MASTER'S DEGREE

DATA ANALYTICS FOR BUSINESS

MASTER'S FINAL WORK

DISSERTATION

CREATING A LARGE DATA SET FOR FORECASTING INFLATION IN THE EUROPEAN CONTEXT USING MACHINE LEARNING TECHNIQUES

VASCO COSTA LEAL

OCTOBER 2024

MASTER'S DEGREE

DATA ANALYTICS FOR BUSINESS

MASTER'S FINAL WORK

DISSERTATION

CREATING A LARGE DATA SET FOR FORECASTING INFLATION IN THE EUROPEAN CONTEXT USING MACHINE LEARNING TECHNIQUES

VASCO COSTA LEAL

SUPERVISION:

PROF. ADRIANA CORNEA-MADEIRA

OCTOBER 2024

Abstract

Inflation forecasting plays a critical role in macroeconomic analysis due to its impact on economic growth. The main contribution of this dissertation is to extend the literature by increasing the data availability to the European context, building the macroeconomic database EUED from multiple Eurostat sources. The second contribution is to use the newly built rich database EUED to forecast inflation in Europe. The main conclusion is that the machine learning models led to more accurate results compared to the benchmarks (the random walk and the autoregressive model).

Keywords:

Inflation forecasting, HICP, EUED, machine learning, LASSO, Elastic-net, medallion architecture

Acknowledgements

A realização desta dissertação não seria possível sem o contributo de diversas pessoas. Neste sentido, gostaria de agradecer à professora Andrea Cornea-Madeiros, pela sua contribuição e orientação que se fez sentir ao longo deste trabalho. À minha namorada, pela paciência, amor e cuidado. Por ser o meu braço direito e por me lembrares da importância de continuar a lutar pelos meus objetivos. Sem o seu apoio esta jornada não seria a possível. À minha mãe, ao meu pai e ao meu avô, por serem uma fonte constante de inspiração e força, por terem estado sempre do meu lado e acreditarem no meu potencial. E por fim, aos meus amigos, pelas longas conversas, pela partilha de conhecimento e pelo companheirismo.

Table of Contents

List of Tables

1. **Introduction**

Inflation can be defined as “the gradual loss of purchasing power, reflected in a broad rise in prices for goods and services” (McKinsey & Company, 2024). This means that with the same amount of money, it’s not possible to buy the same amount of goods and services as before (European Central Bank, n.d.).

The motivation for this study lies in the importance of forecasting inflation due to its uncertainty. Forecasting inflation is a critical task in macroeconomic analysis with far-reaching implications for policy decisions, business strategies and consumers behavior (Gafurdjan, 2024).

Accurate and timely inflation predictions are recommended to support central banks and governments adjust their policies in order to help them pursue price stability, which in turn creates an environment that is favorable for short and long term economic growth (Mandeya & Ho, 2021).

A key inspiration for this work comes from the article *"Forecasting Inflation in a Data-Rich Environment: The Benefits of Machine Learning Methods"* from Medeiros *et al.* (2019). These authors utilize the Federal Reserve Economic Data (FRED) monthly datasets of United States (U.S.) macroeconomic indicators to forecast the Consumer Price Index (CPI). They demonstrate that machine learning methods can outperform traditional econometric models in predicting inflation, highlighting the value of leveraging data-rich environments for more accurate forecasts. Therefore, this dissertation will not only continue but will also be grounded on their work. Building on this idea, this study contributes to literature in three ways.

First, while the previously mentioned paper focuses on forecasting United States (U.S.) inflation through the Consumer Price Index (CPI), our analysis shifts the focus to the European context by forecasting the Harmonized Index of Consumer Price (HICP). The HICP is a more relevant measure for Europe, because it gives a “comparable measure of inflation as they (prices) are calculated according to harmonized definitions” (European Union/Eurostat, 2018).

Second, our approach involves the construction of a comprehensive and unified macroeconomic dataset for Europe by aggregating multiple data sources from Eurostat

(n.d.). This dataset serves as a foundation not only for this study, but also as a valuable resource for future research on European inflation and broader macroeconomic trends.

Third, while that article implements its models and workflow in R, we have opted to implement the entire forecasting workflow in Python. This decision reflects both the popularity of Python in data science and the flexibility of its libraries for data engineering and forecasting tasks with machine learning. This allowed us to emphasize the reproducibility and adaptability of the code for future studies by making it as generic as possible.

The dissertation is organized as follows. Chapter 2 presents the second contribution, the dataset created, as well as the process to make it ready for our analysis. Chapter 3 exhibits the methodology used for the third contribution, the workflow in Python. Chapter 4 reviews the models used to predict inflation. Chapter 5 discusses the results. Chapter 6 concludes the dissertation.

2. Data

The dataset used in this study, which we named European Union Economic Data (EUED), is a contribution to the literature on inflation forecasting in the European context. The EUED is a monthly macroeconomic dataset designed for empirical analysis in data-rich environments. We built this dataset by aggregating multiple datasets retrieved from Eurostat's "European and National Indicators for Short-Term Analysis" database, which provides a wide range of economic indicators at the European level.

Unlike the Federal Reserve Economic Data (FRED) dataset used by Medeiros et al. (2019), which is updated in real-time, the EUED dataset was manually compiled. While this introduces limitations in terms of real-time updates, it also provides a highly customized dataset.

The original sample for the EUED dataset extends from January 1980 to December 2023, resulting in 528 observations and a total of 744 variables for each of four regions: European Union of 27 (EU27), Euro Area of 20 (EA20), Germany, and United Kingdom (UK). For each region, data from various indicators were collected in several forms of adjustment, which can be categorized by their degree of adjustment: seasonally and calendar adjusted (SCA), seasonally adjusted (SA), calendar adjusted (CA), not seasonally adjusted (NSA), and not adjusted (NA).

To manage and process the data efficiently, we implemented a Medallion Architecture approach (Data Bricks, n.d.), which consists of three distinct layers: bronze, silver, and gold. This layered approach allows for data processing and storage at each step, where transformations are well-organized, easily accessible, and replicable, ensuring traceability and flexibility. It also ensures that at any point in the analysis, data from a specific processing step can be revisited, adjusted, or reanalyzed. This architecture was applied individually to each region to account for differences in data availability.

- Bronze Layer: Contains the original version of the data extracted from Eurostat, without any transformation. Holds the full dataset of 528 observations and 744 variables across each of the four regions. This layer forms the foundation of the dataset in which all further transformations are applied, ensuring that the data can be revisited and analyzed in its most basic form.

- Silver Layer: The data in this layer has undergone initial filtering and cleaning. To narrow the focus of this study, we limited the data from the bronze layer to cover the period from January 2000 to December 2023, reducing the number of observations to 288 for each region. Additionally, variables with missing values for all observations in this period were removed. This results in a more refined and manageable dataset that is built upon the foundation established in the bronze layer and can serve as a starting point for future studies by allowing flexibility to explore various strategies to deal with the remaining missing values.

- Gold Layer: Holds the final version of the dataset used in the forecasting models. To prioritize data quality and focus on the Harmonized Index of Consumer Prices for all items (HICP_All), further filtering was applied. Variables with any missing values were removed, and only 3 out of 75 variables related to the indicator HICP were retained to represent the three different measures: index (which we named HICP_All_idx_NA), monthly rate (HICP_All_mor_NA), and annual rate (HICP_All_anr_NA). Additionally, when multiple versions of the same indicator exist with different degrees of adjustment, the version with the highest degree of adjustment is selected – (SCA > SA > CA > NSA > NA). However, these steps were carefully ordered to avoid unintended data loss, for example, if an indicator appears in both a seasonally adjusted variable and a not adjusted variable, but the seasonally adjusted contains missing values, we retained the not adjusted version to preserve the indicator. Otherwise, selecting an indicator with the highest adjustment first and removing those with missing values later could result in the loss of useful data.

To illustrate the impact of these transformations, Table 1 provides an overview of the number of observations (obs) and variables (vars) for each region across the three layers, as well as the HICP variables available for analysis in the final dataset.

Table 1. Medallion Architecture

Layer / Dataset / Region	Bronze Original	Silver Filtered	Gold Final	HICP Variables
EU27	528 obs 744 vars	288 obs 582 vars	288 obs 164 vars	HICP_All_idx_NA HICP_All_mor_NA
EA20	528 obs 744 vars	288 obs 597 vars	288 obs 177 vars	HICP_All_idx_NA HICP_All_mor_NA
Germany	528 obs 744 vars	288 obs 727 vars	288 obs 362 vars	HICP_All_idx_NA HICP_All_mor_NA HICP_All_anr_NA
UK	528 obs 744 vars	288 obs 449 vars	288 obs 6 vars	None

As shown in the table above, the number of variables and observations varies across the different regions in the bronze, silver, and gold layers. A notable finding is the absence of any valid HICP variables for the UK final dataset. While the original dataset contained 744 variables, this number was significantly reduced to just 6 variables in the final dataset and none of these remaining variables are related to the indicator HICP all items. Therefore, the UK region will be omitted in the following chapters, since its final dataset lacks the primary indicator that is central to this study's analysis on forecasting inflation in the European context.

3. Methodology

This chapter details the workflow, developed in Python, to forecast monthly inflation, which represents a contribution of this study, as mentioned in Chapter 0. 1. Introduction.

The steps covered in the workflow include data splitting, data standardization, lagged features, model selection, cross-validation for model tuning, recursive forecasting strategy, and model evaluation.

The workflow was designed to iterate over the three regions – EU27, EA20, and Germany – applying the Medallion Architecture approach, as discuss in Chapter 0. 2. Data, where the HICP all items indicator is the focus of this study, and it could be represented in 3 different types of inflation measures that could lead to 3 target variables to analyze.

3.1. Data Splitting

For each target variable in the set of available target variables for a specific region, the data in the gold layer was split into four distinct datasets: training and testing sets for both the target variable and the feature variables. To achieve this, first, the data was divided into ‘y’ and ‘X’. The ‘y’ dataset holds the values of the target variable, while the ‘X’ dataset contains all other variables, excluding the set of available target variables. Then, the training and testing splits were applied to ensure that the models were trained on historical, in-sample, data and tested on the out-of-sample future values. Given that the time window of the data was filtered in the silver layer, we defined the testing dataset to be the forecasting horizon of 12 months from January 2023 to December 2023. The training dataset ranged from January 2000 to December 2022. Thus, ‘y’ was split into ‘y_train’ and ‘y_test’, and similarly, ‘X’ was divided into ‘X_train’ and X_test’.

3.2. Data Standardization

Since the input variables are measured on different scales, for instance some are in millions of euros while others are in rates or indexes, and that could bias models towards variable with larger magnitudes, standardization was applied to ensure that machine

learning models could accurately compare those variables (Brownlee, 2020). To address this, we applied the StandardScaler object from the scikit-learn module sklearn.preprocessing, which scales each feature independently by subtracting the mean and dividing it by the standard deviation, shifting the distribution to have a mean of zero and a standard deviation of one.

The scaler was fitted only to the training data to avoid data leakage – data leakage occurs when information from the test set is shared with the training set, which can compromise the reliability of predictive models (Rosenblatt et al., 2024). The same mean and standard deviation from the fitted scaler were then applied to standardize both the training and testing datasets.

The scaling transformation applied is represented by:

$$z = \frac{(x - \mu)}{\sigma}$$

where μ is the mean and σ is the standard deviation of the training data.

3.3. Lagged Features

With the data standardized, we decided to generate two types of lagged features – lagged covariates features and lagged target features – to capture the temporal dynamics in the data, with the objective to improve the predictive power of forecast. Both types were created with lags from 1 to 12 months.

3.3.1. Lagged Covariates

The covariate variables, or 'X' variables, represent various macroeconomic indicators that may influence inflation, but their past values might also be relevant for predicting current inflation. Hence, for each variable in 'X', values from previous months were added as new features to the 'X' dataset.

3.3.2. Lagged Target

Like for covariates, the past values of inflation might as well be helpful to predict future inflation. However, these features were carefully incorporated to avoid data leakage from future inflation values into the training dataset, in other words, at this stage

the lagged target features were created based only on the data in ‘y_train’ and were added only to ‘X_train’ as new features.

Later in the workflow, during the recursive forecasting phase, these lagged features are updated to ‘X_test’, using the predicted values rather than future observations. In that phase, further details are provided.

3.3.3. Handling Missing Values

Lagging features introduce missing values depending on the degree of lag. For instance, the first value for every lag of 1 will be missing, and similarly, the first 12 values for every lag of 12 will be missing. To address this, the training datasets, ‘y_train’ and ‘X_train’, were aligned by removing the rows with these new missing values. Consequently, the time window for training was shifted to start 12 months later, equal to the number of lags, from January 2001 to December 2022.

3.4. Model Selection

For each combination of region and target variable, the workflow was designed to iterate over a list of models, which we chose to include two benchmark models and several machine learning models:

- Benchmark models: Random Walk, Autoregressive
- Shrinkage models: LASSO, Ridge, Elastic Net
- Ensemble methods: Bagging, Random Forest

The specifications and configurations for each model are discussed in Chapter 0. 4. Models.

3.5. Cross-Validation for Model Tuning

Before applying a machine learning model for out-of-sample predictions, we used a time series cross-validation method, known as walk-forward validation, to build and tune model’s hyper-parameters. This method splits the training data into multiple time-windows, with each window expanding its size iteratively by adding a new observation to the training set while making a prediction for the next in-sample time step.

To simulate a one-step out-of-sample predictions, we opted to split the training data into 2 one steps time-windows represented by:

- Fold 1: Train from January 2001 to October 2022, predict November 2022.
- Fold 2: Train from January 2001 to November 2022, predict December 2022.

The objective of using cross-validation in this study is to find the parameters that minimize the Root Mean Squared Error, across a grid of parameters that we specified.

3.6. Recursive Forecasting strategy

Once the datasets were prepared and the models were built and tuned, we applied a recursive forecasting strategy. This approach involved fitting a one-step model and using that same model iteratively to predict out-of-sample inflation values for the entire forecasting horizon – 12 months, from January 2023 to December 2023. In each iteration, the predicted value for the current month is used as an input for the next month's prediction. Since the predicted values are used in place of observations, the errors made in earlier predictions could propagate through the subsequent forecasts.

After the model obtained from cross-validation has been trained on the in-sample data, the recursive forecasting process can start, which can be broken down into multiple parts:

- The values of the features from 'X_test', that are used to predict January 2023, are stored in 'X_test_step'.
- The lagged target features are generated from 'y_train' and added to 'X_test_step'.
- The fitted model is used to predict the inflation value for January 2023, considering the data in the 'X_test_step'.
- The predicted value is added to 'y_train' to be used as an input to predict the value for the next month, February 2023.

This process is repeated iteratively for each month until December 2023 is predicted. Therefore, data leakage from future inflation values was avoided by generating iteratively the lagged target features.

3.7. Model Evaluation

The workflow ends by evaluating the performance for out-of-sample predictions on the testing dataset using three metrics: root mean squared error (RMSE), mean absolute error (MAE), and mean absolute percentage error (MAPE).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

4. Models

4.1. Benchmark

4.1.1. Random Walk (RW)

The underlying concept of a RW model is that, given the random and unpredictable nature of future movements, the current value provides the best estimate for the next time step. Therefore, the naïve forecasting approach, in which the forecast for every future period is set to the last observed value in the series, is based on the random walk model (Hyndman & Athanasopoulos, 2021).

Where for $h = 1, \dots, 12$, the forecasts can be written as:

$$\hat{y}_{T+h|T} = y_T$$

4.1.2. Autoregressive (AR)

In an AR model of order p, the target variable is forecasted by using its lagged values as predictors. This essentially turns the forecasting task into a multiple regression problem, where the number of features for the model corresponds to the order of p (Hyndman & Athanasopoulos, 2021). Thus, setting $p = 12$, an AR(12) model can be written as:

$$\hat{y}_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_{12} y_{t-12}$$

This was implemented using the AutoReg object from the statsmodels module statsmodels.tsa.ar_model (Statsmodels, 2024).

4.2. Shrinkage

Shrinkage models are a type of linear models that aims to reduce overfitting by applying a penalty to the features that have low predictive outcomes by shrinking their coefficients, in other words apply additional constraints to those coefficients.

Letting y_i be the target variable value and $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^T$ be the vector of p features for the *i*th case, and $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^T$ be a vector of coefficients, a popular method to estimate $\boldsymbol{\beta}$ is the *least squares*, that minimize the *residual sum of squares* (RSS) and can be written as:

$$RSS = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

In shrinkage models, the coefficients are estimated by minimizing a *penalized residuals sum of squared*:

$$PRSS = RSS + shrinkage\ penalty$$

The *shrinkage penalty* is calculated based on the *regularization parameter* or penalty parameter λ and on the penalty term. The λ controls the strength of the penalty, where the larger the value, the greater the amount of shrinkage, on the other hand if λ is set to 0, the *shrinkage penalty* has no effect so the estimations for the coefficients would be produced by *least squares*. The penalty term depends on the specific type of shrinkage model applied. Shrinkage models were implemented using Scikit-learn (n.d.-a) and García-Nieto et al. (2021).

4.2.1. Ridge regression (RR)

Hoerl and Kennard (1970) introduced the Ridge model, which approaches the task of estimating the coefficients by minimizing the loss function:

$$L^{RR}(\beta) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

As lambda increases, RR shrinks the coefficients estimates, which significantly lowers the variance of precipitations, at the expense of a slight increase in bias. This

shrinkage reduces overfitting and aims to improve prediction accuracy by penalizing large regression coefficients.

Even though RR shrinks all the coefficients toward zero, it does not perform feature selection because it will not set them exactly zero for any size of $\lambda \neq \infty$ and therefore it does not enhance the interpretability of the model (Scikit-learn, n.d.-b).

4.2.2. LASSO regression (LR)

Tibshirani (1996) introduced the LASSO (short for Least Absolute Shrinkage and Selection Order) model, that estimates the coefficients by minimizing:

$$L^{LR}(\beta) = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j|$$

Similarly to RR, LR also shrinks the coefficients estimates towards zero. However, unlike ridge, lasso's *shrinkage penalty*, can force some coefficient estimates to become exactly zero when lambda is set to be sufficiently large. This leads to variable selection, making the resulting models generally easier to interpret compared to those produced by RR (Scikit-learn, n.d.-c).

4.2.3. Elastic-net regression (ENR)

Zou and Hastie (2005) introduced the Elastic-net model, that aims to minimize:

$$L^{ENR}(\beta) = \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\left(\frac{1-\alpha}{2}\sum_{j=1}^{p}\beta_j^2 + \alpha\sum_{j=1}^{p}|\beta_j|\right)$$

The ENR model combines the *shrinkage penalties* from both Ridge and Lasso regressions. It is a convex combination of these two methods, controlled by a mixing parameter α, where $\alpha = 0$ corresponds to Ridge and $\alpha = 1$ corresponds to Lasso. This introduces two parameters to tune: λ and α. This model is particularly useful when the

number of features (*p*) is much bigger than the number of observations (*n*) (Scikit-learn, n.d.-d).

4.3. Ensemble

Ensemble methods are learning algorithms that combine the predictions of several base estimators built with a given learning algorithm to produce a stronger model, aiming to improve predictive accuracy over a single estimator (Scikit-learn, n.d.-e).

4.3.1. Bagging regression (BR)

Breiman (1996) introduced the Bagging (short for Bootstrap Aggregating) model, which, in this study's case, involves the following steps (Scikit-learn, n.d.-f):

- From the original dataset with all the features, draw multiple new datasets by sampling the available features (bootstrap sampling).
- For each bootstrapped dataset, train a base Linear regression model.
- Use the trained model to generate predictions.
- Repeat the process for all datasets.
- Aggregate the results and compute the final prediction by averaging the forecasts from the models.

4.3.2. Random Forest regression (RFR)

Breiman (2001) introduced the Random Forest model, which consists of training multiple regression trees based on bootstrapped samples of the data.

We followed similar steps as for the Bagging model. Yet, we allowed it to sample from observations, in addition to having the tree learning algorithm to select a random subset of the features. The steps can be represented as (Scikit-learn, n.d.-g):

- Create a set of datasets by sampling the available observations.
- For each bootstrapped dataset, train a tree where, at each candidate split, only a random subset of features is considered.

- Once all trees have been built, like for Bagging, their individual forecasts are aggregated to compute the final prediction by averaging them.

Since we are dealing with time series data, it could have been beneficial to implement *block bootstrapping*, that aims to keep some temporal dependencies while bootstrapping the original dataset. However, by implementing 12 lagged features for both the target variable and covariates, we intend to account, as much as possible, for time dependencies while benefitting from the randomness from Random Forest.

5. Results and Discussion

The models were compared according to the three metrics mentioned in Section 3.7. Model Evaluation: RMSE, MAE, and MAPE.

The results are organized in the next seven tables as follows. Each table corresponds to a region with one target variable and its values are ordered from the lowest to the highest RMSE.

Table 2: EU27 results for HICP_All_idx_NA

Model	***RMSE***	***MAE***	***MAPE***
lasso	0.5648	0.4303	0.0034
elastic_net	0.5957	0.4499	0.0035
bagging	0.7487	0.6843	0.0054
autoregressive	0.9775	0.7394	0.0058
ridge	2.4092	2.111	0.0166
random_walk	3.4399	3.1958	0.0252
random_forest	8.1125	7.707	0.0608

Table 3: EU27 results for HICP_All_mor_NA

Model	***RMSE***	***MAE***	***MAPE***
ridge	0.2797	0.268	1.50985E+14
elastic_net	0.3646	0.2845	1.44718E+14
lasso	0.3646	0.2845	1.44718E+14
random_forest	0.3888	0.3343	2.90857E+14
bagging	0.5278	0.4138	2.09626E+14
random_walk	0.5986	0.5167	1.5012E+14
autoregressive	0.7704	0.6387	5.38028E+14

Table 4: EA20 results for HICP_All_idx_NA

Model	***RMSE***	***MAE***	***MAPE***
lasso	0.3507	0.2536	0.0021
elastic_net	0.3609	0.2528	0.0021
autoregressive	0.7012	0.6202	0.005
bagging	1.1715	1.0319	0.0084
random_walk	2.9113	2.6767	0.0216
ridge	3.9341	3.4008	0.0275
random_forest	7.6716	7.2674	0.0588

Table 5: EA20 results for HICP_All_mor_NA

Model	***RMSE***	***MAE***	***MAPE***
ridge	0.3769	0.3406	9.28367E+12
elastic_net	0.4178	0.3383	6.38294E+13
lasso	0.4178	0.3383	6.38294E+13
random_forest	0.6382	0.5409	3.40272E+14
autoregressive	0.7315	0.6111	3.00274E+14
bagging	0.7558	0.6229	9.97594E+13
random_walk	0.7561	0.6667	1.5012E+14

Table 6: Germany results for HICP_All_idx_NA

Model	***RMSE***	***MAE***	***MAPE***
elastic_net	1.6964	1.5572	0.0123
lasso	1.7734	1.6545	0.0131
autoregressive	2.3657	2.1925	0.0174
random_walk	4.1069	3.8667	0.0306
bagging	5.2201	4.8527	0.0384
ridge	5.4593	5.0542	0.04
random_forest	8.1768	7.6587	0.0606

Table 7: Germany results for HICP_All_mor_NA

Model	***RMSE***	***MAE***	***MAPE***
random_forest	0.4313	0.3462	0.9548
lasso	0.5396	0.4572	1.5514
elastic_net	0.5419	0.4578	1.5444
autoregressive	0.8552	0.6944	2.4484
ridge	0.892	0.7525	1.8702
bagging	1.0486	0.9278	2.3513
random_walk	1.5927	1.5167	3.9004

Table 8: Germany results for HICP_All_anr_NA

Model	***RMSE***	***MAE***	***MAPE***
lasso	1.6214	1.3837	0.2296
bagging	1.6547	1.2619	0.3433
random_forest	1.8216	1.1708	0.3608
ridge	1.8322	1.4529	0.3853
autoregressive	2.4477	2.142	0.4947
random_walk	4.1297	3.4917	0.8846
elastic_net	4.2378	3.4825	0.8844

In this study the focus of the analysis is to compare models, target variables, and metric and therefore exclude comparison between regions. Following this train of thoughts, it's possible to retrieve a few insights from Tables 2 to 8.

When the inflation indicator is measured by index, through the variable HICP_All_idx_NA, the model Random Forest performs considerably worst, across the three metrics, compared to the rest, while the models LASSO and Elastic-net perform the best. On the other hand, when it's measured by monthly rate (HICP_All_mor_NA) the best model is Ridge followed by Elastic-net for the aggregate regions EU27 and EA20, for Germany is the model Random Forest. Still on this variable, the worst candidates were

the two benchmarks as well as the Bagging. Finally, using the annual rate measure (HICP_All_anr_NA), that was only observed for Germany, the best models were LASSO and Bagging, and the worst were Random Walk and Elastic-net. Table 9 aims to help visualize the number of times a model was the best and the worst in terms of performance. Note that it's an illustration and it should not be used alone to decide on which model is the best and worst overall, for instance, the model Ridge was the best 4 times in total across all metrics and was never the worst, however looking again to tables 2 to 8, when it was not on top, it was closer to the bottom of the list.

Table 9: Summary statistics for metrics

Model	*#Min. RMSE*	*#Min. MAE*	*#Min. MAPE*	*Count Min*	*#Max. RMSE*	*#Max. MAE*	*#Max. MAPE*	*Count Max*
RW				0	2	3	2	7
AR				0	1	1	1	3
RR	2	1	1	4				0
LR	3	1	2	6				0
ENR	1	3	3	7	1			1
BR				0				0
RFR	1	2	1	4	3	3	4	10

As shown on the Table 10 (below), on average, the models were able to predict with better accuracy using the measure monthly rate (HICP_All_mor_NA), resulting in a lower RMSE and MAE, however the values for MAPE are way off, excluding for Germany.

Table 10: Average result metrics

Region	Target variable	Avg. RMSE	Avg. MAE	Avg. MAPE
EU27	HICP_All_idx_NA	2.4069	2.1882	0.0172
EU27	HICP_All_mor_NA	0.4706	0.3915	2.3272E+14
EA20	HICP_All_idx_NA	2.4430	2.2148	0.0179
EA20	HICP_All_mor_NA	0.5849	0.4941	1.46767E+14

Germany	HICP_All_idx_NA	4.1141	3.8337	0.0303
Germany	HICP_All_mor_NA	0.8430	0.7361	2.0887
Germany	HICP_All_anr_NA	2.5350	2.0551	0.5118

6. Conclusion

This dissertation's main objective was to implement machine learning methods for monthly inflation forecasting in a European context with an emphasis on EU27, EA20, and Germany. To achieve this, we built the EUED database, created the workflow in Python and applied well established models.

The analysis demonstrated that machine learning models, such as LASSO and Elastic-net, were effective in forecasting inflation. Furthermore, it showed that using the inflation indicator measured in monthly rates leads to more consistent results across a variety of models. Finally, aside from Random Forest, the machine learning models outperform the benchmark models in terms of predictions accuracy, highlighting the benefits of these techniques for macroeconomic forecasts.

Apart from the methodological contributions addressed in this work, the database built stands out as one of the most noteworthy and lasting achievements of this research. It represents a valuable resource for future research in the field of macroeconomics as it consists in an aggregation of multiple datasets from macroeconomic Eurostat sources. Most efforts were invested in building this database and ensuring data quality through data engineering techniques, like the approach of Medallion Architecture, to make it ready to be the backbone used for forecasting models. On the topic of this approach, it was applied to ensure that future researchers can implement data transformations to fit their needs.

While this study provides both the database and the Python workflow, it is important to note certain limitations, such as the fact that the database is not updated in real-time and that the block bootstrap was not included in the workflow, which could have improved the accuracy of the Random Forest model. A point that went against our wishes was in cross-validation. One of our goals was to implement cross-validation with 12 time-windows, to simulate the 12 one-steps out-of-sample predictions with a recursive strategy, using an extensive grid of parameters to tune the model, but close to the deadline, we discovered an error in the Python code and we had to rerun the entire process to get the corrected results, and because training the models was taking a lot of time, we had to adapt the cross-validation to fit the schedule.

Future studies could begin by addressing the limitations listed above. Incorporating real-time data updates as soon as they become available in Eurostat would be a significant improvement since, as time passes by, the database will become outdated for analysis that required rapid decisions. Comparing a 12 forecast horizon using a recursive strategy, like we did, against a direct strategy would be an interesting starting point for future studies.

The use of machine learning techniques, workflow and data gathered lays a solid foundation for future research, with the hope that this work will contribute to the continuous development of more robust and reliable economic forecasting models.

References

Bentéjac, C., Csörgő, A., & Martínez-Muñoz, G. (2021). A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, *54*, 1937-1967. https://doi.org/10.1007/s10462-020-09896-5

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, 24, 123–140.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45, 5–32.

Brownlee, J. (2020). *How to Use StandardScaler and MinMaxScaler Transforms in Python*. Machine Learning Mastery. https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/

Dantas, T. M., & Oliveira, F. L. C. (2018). Improving time series forecasting: An approach combining bootstrap aggregation, clusters and exponential smoothing. *International Journal of Forecasting*, 34(4), 748-761. https://doi.org/10.1016/j.ijforecast.2018.05.006

Data Bricks. (n.d.). *What is a medallion architecture?*. https://www.databricks.com/glossary/medallion-architecture

European Central Bank. (n.d.). *What is inflation?*. https://www.ecb.europa.eu/ecb-and-you/explainers/tell-me-more/html/what_is_inflation.en.html

European Union/Eurostat. (2018). Harmonised Index of Consumer Prices (HICP). *Methodological Manual*, 360.

Eurostat. (n.d.). *Euro Indicators Database*. https://ec.europa.eu/eurostat/web/euro-indicators/database

Gafurdjan , Z. (2024). INFLATION AND ITS EFFECTS ON CONSUMER BEHAVIOR AND ECONOMIC POLICIES. *QO‘QON UNIVERSITETI XABARNOMASI*, *10*(10), 3–6. https://doi.org/10.54613/ku.v10i10.895

García-Nieto, P. J., Garcia-Gonzalo, E., & Paredes-Sánchez, J. P. (2021). Prediction of the critical temperature of a superconductor by using the WOA/MARS, Ridge, Lasso and Elastic-net machine learning techniques. *Neural Computing and Applications*, *33*, 17131-17145. https://doi.org/10.1007/s00521-021-06304-z

Hoerl, A. E., & Kennard, R. W. (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1), 55–67. https://doi.org/10.1080/00401706.1970.10488634

Hyndman, R.J., & Athanasopoulos, G. (2021) *Forecasting: principles and practice*, 3rd edition, OTexts.

Mandeya, S. M. T., & Ho, S. Y. (2021). Inflation, inflation uncertainty and the economic growth nexus: An impact study of South Africa. *MethodsX*, *8*. https://doi.org/10.1016/j.mex.2021.101501

McKinsey & Company. (2024). *What is inflation?*. https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-inflation

Medeiros, M. C., Vasconcelos, G. F., Veiga, Á., & Zilberman, E. (2019). Forecasting inflation in a data-rich environment: the benefits of machine learning methods. *Journal of Business & Economic Statistics*, *39*(1), 98-119. https://doi.org/10.1080/07350015.2019.1637745

Rosenblatt, M., Tejavibulya, L., Jiang, R., Noble, S., & Scheinost, D. (2024). Data leakage inflates prediction performance in connectome-based machine learning models. *Nature Communications*, *15*(1). https://doi.org/10.1038/s41467-024-46150-w

Scikit-learn. (n.d.-a). *Linear Models*. https://scikit-learn.org/stable/modules/linear_model.html

Scikit-learn. (n.d.-b). *Ridge*. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html

Scikit-learn. (n.d.-c). *Lasso*. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html

Scikit-learn. (n.d.-d). *ElasticNet*. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html

Scikit-learn. (n.d.-e). *Ensembles: Gradient boosting, random forests, bagging, voting, stacking*. https://scikit-learn.org/stable/modules/ensemble.html

Scikit-learn. (n.d.-f). *BaggingRegressor*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html#sklearn.ensemble.BaggingRegressor

Scikit-learn. (n.d.-g). *RandomForestRegressor*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html#sklearn.ensemble.RandomForestRegressor

Statsmodel, (2024). *AutoReg*. https://www.statsmodels.org/stable/generated/statsmodels.tsa.ar_model.AutoReg.html#statsmodels.tsa.ar_model.AutoReg

Tibshirani, R. (1996). Regression Shrinkage and Selection via the LASSO. *Journal of the Royal Statistical Society*, 58, 267–288.

Zou, H., & Hastie, T. (2005). Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society*, Series B, 67, 301–320. https://doi.org/10.1111/j.1467-9868.2005.00503.x