and Giannone, D. (2019). Vulnerable growth. *American Economic Review*, 109(4):1263–89.

Almon, S. (1965). The distributed lag between capital appropriations and expenditures. *Econometrica*, 33(1):178–196.

Andreou, E., Ghysels, E., and Kourtellos, A. (2013). Should macroeconomic forecasters use daily financial data and how? *Journal of Business & Economic Statistics*, 31(2):240–251.

Asimakopoulos, S., Paredes, J., and Warmedinger, T. (2013). Forecasting fiscal time series using mixed frequency data. *ECB papers, Working Paper No. 1550*.

Babii, A., Ghysels, E., and Striaukas, J. (2022). Machine learning time series regressions with an application to nowcasting. *Journal of Business & Economic Statistics*, 40(3):1094–1106.

Baffigi, A., Golinelli, R., and Parigi, G. (2004). Bridge models to forecast the euro area gdp. *International Journal of Forecasting*, 20(3):447–460.

Bańbura, M., Giannone, D., Modugno, M., and Reichlin, L. (2013). Chapter 4 - now-casting and the real-time data flow. *Handbook of Economic Forecasting*, 2:195–237.

Bokati, L. and Kreinovich, V. (2022). Why exponential almon lag works well in econometrics: An invariance-based explanation. *Departmental Technical Reports (CS)*.

Breiman, L. (1984). *Classification and regression trees*. Routledge.

Breiman, L. (2001). Random forests. *Machine learning*, 45:5–32.

Brownlee, J. (2016). Xgboost python api reference. *Machine Learning Mastery*.

Brownlee, J. (2018). A gentle introduction to k-fold cross-validation. *Machine learning mastery*, 2019.

Bureau of Labor Statistics (2023a). Consumer price index- U.S. bureau of labor statistics. https://www.bls.gov/cpi/.

Bureau of Labor Statistics (2023b). Consumer price index overview- U.S. bureau of labor statistics. https://www.bls.gov/cpi/overview.htm.

Cavallo, A. and Rigobon, R. (2016). The billion prices project: Using online prices for measurement and research. *Journal of Economic Perspectives*, 30(2):151–78.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 785–794. Association for Computing Machinery.

Choi, H. and Varian, H. (2012). Predicting the present with google trends. *Economic record*, 88:2–9.

Clark, T. E., Huber, F., Koop, G., and Marcellino, M. (2022). Forecasting us inflation using bayesian nonparametric models. *Federal Reserve Bank of Cleveland, Working Paper No. 22-05*.

Dauphin, J.-F., Dybczak, K., Maneely, M., Sanjani, M.-T., Suphaphiphat, N., Wang, Y., and Zhang, H. (2022). *Nowcasting GDP-A Scalable Approach Using DFM, Machine Learning and Novel Data, Applied to European Economies*. International Monetary Fund, Workning Paper No. 2022/052.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *Ann. Statist.*, 32(2):407–499.

Eric Ghysels, M. M. (2018). *Applied Economic Forecasting using Time Series Methods*. Oxford University Press.

Foroni, C. and Marcellino, M. G. (2013). A survey of econometric methods for mixed-frequency data. *Norges Bank Research*.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*.

Ghysels, E., Kvedaras, V., and Zemlys, V. (2016). Mixed frequency data sampling regression models: the R package midasr. *Journal of statistical software*.

Ghysels, E., Santa-Clara, P., and Valkanov, R. (2006). Predicting volatility: getting the most out of return data sampled at different frequencies. *Journal of Econometrics*, 131(1-2):59–95.

Giannone, D., Reichlin, L., and Small, D. (2008). Nowcasting: The real-time informational content of macroeconomic data. *Journal of Monetary Economics*, 55(4):665–676.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Kitchen, J. and Monaco, R. (2003). Real-time forecasting in practice: The U.S. treasury staff's real-time gdp forecast system. *Business Economics*, 38(4):10–19.

Knotek, E. S. and Zaman, S. (2017). Nowcasting us headline and core inflation. *Journal of Money, Credit and Banking*, 49(5):931–968.

Knotek II, E. S. and Zaman, S. (2022). Real-time density nowcasts of us inflation: A model combination approach. *International Journal of Forecasting*.

Kuzin, V., Marcellino, M., and Schumacher, C. (2011). Midas vs. mixed-frequency var: Nowcasting gdp in the euro area. *International Journal of Forecasting*, 27(2):529–542.

Lindgren, G., Rootzén, H., and Sandsten, M. (2013). *Stationary stochastic processes for scientists and engineers*. CRC press.

Mass, C., Mass, C. F., et al. (2011). Nowcasting: The next revolution in weather prediction. *Bulletin of the American Meteorological Society*, pages 1–23.

Mendez-Civieta, A., Aguilera-Morillo, M. C., and Lillo, R. E. (2021). Adaptive sparse group lasso in quantile regression. *Advances in Data Analysis and Classification*, 15(3):547–573.

Méndez Civieta, Á., Aguilera-Morillo, M. C., and Lillo, R. E. (2021). Asgl: A python package for penalized linear and quantile regression. *arXiv e-prints*, pages arXiv–2111.

Michael W. McCracken, S. N. (2015). Fred-md: A monthly database for macroeconomic research. *Journal of Business & Economic Statistics*.

Modugno, M. (2013). Now-casting inflation using high frequency data. *International Journal of Forecasting*, 29(4):664–675.

Mogliani, M. and Simoni, A. (2021a). Bayesian midas penalized regressions: Estimation, selection, and prediction. *Journal of Econometrics*, 222(1, Part C):833–860.

Mogliani, M. and Simoni, A. (2021b). Bayesian midas penalized regressions: estimation, selection, and prediction. *Journal of Econometrics*, 222(1):833–860.

Monteforte, L. and Moretti, G. (2010). Real time forecasts of inflation: the role of financial variables. Temi di discussione (Economic working papers) 767, Bank of Italy, Economic Research and International Relations Area.

Monteforte, L. and Moretti, G. (2013). Real-time forecasts of inflation: The role of financial variables. *Journal of Forecasting*, 32(1):51–61.

Moré, J. J. (1978). The levenberg-marquardt algorithm: Implementation and theory. *Numerical Analysis*, pages 105–116.

Nystrup, P., Lindström, E., Møller, J. K., and Madsen, H. (2021). Dimensionality reduction in forecasting with temporal hierarchies. *International Journal of Forecasting*, 37(3):1127–1146.

Pasaogullari, M. and Waiwood, P. (2014). Do oil prices predict inflation? *Economic Commentary*, (Feb.):1.

Pettenuzzo, D., Timmermann, A., and Valkanov, R. (2016). A midas approach to modeling first and second moment dynamics. *Journal of Econometrics*, 193(2):315–334. The Econometric Analysis of Mixed Frequency Data Sampling.

Phelps, E. S. (1967). Phillips curves, expectations of inflation and optimal unemployment over time. *Economica*, 34(135):254–281.

Prashant, B. (2023). A guide on xgboost hyperparameters tuning. https://xgboost.readthedocs.io/en/stable/python/python_api.html.

Rünstler, G. and Sédillot, F. (2003). Short-term estimates of euro area real gdp by means of monthly data. Technical report, ECB working paper No. 276.

Schumacher, C. (2014). Midas and bridge equations. *Available at SSRN 2797010*.

Scikit-learn (2023). Random forest regressor. https://scikitlearn.org/stable/modules/generated/sklearn.ensemble .RandomForestRegressor.html.

Sharoon, S. (2020). A beginner's guide to random forest hyperparameter tuning. *Analyticsvidhya*.

Stock, J. H. and Watson, M. W. (1989). New indexes of coincident and leading economic indicators. *NBER macroeconomics annual*, 4:351–394.

Stock, J. H. and Watson, M. W. (2006). Why has U.S. inflation become harder to forecast? Technical Report 12324, National Bureau of Economic Research.

Sun, T. and Zhou, Z.-H. (2018). Structural diversity for decision tree ensemble learning. *Frontiers of Computer Science*, 12:560–570.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Tretina, K. (2022). What is transitory inflation? *Forbes*. https://www.forbes.com/advisor/investing/transitory-inflation/.

Venkat, N. (2018). The curse of dimensionality: Inside out. *Pilani (IN): Birla Institute of Technology and Science, Pilani, Department of Computer Science and Information Systems*, 10.

XGBoost (2023). Xgboost python package. https://xgboost.readthedocs.io/en/stable/python/python$_i$*ntro.html*.

Xu, Q., Zhuo, X., Jiang, C., and Liu, Y. (2019). An artificial neural network for mixed frequency data. *Expert Systems with Applications*, 118:127–139.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320.