Simple Moldtsova and Papell Strategy

Hyperparameters Tunning

ECON 409 Project Proposal

Idalee Vargas, Lora Yovcheva, Mauricio Vargas, Santiago Naranjo

As a reminder, the proposed model consists of:

$$\Delta s_{t+1} = \beta_0 + \beta_1(\tilde{\pi}_{t-1} - \pi_{t-1}) + \beta_2(\gamma_1^2 \tilde{y}_{t-2} - \alpha_1 y_{t-1}) + \epsilon_t$$

Where:

- Δs_{t+1}: Difference in the logarithms of the exchange rate at $t+1$.
- π_{t-1}: Inflation in the United States at $t-1$.
- $\tilde{\pi}_{t-1}$: Inflation in the United Kingdom at $t-1$.
- y_{t-1}: Output gap in the United States at $t-1$.
- $\tilde{y}_{t-2}$: Output gap in the United Kingdom at $t-2$.

Ridge Regression

We define the loss function as:

$$L(\widehat{\Delta s}_{t+1}|\beta_0, \beta_1, \beta_2) = MSE(\widehat{\Delta s}_{t+1}) + \lambda \sum_{i=0}^{2} \beta_i^2$$

where:

$$\beta^* = (\beta_0^*, \beta_1^*, \beta_2^*)' = \text{arg min}_{\beta \in \mathbb{R}^3} L(\widehat{\Delta s}_{t+1}|\beta)$$

Tuning Parameters

1. λ: Regularization term in ***Ridge*** regression.
2. h: Rolling window size for model estimation and the output gaps, to avoid including structural breaks that could affect forecasts.

Hyperparameter Selection Procedure

Detailed Algorithm

A value for λ and a time window size h are selected, where λ is the regularization parameter for Ridge regression and h represents the size of the time window of the sample to estimate a model.

Considering that T observations are available, the process proceeds as follows using the first h data of the sample:

1. The Akaike Information Criterion (AIC) is calculated for the selected sample. A bootstrap method is used on the residuals to generate an extended sample of these up to a total of 10,000 residuals, from which the AIC is calculated.
2. An out-of-sample forecast is made for the period $h + 1$, and the error of this forecast is calculated.

This procedure is repeated by shifting the time window forward by one period in each iteration. For example, the second iteration covers from $t = 2$ to $t = h + 1$, and so on until the last iteration includes data from $t = T - h - 1$ to $t = T - 1$.

This process generates two vectors: one with the AIC values calculated in each time window and another with the squared errors one period ahead out-of-sample. The values of both vectors are averaged to obtain an average AIC and a Mean Squared Forecast Error (MSFE) for the specific configuration of λ and h chosen.

Algorithm Pseudocode
Inputs:

- T: total number of available observations.
- lambda: regularization parameter for Ridge regression.
- h: time window size for the sample.

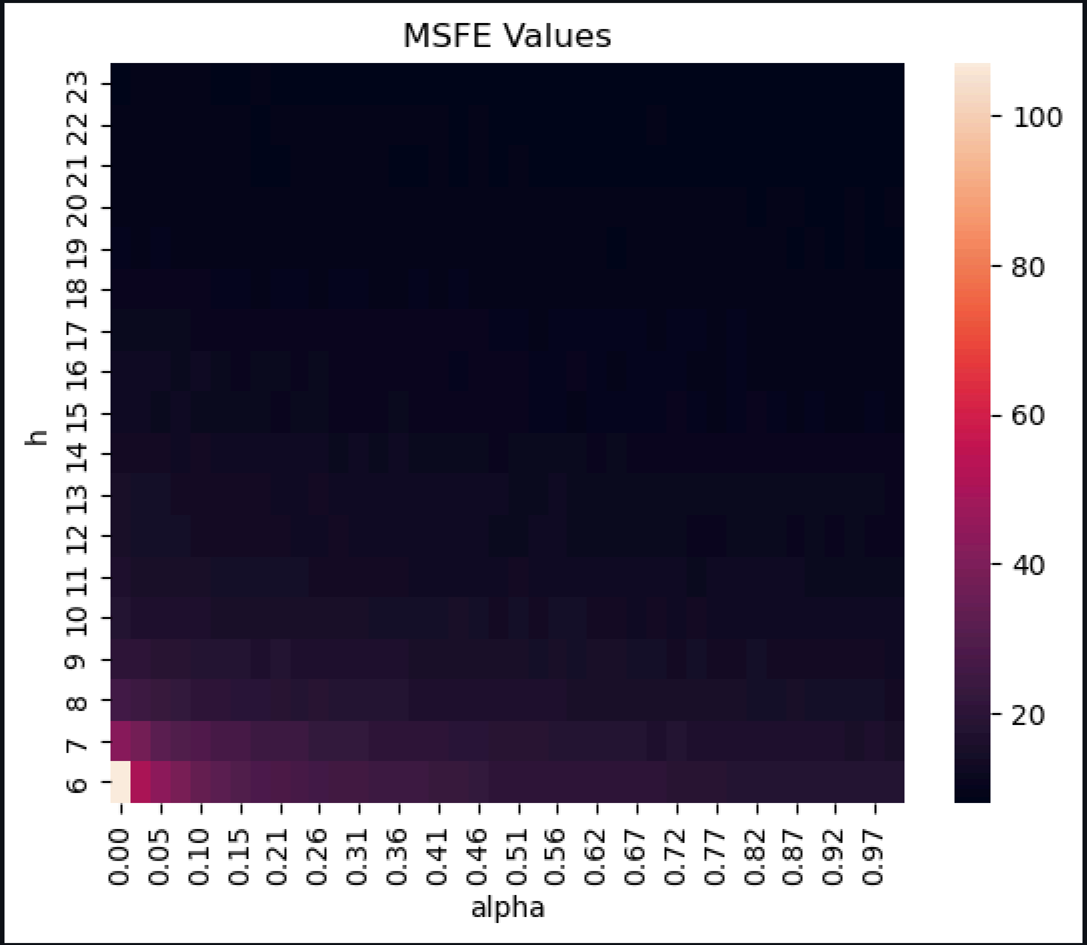
Process:

```
1. Initialize empty AIC_vector and MSFE_vector arrays.
2. For each time window from t=1 to t=T-h do:
      i. Select the data sample corresponding to the window [t, t+h-1].
     ii. Estimate the Ridge regression model with the current window data:
         Model: $\Delta s_{t+1} = \beta_0 + \beta_1(\tilde{\pi}_{t-1} - \pi_{t-1}) + \beta_2(\gamma_1^2 \tilde{y}_{t-2} - \alpha_1 y_{t-1}) + \varepsilon_t$
    iii. Calculate the AIC for the estimated model.
     iv. Apply bootstrap to the model residuals to generate an extended sample of 10000 residuals.
      v. Calculate the AIC using the extended sample of residuals.
     vi. Perform an out-of-sample forecast for the period t+h and calculate the forecast error.
    vii. Add the AIC calculated in step 2.5 to the AIC_vector.
   viii. Add the forecast mean square error to the MSFE_vector.
3. Calculate the average AIC from the values in AIC_vector.
4. Calculate the average MSFE from the values in MSFE_vector.
```

Outputs:

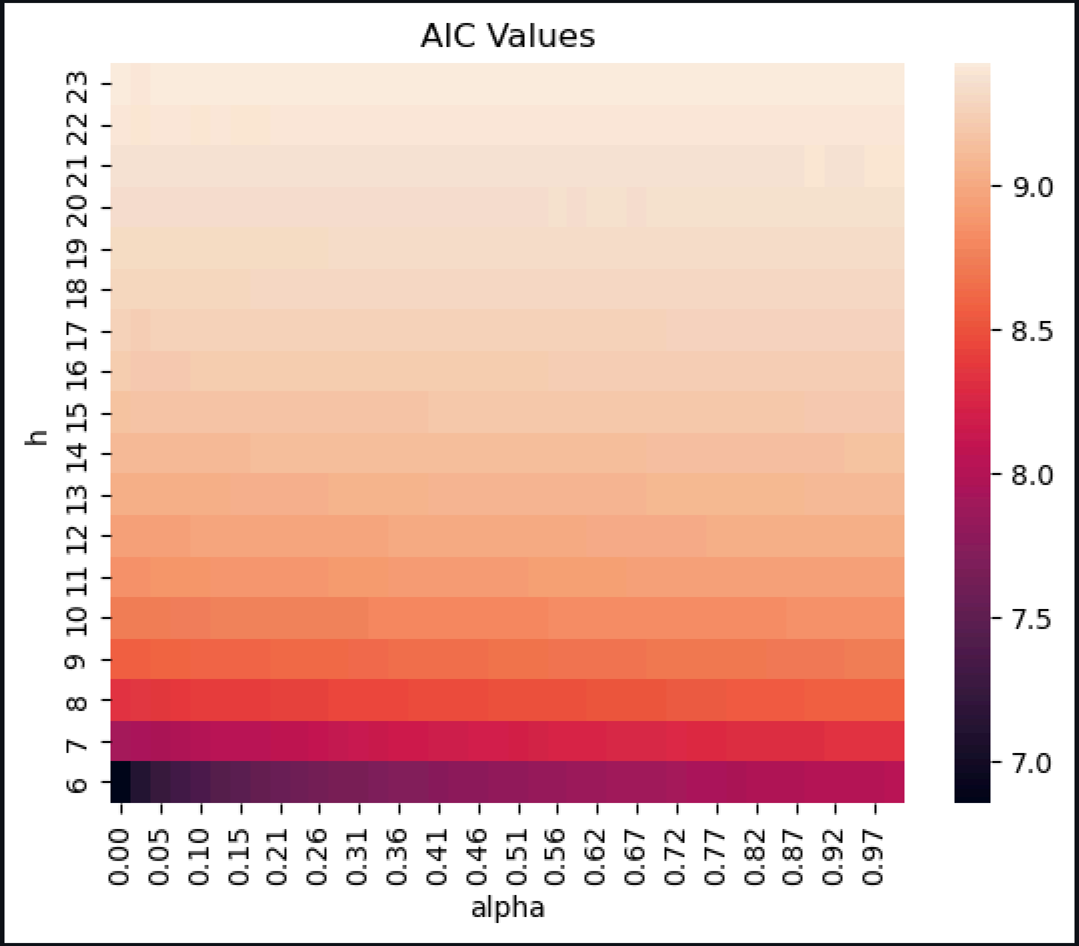
- Average AIC: average value of the AIC calculated across all time windows.
- Average MSFE: average value of the out-of-sample mean square forecast error.

Results: GridSearch With MSFE



- It can be noted that with small time windows, the prediction error increases. This is because it's not possible to collect enough observations to properly estimate the linear model, as well as to estimate the output gaps.
- Additionally, it's observed that from a sample of 12 months and a ridge parameter close to 0.2, the forecasts do not show a significant improvement.

Results: GridSearch With AIC



Procedure for Evaluating the Simple Papell Model

Objective: Adjust the hyperparameters of the proposed simple Papell model in order to maximize the model's accuracy.

- Evaluation Metric: AIC
- Evaluation Procedure: Cross-validation on a moving time window of size h.

For a set of parameters (h, λ):

- Information from $t - h$ to t. Calculated with a simple *Nowcasting* procedure.
  - Nowcasting $\mathbb{E}_{t-1}[\pi_t]$.
  - Nowcasting $\mathbb{E}_{t-1}[\tilde{\pi}_t]$.
  - Nowcasting $\mathbb{E}_{t-1}[y_t]$.
  - Nowcasting $\mathbb{E}_{t-2}[\tilde{y}_t]$.

In [ ]:
```python
import matplotlib.pyplot as plt
import pandas as pd
import toolz as tz
import numpy as np
import src

import itertools as itt

import matplotlib.pyplot as plt
import seaborn as sns

import multiprocessing
```

In [ ]:
```python
data = pd.read_csv('data/processed_data.csv', index_col=0, parse_dates=True)
```

In [ ]:
```python
# Defining the variables to be used in the nowcasting procedure and the lags
# to update.
variables = {
    'pi': {
        'lag': 1,
        'nowcast': src.pi_nowcast_
    },
    'pi_star': {
        'lag': 1,
        'nowcast': src.pi_nowcast_
    },
    'Y': {
        'lag': 1,
        'nowcast': src.gap_nowcast_
    },
    'Y_star': {
        'lag': 2,
        'nowcast': src.gap_nowcast_
    }
}
```

In [ ]:
```python
# Defining the rolling AIC function to be used in the grid search procedure.
# Bootstrap is set to True.
def rolling_aic(X: pd.DataFrame, y: pd.Series, h: int, alpha: float) -> float:
    data_nowcasted = src.nowcasting_(X, y, lags=variables)
    data_papell = tz.pipe(
        data_nowcasted,
        lambda x: x.assign(diff_inf = x.iloc[:, 2] - x.iloc[:, 1]),
        lambda x: x.assign(diff_y = x.iloc[:, 4] - x.iloc[:, 3]),
        lambda x: x.iloc[:, [0, 5, 6]]
    )
    return src.rolling_aic_model_(
        data = data_nowcasted,
        alpha = alpha,
        h = h,
        agg=np.mean,
        bootstrap=True
    )
```

In [ ]:
```python
#Defining the rolling MSFE function to be used in the grid search procedure.
#Bootstrap is set to True.
def rolling_msfe(X: pd.DataFrame, y: pd.Series, h: int, alpha: float) -> float:
    data_nowcasted = src.nowcasting_(X, y, lags=variables)
    data_papell = tz.pipe(
        data_nowcasted,
        lambda x: x.assign(diff_inf = x.iloc[:, 2] - x.iloc[:, 1]),
        lambda x: x.assign(diff_y = x.iloc[:, 4] - x.iloc[:, 3]),
        lambda x: x.iloc[:, [0, 5, 6]]
    )
    return src.rolling_msfe_model_(
        data = data_nowcasted,
        alpha = alpha,
        h = h,
        bootstrap=True,
        agg=lambda x: np.mean(x**2)
    )
```

```
In [ ]: # Grid search for the hyperparameters h and alpha.
        grid_ = {
            'h':list(range(6, 24)),
            'alpha': np.linspace(0, 1, 40)
        }
        grid_ = list(itt.product(grid_['h'], grid_['alpha']))
```

```
In [ ]: # Parallelizing the grid search.
        def map_rolling_aic(args):
            x, h, alpha = args
            return rolling_aic(data.drop(columns='S'), data['S'], h, alpha)
```

```
In [ ]: def map_rolling_msfe(args):
            x, h, alpha = args
            return rolling_msfe(data.drop(columns='S'), data['S'], h, alpha)
```

```
In [ ]: with multiprocessing.Pool() as p:
            msfe_values = p.map(
                map_rolling_msfe,
                [(data, x[0], x[1]) for x in grid_]
            )
```

```
In [ ]: with multiprocessing.Pool() as p:
            aic_values = p.map(
                map_rolling_aic,
                [(data, x[0], x[1]) for x in grid_]
            )
```

```
In [ ]: # Dataframe with MSFE values and hyperparameters.
        msfe_values = pd.DataFrame(
            {
                'h': [x[0] for x in grid_],
                'alpha': [x[1] for x in grid_],
                'msfe': msfe_values
            }
        )
```

```
In [ ]: # Dataframe with AIC values and hyperparameters.
        aic_values = pd.DataFrame(
            {
                'h': [x[0] for x in grid_],
                'alpha': [x[1] for x in grid_],
                'aic': aic_values
            }
        )
```

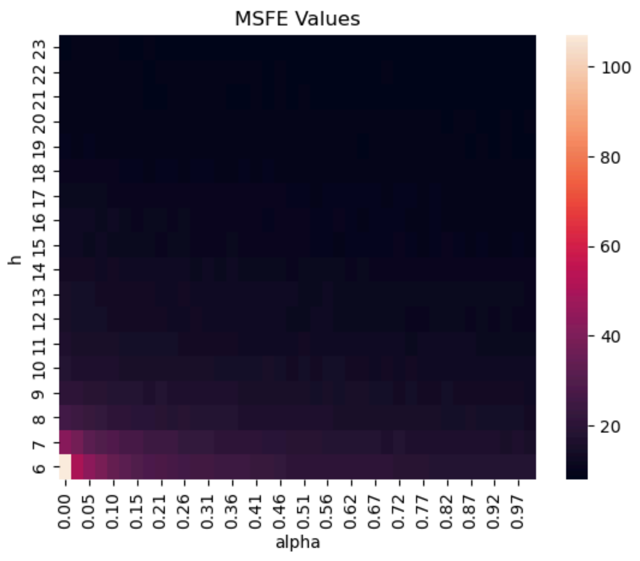
```
In [ ]: aic_values.to_csv('data/aic_values.csv', index=False)
```

```
In [ ]: # Heatmap for the MSFE values.
        msfe_values_heatmap = msfe_values.pivot(index='h', columns='alpha', values='msfe')
        msfe_values_heatmap = msfe_values_heatmap.sort_index(ascending=False)

        fig, ax = plt.subplots()
        sns.heatmap(msfe_values_heatmap, ax=ax)
        x_tick_labels = [item.get_text() for item in ax.get_xticklabels()]
        formatted_labels = [f"{float(label):.2f}" for label in x_tick_labels]
        ax.set_xticklabels(formatted_labels, rotation=90)  # Aplicar las etiquetas formateadas al eje X

        ax.set_title('MSFE Values')

        plt.show()
```

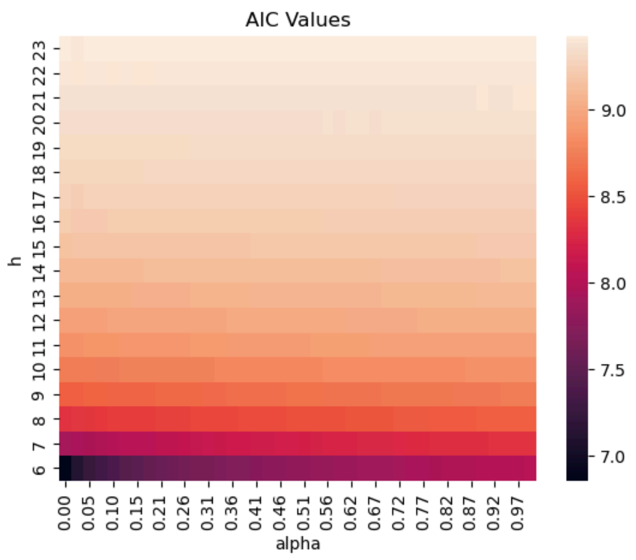


In [ ]:
```python
# Heatmap for the AIC values.
aic_values_heatmap = aic_values.pivot(index='h', columns='alpha', values='aic')
aic_values_heatmap = aic_values_heatmap.sort_index(ascending=False)

fig, ax = plt.subplots()
sns.heatmap(aic_values_heatmap, ax=ax)
x_tick_labels = [item.get_text() for item in ax.get_xticklabels()]
formatted_labels = [f"{float(label):.2f}" for label in x_tick_labels]
ax.set_xticklabels(formatted_labels, rotation=90)  # Aplicar las etiquetas formateadas al eje X

ax.set_title('AIC Values')

plt.show()
```



In [ ]:
```

```

- The AIC criterion can be used as an estimator for out-of-sample error. According to our results, the smallest window without penalty has the best balance between bias and variance.
- This result is counterintuitive and arises from the resampling method of errors, indicating that further research is necessary.