# Proj_PNAS

November 18, 2023

## 1 Sentiment of ECB speeches and the perceived effect on Inflation.

In the followin Notebook we are going to lay the foundation for our research regarding the impact that the speeches of the ECB might play in the market and the sentiment regard inflation. #### Import the libraries we need:

[ ]:
```
import pandas as pd
from tqdm.notebook import tqdm
import os
import nltk
from nltk.tokenize import word_tokenize
```

[ ]:
```
nltk.download([
     "names",
     "stopwords",
     "state_union",
     "twitter_samples",
     "movie_reviews",
     "averaged_perceptron_tagger",
     "vader_lexicon",
     "punkt",
])
```

```
[nltk_data] Downloading package names to
[nltk_data]     C:\Users\Utente\AppData\Roaming\nltk_data…
[nltk_data]   Package names is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Utente\AppData\Roaming\nltk_data…
[nltk_data]   Unzipping corpora\stopwords.zip.
[nltk_data] Downloading package state_union to
[nltk_data]     C:\Users\Utente\AppData\Roaming\nltk_data…
[nltk_data]   Package state_union is already up-to-date!
[nltk_data] Downloading package twitter_samples to
[nltk_data]     C:\Users\Utente\AppData\Roaming\nltk_data…
[nltk_data]   Package twitter_samples is already up-to-date!
[nltk_data] Downloading package movie_reviews to
[nltk_data]     C:\Users\Utente\AppData\Roaming\nltk_data…
[nltk_data]   Package movie_reviews is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
```

```
[nltk_data]     C:\Users\Utente\AppData\Roaming\nltk_data…
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\Utente\AppData\Roaming\nltk_data…
[nltk_data]   Package vader_lexicon is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Utente\AppData\Roaming\nltk_data…
[nltk_data]   Unzipping tokenizers\punkt.zip.
```

[ ]:
```
True
```

**Import the Data we need:** in the following section of our notebook we will introduce the Data scraped from the ECB up until 2023

[ ]:
```
main_ds = pd.read_csv("Inputs//all_ECB_speeches.csv", sep="|") #if the file has␣
↪to be changed, be careful with the sep option.
main_ds.head()
```

[ ]:
```
         date           speakers  \
0  2023-10-30    Luis de Guindos
1  2023-10-25  Christine Lagarde
2  2023-10-17    Luis de Guindos
3  2023-10-14  Christine Lagarde
4  2023-10-04    Luis de Guindos

                                               title  \
0  The euro area economy and our monetary policy …
1  Remarks delivered at the Bank of Greece on the…
2  Macroprudential policy and research: learning …
3                                     IMFC Statement
4  The inflation outlook and monetary policy in t…

                                            subtitle  \
0  Remarks by Luis de Guindos, Vice-President of …
1  Speech by Christine Lagarde, President of the …
2  Dinner speech by Luis de Guindos, Vice-Preside…
3  Statement by Christine Lagarde, President of t…
4  Keynote speech by Luis de Guindos, Vice-Presid…

                                            contents
0    SPEECH  The euro area economy and our moneta…
1    SPEECH   Remarks delivered at the Bank of G…
2    SPEECH   Macroprudential policy and researc…
3    SPEECH  IMFC Statement   Statement by Christ…
4    SPEECH  The inflation outlook and monetary p…
```

```
[ ]: # we need to combine the speeches uniting title, subtitle and contents

main_ds["speech_merged"] = main_ds["title"] + "\n" + main_ds["subtitle"] + "\n"␣
↪+ main_ds["contents"]
main_ds = main_ds.drop(columns=["title", "subtitle", "contents"])
```

```
[ ]: import nltk
from nltk.tokenize import word_tokenize
from tqdm import tqdm

# Ensure that the necessary NLTK resources have been downloaded
nltk.download('punkt')
nltk.download('stopwords')

def preprocessing_df(ds_input):
    # Load stopwords once, outside the loop
    stopwords = nltk.corpus.stopwords.words("english")

    # Define a preprocessing routine
    def preprocess_text(speech):
        tokens = word_tokenize(speech)
        tokens = [w.lower() for w in tokens if w.isalpha() or w.isdigit() and w.
↪lower() not in stopwords]
        return ' '.join(tokens)

    # Use tqdm with apply to display a progress bar
    tqdm.pandas(desc="Processing rows")
    ds_input['speech_merged'] = ds_input['speech_merged'].astype(str).
↪progress_apply(preprocess_text)

    return ds_input

# Apply this function to your DataFrame
# ds_prep = preprocessing_df(your_dataframe)
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Utente\AppData\Roaming\nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Utente\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

```
[ ]: ds_prep = preprocessing_df(main_ds)
ds_prep.to_excel("Outputs//Intermediate//prep_new.xlsx", index=False)
```

```
Processing rows:   0%|          | 0/2736 [00:00<?, ?it/s]Processing rows:
100%|      | 2736/2736 [00:55<00:00, 49.65it/s]
```

[ ]:
```
import pandas as pd
from tqdm.notebook import tqdm
import os
import nltk
from nltk.tokenize import word_tokenize
import numpy as np
```

**Analysis of the Sentiment** Here below, after having done the pre-processing of our file, we use the VADER algorithm to establish the sentiment score of each speech

[ ]:
```
ds_prep = pd.read_excel("Outputs//Intermediate//prep_new.xlsx")
```

[ ]:
```
from nltk.sentiment import SentimentIntensityAnalyzer
sia = SentimentIntensityAnalyzer()
scores = []
for sentence in tqdm(ds_prep["speech_merged"]):
    score = sia.polarity_scores(str(sentence))
    scores.append(score)
neg_scores = [score['neg'] for score in scores]
neu_scores = [score['neu'] for score in scores]
pos_scores = [score['pos'] for score in scores]
compound_scores = [score['compound'] for score in scores]

ds_prep['neg_score'] = neg_scores
ds_prep['neu_score'] = neu_scores
ds_prep['pos_score'] = pos_scores
ds_prep['compound_score'] = compound_scores
```

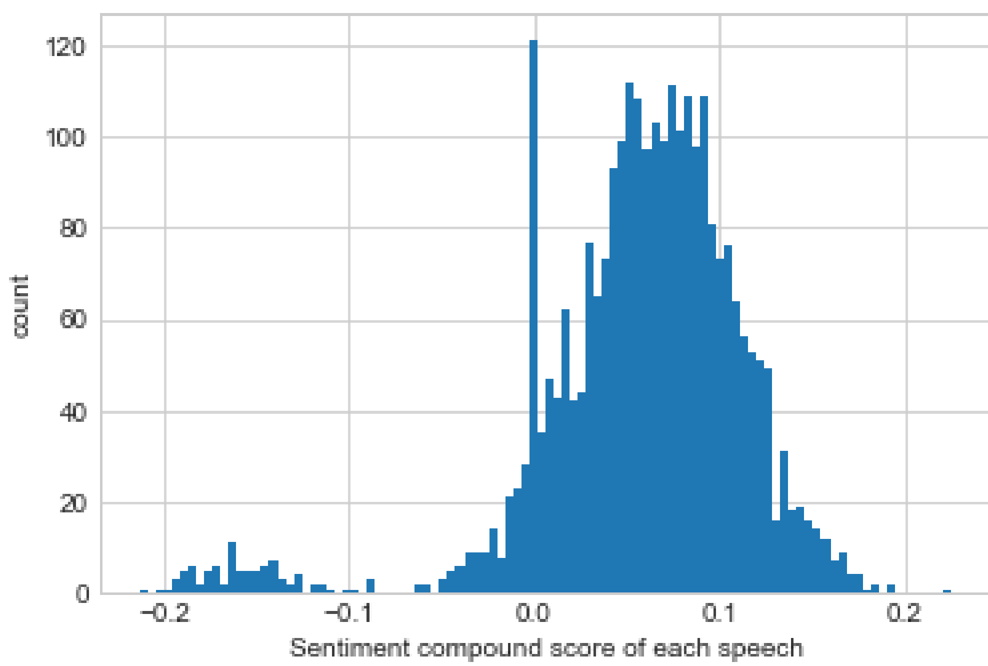
```
  0%|          | 0/2736 [00:00<?, ?it/s]
```

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
ds_prep['tanh_normalized_compound'] = np.
↪tanh(ds_prep['pos_score']-ds_prep["neg_score"])

data = np.tanh(ds_prep['tanh_normalized_compound'])
fig = plt.figure() ; ax = plt.axes( xlabel= "Sentiment compound score of each␣
↪speech", ylabel= "count")
ax.hist(data, bins=100)
```

[ ]:
```
(array([  1.,   0.,   1.,   1.,   3.,   5.,   6.,   2.,   5.,   6.,   2.,
         11.,   5.,   5.,   5.,   6.,   7.,   3.,   2.,   4.,   0.,   2.,
          2.,   1.,   0.,   1.,   1.,   0.,   3.,   0.,   0.,   0.,   0.,
          0.,   2.,   2.,   0.,   3.,   5.,   6.,   9.,   9.,   9.,  14.,
          8.,  21.,  23.,  28., 121.,  35.,  47.,  43.,  62.,  42.,  44.,
         77.,  65.,  73.,  93.,  99., 112., 108.,  97., 103.,  99., 111.,
        101., 109.,  98., 109.,  81.,  73.,  76.,  64.,  56.,  53.,  51.,
         49.,  16.,  31.,  18.,  19.,  16.,  14.,  12.,   7.,   9.,   4.,
```

```
         4.,    1.,    2.,    0.,    2.,    0.,    0.,    0.,    0.,    0.,    0.,
         1.]),
array([-0.21319687, -0.20881531, -0.20443375, -0.2000522 , -0.19567064,
       -0.19128908, -0.18690753, -0.18252597, -0.17814441, -0.17376286,
       -0.1693813 , -0.16499974, -0.16061819, -0.15623663, -0.15185507,
       -0.14747352, -0.14309196, -0.1387104 , -0.13432885, -0.12994729,
       -0.12556573, -0.12118418, -0.11680262, -0.11242106, -0.10803951,
       -0.10365795, -0.09927639, -0.09489484, -0.09051328, -0.08613172,
       -0.08175017, -0.07736861, -0.07298705, -0.0686055 , -0.06422394,
       -0.05984238, -0.05546083, -0.05107927, -0.04669771, -0.04231616,
       -0.0379346 , -0.03355304, -0.02917149, -0.02478993, -0.02040837,
       -0.01602682, -0.01164526, -0.0072637 , -0.00288215,  0.00149941,
        0.00588097,  0.01026252,  0.01464408,  0.01902564,  0.02340719,
        0.02778875,  0.03217031,  0.03655186,  0.04093342,  0.04531498,
        0.04969653,  0.05407809,  0.05845965,  0.0628412 ,  0.06722276,
        0.07160432,  0.07598587,  0.08036743,  0.08474899,  0.08913054,
        0.0935121 ,  0.09789366,  0.10227521,  0.10665677,  0.11103833,
        0.11541988,  0.11980144,  0.124183  ,  0.12856455,  0.13294611,
        0.13732767,  0.14170922,  0.14609078,  0.15047234,  0.15485389,
        0.15923545,  0.16361701,  0.16799856,  0.17238012,  0.17676168,
        0.18114323,  0.18552479,  0.18990634,  0.1942879 ,  0.19866946,
        0.20305101,  0.20743257,  0.21181413,  0.21619568,  0.22057724,
        0.2249588 ]),
<BarContainer object of 100 artists>)
```



[ ]:
```
ds_prep.to_excel("Outputs//Intermediate//sentiment.xlsx")
```

### 1.1 ECONOMETRIC ANALYSIS

[ ]:
```
import pandas as pd
from tqdm.notebook import tqdm
import os
import nltk
from nltk.tokenize import word_tokenize
import numpy as np
```

[ ]:
```
ds_prep = pd.read_excel("Outputs//Intermediate//sentiment.xlsx")
```

We now plot and do some descriptive analysis:

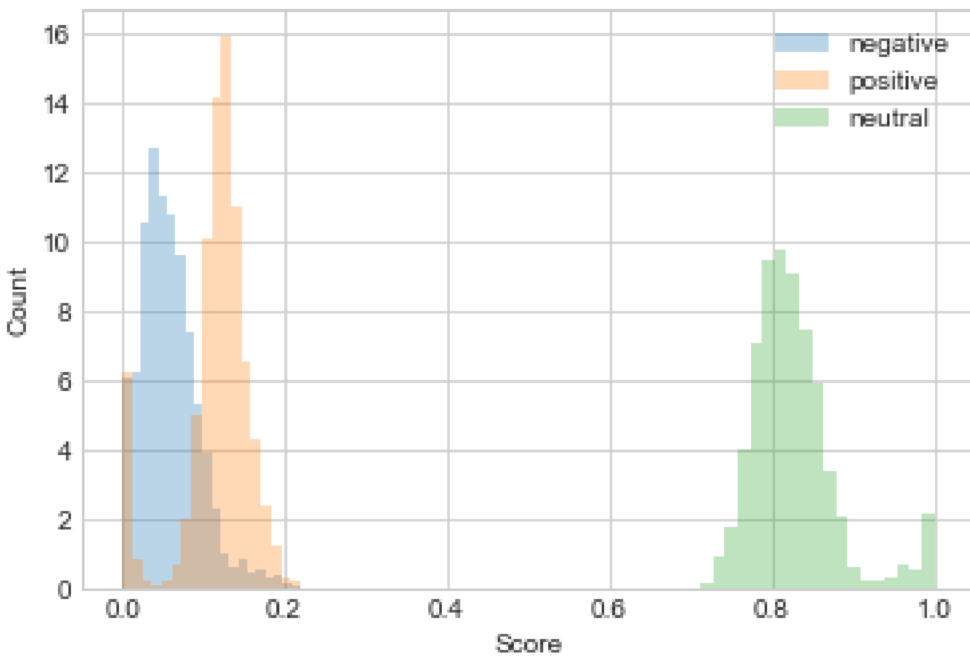
[ ]:
```
import matplotlib.pyplot as plt

plt.style.use("seaborn-whitegrid")
fig = plt.figure()
ax = plt.axes(xlabel="Score", ylabel="Count")

x1 = ds_prep["neg_score"]
x2 = ds_prep["pos_score"]
x3 = ds_prep["neu_score"]

kwargs = dict(histtype="stepfilled", alpha=0.3, density=True, bins=20)
ax.hist(x1, label="negative", **kwargs)
ax.hist(x2,label="positive", **kwargs)
ax.hist(x3,label="neutral", **kwargs)

plt.legend()
plt.show()
```




We create now the Timeseries for our compound Score

[ ]:
```
import numpy as np

ds_prep['date'] = pd.to_datetime(ds_prep['date'])

ds_prep['year_month'] = ds_prep['date'].dt.to_period('M')

# Apply tanh function to the 'compound_score' column
# It seems the 'tanh_normalized_compound' column might already be the tanh of␣
↪compound_score,
# but we will compute it again to be sure

# Group by year and month, and calculate the average of the tanh compound score
monthly_avg_tanh_compound = ds_prep.
↪groupby('year_month')['tanh_normalized_compound'].mean().reset_index()

monthly_avg_tanh_compound.head()
```

[ ]:

| | year_month | tanh_normalized_compound |
|---|---|---|
| 0 | 1997-02 | 0.058932 |
| 1 | 1997-03 | 0.044970 |
| 2 | 1997-04 | 0.068392 |
| 3 | 1997-05 | 0.063913 |
| 4 | 1997-06 | 0.092728 |

We now import our other variables

[ ]:
```
ds2 = pd.read_excel("Inputs\INFYOY 1990-2023.xlsx")
ds2['Date'] = pd.to_datetime(ds2['Date'])
ds2['year_month'] = ds2['Date'].dt.to_period('M')
ds2.drop(columns="Date")
```

[ ]:
```
     M3_YOY_MON  HICP_YOY_MON  Interbank3  M1_YOY_MON year_month
0           3.8           1.8    4.430000      -3.654    1997-02
1           3.6           1.6    4.500000      -2.444    1997-03
2           3.5           1.3    4.390000      -4.117    1997-04
3           4.0           1.4    4.300000      -2.415    1997-05
4           4.4           1.4    4.290000      -3.739    1997-06
..          …             …         …           …          …
317        -0.4           5.3    3.671810      -2.375    2023-07
318        -1.3           5.2    3.780304      -3.440    2023-08
319        -1.2           4.3    3.880048      -2.851    2023-09
320         NaN           2.9    3.967636         NaN    2023-10
321         NaN           NaN         NaN         NaN    2023-11

[322 rows x 5 columns]
```

[ ]:
```
merged = pd.merge(ds2,monthly_avg_tanh_compound, on='year_month', how='outer')
merged.drop(columns='Date')
```

[ ]:
```
     M3_YOY_MON  HICP_YOY_MON  Interbank3  M1_YOY_MON year_month  \
0           3.8           1.8    4.430000      -3.654    1997-02
1           3.6           1.6    4.500000      -2.444    1997-03
2           3.5           1.3    4.390000      -4.117    1997-04
3           4.0           1.4    4.300000      -2.415    1997-05
4           4.4           1.4    4.290000      -3.739    1997-06
..          …             …         …           …          …
317        -0.4           5.3    3.671810      -2.375    2023-07
318        -1.3           5.2    3.780304      -3.440    2023-08
319        -1.2           4.3    3.880048      -2.851    2023-09
320         NaN           2.9    3.967636         NaN    2023-10
321         NaN           NaN         NaN         NaN    2023-11

     tanh_normalized_compound
0                    0.058932
1                    0.044970
2                    0.068392
3                    0.063913
4                    0.092728
..                        …
317                  0.061241
318                  0.011663
319                  0.034852
```

```
320                    0.055033
321                         NaN

[322 rows x 6 columns]
```

### 1.2 We now do the Test the Dynamic Linear Model: https://pydlm.github.io/discounting.html

[ ]:
```
merged.to_excel("Outputs//Intermediate//merged.xlsx")
```

[ ]:
```
!pip install pydlm
```

```
Requirement already satisfied: pydlm in w:\programmi\anaconda\lib\site-packages
(0.1.1.12)
Requirement already satisfied: matplotlib in w:\programmi\anaconda\lib\site-
packages (from pydlm) (3.4.2)
Requirement already satisfied: numpy in w:\programmi\anaconda\lib\site-packages
(from pydlm) (1.20.3)
Requirement already satisfied: pillow>=6.2.0 in w:\programmi\anaconda\lib\site-
packages (from matplotlib->pydlm) (8.2.0)
Requirement already satisfied: cycler>=0.10 in w:\programmi\anaconda\lib\site-
packages (from matplotlib->pydlm) (0.10.0)
Requirement already satisfied: pyparsing>=2.2.1 in
w:\programmi\anaconda\lib\site-packages (from matplotlib->pydlm) (2.4.7)
Requirement already satisfied: python-dateutil>=2.7 in
w:\programmi\anaconda\lib\site-packages (from matplotlib->pydlm) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
w:\programmi\anaconda\lib\site-packages (from matplotlib->pydlm) (1.3.1)
Requirement already satisfied: six in w:\programmi\anaconda\lib\site-packages
(from cycler>=0.10->matplotlib->pydlm) (1.16.0)
```

[ ]:
```
from pydlm import dlm, trend, seasonality, dynamic

# Extracting the required time series from the data
time_series_data = merged[['HICP_YOY_MON', 'M1_YOY_MON_lag_12',␣
↪'M1_YOY_MON_lag_24',
                              'tanh_normalized_compound_lag_12',␣
↪'tanh_normalized_compound_lag_24']]

# The target variable
target = time_series_data['HICP_YOY_MON']

time_series_data.dropna(inplace=True)

# The features (lagged variables)
features = time_series_data[['M1_YOY_MON_lag_12', 'M1_YOY_MON_lag_24',
```

```
                              'tanh_normalized_compound_lag_12',␣
↪'tanh_normalized_compound_lag_24']].values

# Creating the Dynamic Linear Model
myDLM = dlm(target) + dynamic(features, name='dynamic', discount=0.9)

# Fitting the model
myDLM.fit()

# Summary of the fitted model
myDLM.plot()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-94-3dabba55eccb> in <module>
      2
      3 # Extracting the required time series from the data
----> 4 time_series_data = merged[['HICP_YOY_MON', 'M1_YOY_MON_lag_12',␣
 ↪'M1_YOY_MON_lag_24',
      5                               'tanh_normalized_compound_lag_12',␣
 ↪'tanh_normalized_compound_lag_24']]
      6

w:\Programmi\Anaconda\lib\site-packages\pandas\core\frame.py in␣
 ↪__getitem__(self, key)
   3028             if is_iterator(key):
   3029                 key = list(key)
-> 3030             indexer = self.loc._get_listlike_indexer(key, axis=1,␣
 ↪raise_missing=True)[1]
   3031
   3032         # take() does not accept boolean indexers

w:\Programmi\Anaconda\lib\site-packages\pandas\core\indexing.py in␣
 ↪_get_listlike_indexer(self, key, axis, raise_missing)
   1264             keyarr, indexer, new_indexer = ax._reindex_non_unique(keyarr)
   1265
-> 1266         self._validate_read_indexer(keyarr, indexer, axis,␣
 ↪raise_missing=raise_missing)
   1267         return keyarr, indexer
   1268

w:\Programmi\Anaconda\lib\site-packages\pandas\core\indexing.py in␣
 ↪_validate_read_indexer(self, key, indexer, axis, raise_missing)
   1314             if raise_missing:
   1315                 not_found = list(set(key) - set(ax))
-> 1316                 raise KeyError(f"{not_found} not in index")
   1317
```

```
   1318                 not_found = key[missing_mask]

KeyError: "['tanh_normalized_compound_lag_24',␣
 ↪'tanh_normalized_compound_lag_12'] not in index"
```

### 1.3 We now do the VAR

[ ]:
```python
merged.to_excel("Outputs//Intermediate//merged.xlsx")
```

[ ]:
```python
from statsmodels.tsa.api import VAR
import statsmodels.api as sm
import pandas as pd
from tqdm.notebook import tqdm
import os
import nltk
from nltk.tokenize import word_tokenize
import numpy as np

merged = pd.read_excel("Outputs//Intermediate//merged.xlsx")
# Set the 'Date' column as the index
merged.set_index('year_month', inplace=True)

# Drop unnecessary columns
merged.drop(columns=[ 'M3_YOY_MON', 'Interbank3'], inplace=True)

# Assuming monthly data, create 1 and 2 years lags (12 and 24 months)
merged['HICP_YOY_MON_lag_12'] = merged['HICP_YOY_MON'].shift(12)
merged['M1_YOY_MON_lag_12'] = merged['M1_YOY_MON'].shift(12)
merged['tanh_normalized_compound_lag_1'] = merged['tanh_normalized_compound'].
 ↪shift(1)

merged['HICP_YOY_MON_lag_24'] = merged['HICP_YOY_MON'].shift(24)
merged['M1_YOY_MON_lag_24'] = merged['M1_YOY_MON'].shift(24)
merged['tanh_normalized_compound_lag_2'] = merged['tanh_normalized_compound'].
 ↪shift(2)


# Drop any rows with NaN values resulting from the lag
merged.dropna(inplace=True)

selected_columns = ['HICP_YOY_MON', 'M1_YOY_MON', 'tanh_normalized_compound',
                    'HICP_YOY_MON_lag_12', 'M1_YOY_MON_lag_12',␣
 ↪'tanh_normalized_compound_lag_1',
                    'HICP_YOY_MON_lag_24', 'M1_YOY_MON_lag_24',␣
 ↪'tanh_normalized_compound_lag_2']
data_selected = merged[selected_columns]
```

```python
# Fit the VAR model
model = VAR(data_selected)
results = model.fit()  # Using 'aic' to select the optimal lag order up to 24

# Summary of the results
results_summary = results.summary()
results_summary
```

```
w:\Programmi\Anaconda\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581:
ValueWarning: A date index has been provided, but it has no associated frequency
information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
```

[ ]:

```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Sat, 18, Nov, 2023
Time:                     10:47:12
--------------------------------------------------------------------
No. of Equations:         9.00000    BIC:                   -38.4984
Nobs:                     289.000    HQIC:                  -39.1827
Log likelihood:           2127.35    FPE:                6.08949e-18
AIC:                     -39.6402    Det(Omega_mle):     4.48354e-18
--------------------------------------------------------------------
Results for equation HICP_YOY_MON
================================================================================
====================
                                       coefficient       std. error
t-stat            prob
--------------------------------------------------------------------------------
--------------------
const                                     0.112394         0.075904
1.481           0.139
L1.HICP_YOY_MON                           1.017599         0.013059
77.922           0.000
L1.M1_YOY_MON                             0.004346         0.001748
2.487           0.013
L1.tanh_normalized_compound              -0.373099         0.716800
-0.521           0.603
L1.HICP_YOY_MON_lag_12                   -0.082096         0.016635
-4.935           0.000
L1.M1_YOY_MON_lag_12                      0.001885         0.001885
1.000           0.317
L1.tanh_normalized_compound_lag_1        -0.198598         0.731134
-0.272           0.786
```

```
L1.HICP_YOY_MON_lag_24                  -0.013602         0.023226
-0.586           0.558
L1.M1_YOY_MON_lag_24                    -0.000938         0.001791
-0.524           0.600
L1.tanh_normalized_compound_lag_2        0.353799         0.722056
0.490           0.624
================================================================================
====================

Results for equation M1_YOY_MON
================================================================================
====================
                                       coefficient       std. error
t-stat            prob
--------------------------------------------------------------------------------
--------------------
const                                    1.660609         1.013155
1.639           0.101
L1.HICP_YOY_MON                         -0.542851         0.174314
-3.114           0.002
L1.M1_YOY_MON                            0.920823         0.023327
39.475           0.000
L1.tanh_normalized_compound            -15.290463         9.567800
-1.598           0.110
L1.HICP_YOY_MON_lag_12                   0.635522         0.222045
2.862           0.004
L1.M1_YOY_MON_lag_12                     0.032336         0.025162
1.285           0.199
L1.tanh_normalized_compound_lag_1       11.743320         9.759134
1.203           0.229
L1.HICP_YOY_MON_lag_24                  -0.406964         0.310014
-1.313           0.189
L1.M1_YOY_MON_lag_24                    -0.014651         0.023902
-0.613           0.540
L1.tanh_normalized_compound_lag_2       -6.569927         9.637954
-0.682           0.495
================================================================================
====================

Results for equation tanh_normalized_compound
================================================================================
====================
                                       coefficient       std. error
t-stat            prob
--------------------------------------------------------------------------------
--------------------
const                                    0.030553         0.006192
```

```
4.935           0.000
L1.HICP_YOY_MON                         -0.001435         0.001065
-1.347           0.178
L1.M1_YOY_MON                            0.000159         0.000143
1.118           0.264
L1.tanh_normalized_compound              0.188967         0.058471
3.232           0.001
L1.HICP_YOY_MON_lag_12                   0.000401         0.001357
0.295           0.768
L1.M1_YOY_MON_lag_12                     0.000204         0.000154
1.329           0.184
L1.tanh_normalized_compound_lag_1        0.083218         0.059640
1.395           0.163
L1.HICP_YOY_MON_lag_24                  -0.001521         0.001895
-0.803           0.422
L1.M1_YOY_MON_lag_24                     0.000083         0.000146
0.569           0.569
L1.tanh_normalized_compound_lag_2        0.209996         0.058900
3.565           0.000
================================================================================
====================

Results for equation HICP_YOY_MON_lag_12
================================================================================
====================
                                      coefficient       std. error
t-stat            prob
--------------------------------------------------------------------------------
--------------------
const                                    0.010866         0.063939
0.170           0.865
L1.HICP_YOY_MON                          0.065509         0.011001
5.955           0.000
L1.M1_YOY_MON                           -0.003643         0.001472
-2.475           0.013
L1.tanh_normalized_compound              0.197821         0.603809
0.328           0.743
L1.HICP_YOY_MON_lag_12                   0.995372         0.014013
71.032           0.000
L1.M1_YOY_MON_lag_12                     0.000935         0.001588
0.589           0.556
L1.tanh_normalized_compound_lag_1       -0.340429         0.615883
-0.553           0.580
L1.HICP_YOY_MON_lag_24                  -0.011196         0.019565
-0.572           0.567
L1.M1_YOY_MON_lag_24                    -0.000772         0.001508
-0.512           0.609
```

```
L1.tanh_normalized_compound_lag_2        -0.916911         0.608236
-1.507           0.132
================================================================================
====================

Results for equation M1_YOY_MON_lag_12
================================================================================
====================
                                       coefficient       std. error
t-stat            prob
--------------------------------------------------------------------------------
--------------------
const                                    -1.263971         1.021872
-1.237           0.216
L1.HICP_YOY_MON                           0.363951         0.175814
2.070           0.038
L1.M1_YOY_MON                             0.082699         0.023528
3.515           0.000
L1.tanh_normalized_compound              11.937629         9.650122
1.237           0.216
L1.HICP_YOY_MON_lag_12                   -0.693442         0.223956
-3.096           0.002
L1.M1_YOY_MON_lag_12                      0.886365         0.025379
34.926           0.000
L1.tanh_normalized_compound_lag_1         8.377730         9.843103
0.851           0.395
L1.HICP_YOY_MON_lag_24                    0.697771         0.312682
2.232           0.026
L1.M1_YOY_MON_lag_24                      0.005928         0.024108
0.246           0.806
L1.tanh_normalized_compound_lag_2        -5.235076         9.720880
-0.539           0.590
================================================================================
====================

Results for equation tanh_normalized_compound_lag_1
================================================================================
====================
                                       coefficient       std. error
t-stat            prob
--------------------------------------------------------------------------------
--------------------
const                                    -0.000102         0.000143
-0.709           0.478
L1.HICP_YOY_MON                          -0.000001         0.000025
-0.038           0.970
L1.M1_YOY_MON                             0.000001         0.000003
```

```
0.356           0.722
L1.tanh_normalized_compound              0.999157         0.001353
738.336           0.000
L1.HICP_YOY_MON_lag_12                   0.000000         0.000031
0.007           0.994
L1.M1_YOY_MON_lag_12                     0.000001         0.000004
0.272           0.786
L1.tanh_normalized_compound_lag_1        0.001595         0.001380
1.155           0.248
L1.HICP_YOY_MON_lag_24                   0.000007         0.000044
0.149           0.882
L1.M1_YOY_MON_lag_24                    -0.000003         0.000003
-0.852           0.394
L1.tanh_normalized_compound_lag_2        0.000139         0.001363
0.102           0.919
================================================================================
====================

Results for equation HICP_YOY_MON_lag_24
================================================================================
====================
                                       coefficient       std. error
t-stat            prob
--------------------------------------------------------------------------------
--------------------
const                                   -0.033394         0.059151
-0.565           0.572
L1.HICP_YOY_MON                         -0.014471         0.010177
-1.422           0.155
L1.M1_YOY_MON                            0.001507         0.001362
1.106           0.269
L1.tanh_normalized_compound             -0.159193         0.558592
-0.285           0.776
L1.HICP_YOY_MON_lag_12                   0.071921         0.012964
5.548           0.000
L1.M1_YOY_MON_lag_12                    -0.002665         0.001469
-1.814           0.070
L1.tanh_normalized_compound_lag_1        0.186730         0.569762
0.328           0.743
L1.HICP_YOY_MON_lag_24                   0.938637         0.018099
51.860           0.000
L1.M1_YOY_MON_lag_24                     0.001844         0.001395
1.322           0.186
L1.tanh_normalized_compound_lag_2        0.554265         0.562688
0.985           0.325
================================================================================
====================
```

```
Results for equation M1_YOY_MON_lag_24
================================================================================
====================
                                       coefficient       std. error
t-stat            prob
--------------------------------------------------------------------------------
--------------------
const                                     0.188308         1.001686
0.188           0.851
L1.HICP_YOY_MON                           0.031014         0.172340
0.180           0.857
L1.M1_YOY_MON                             0.012138         0.023063
0.526           0.599
L1.tanh_normalized_compound               2.544015         9.459487
0.269           0.788
L1.HICP_YOY_MON_lag_12                    0.209821         0.219532
0.956           0.339
L1.M1_YOY_MON_lag_12                      0.097187         0.024877
3.907           0.000
L1.tanh_normalized_compound_lag_1        -4.839571         9.648656
-0.502           0.616
L1.HICP_YOY_MON_lag_24                   -0.571532         0.306505
-1.865           0.062
L1.M1_YOY_MON_lag_24                      0.916882         0.023632
38.799           0.000
L1.tanh_normalized_compound_lag_2         3.572182         9.528848
0.375           0.708
================================================================================
====================

Results for equation tanh_normalized_compound_lag_2
================================================================================
====================
                                       coefficient       std. error
t-stat            prob
--------------------------------------------------------------------------------
--------------------
const                                    -0.000028         0.000082
-0.336           0.737
L1.HICP_YOY_MON                           0.000003         0.000014
0.216           0.829
L1.M1_YOY_MON                            -0.000000         0.000002
-0.019           0.985
L1.tanh_normalized_compound               0.000621         0.000776
0.800           0.424
L1.HICP_YOY_MON_lag_12                   -0.000001         0.000018
```

```
-0.069          0.945
L1.M1_YOY_MON_lag_12                      0.000000         0.000002
0.102          0.919
L1.tanh_normalized_compound_lag_1         0.999680         0.000792
1262.523          0.000
L1.HICP_YOY_MON_lag_24                    0.000007         0.000025
0.281          0.778
L1.M1_YOY_MON_lag_24                      0.000002         0.000002
0.841          0.400
L1.tanh_normalized_compound_lag_2         0.000101         0.000782
0.130          0.897
================================================================================
====================

Correlation matrix of residuals
                                  HICP_YOY_MON  M1_YOY_MON
tanh_normalized_compound  HICP_YOY_MON_lag_12  M1_YOY_MON_lag_12
tanh_normalized_compound_lag_1  HICP_YOY_MON_lag_24  M1_YOY_MON_lag_24
tanh_normalized_compound_lag_2
HICP_YOY_MON                          1.000000   -0.038982
0.030954            -0.405101            0.010862                        0.079943
-0.088193          -0.036342                       -0.115433
M1_YOY_MON                           -0.038982    1.000000
-0.016059             0.004663            -0.509273
0.035903             0.060184            0.004365                        0.050389
tanh_normalized_compound              0.030954   -0.016059
1.000000             0.041384            0.078988                       -0.012267
0.045930          -0.057383                        0.016035
HICP_YOY_MON_lag_12                  -0.405101    0.004663
0.041384             1.000000            0.006153                       -0.112895
-0.372172           0.004520                        0.086142
M1_YOY_MON_lag_12                     0.010862   -0.509273
0.078988             0.006153            1.000000                       -0.091501
0.046486          -0.540554                       -0.090499
tanh_normalized_compound_lag_1        0.079943    0.035903
-0.012267             -0.112895            -0.091501
1.000000            -0.123498            0.012373                       -0.831044
HICP_YOY_MON_lag_24                  -0.088193    0.060184
0.045930            -0.372172            0.046486                       -0.123498
1.000000          -0.050110                        0.049742
M1_YOY_MON_lag_24                    -0.036342    0.004365
-0.057383             0.004520            -0.540554
0.012373            -0.050110            1.000000                        0.031372
tanh_normalized_compound_lag_2       -0.115433    0.050389
0.016035             0.086142            -0.090499                       -0.831044
0.049742            0.031372                        1.000000
```

### 1.4 We now use the distributed lag model

[ ]:
```python
import statsmodels.api as sm
merged = pd.read_excel("Outputs//Intermediate//merged.xlsx")

# Define the dependent and independent variables
dependent_var = merged['HICP_YOY_MON']
independent_vars = merged[['M1_YOY_MON', 'tanh_normalized_compound',
                           'M1_YOY_MON_lag_12', 'tanh_normalized_compound_lag_12',
                           'M1_YOY_MON_lag_24',␣
 ↪'tanh_normalized_compound_lag_24']]

# Add a constant to the independent variables
independent_vars = sm.add_constant(independent_vars)

# Create the model
model = sm.OLS(dependent_var, independent_vars)

# Fit the model
results = model.fit()

# Summary of the results
results_summary = results.summary()
results_summary
```

[ ]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:           HICP_YOY_MON   R-squared:                       0.137
Model:                            OLS   Adj. R-squared:                  0.118
Method:                 Least Squares   F-statistic:                     7.090
Date:                Sat, 18 Nov 2023   Prob (F-statistic):           5.05e-07
Time:                        02:36:08   Log-Likelihood:                -531.75
No. Observations:                 275   AIC:                             1077.
Df Residuals:                     268   BIC:                             1103.
Df Model:                           6
Covariance Type:            nonrobust
================================================================================
===================
                                      coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
-------------------
const                               2.3100      0.351      6.590      0.000
```

```
1.620       3.000
M1_YOY_MON                      -0.0284      0.010     -2.964      0.003
-0.047      -0.010
tanh_normalized_compound        -5.4811      3.828     -1.432      0.153
-13.017      2.055
M1_YOY_MON_lag_12                0.0368      0.010      3.855      0.000
0.018       0.056
tanh_normalized_compound_lag_12 -5.2398      3.879     -1.351      0.178
-12.876      2.397
M1_YOY_MON_lag_24                0.0327      0.010      3.428      0.001
0.014       0.051
tanh_normalized_compound_lag_24 -0.8862      3.953     -0.224      0.823
-8.670       6.897
==============================================================================
Omnibus:                      115.526   Durbin-Watson:                   0.080
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              435.901
Skew:                           1.791   Prob(JB):                     2.21e-95
Kurtosis:                       8.021   Cond. No.                         793.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

## 1.5 Distributed Lag but differet lags

[ ]:
```python
import statsmodels.api as sm
merged = pd.read_excel("Outputs//Intermediate//merged.xlsx")

# Assuming 'merged' is your DataFrame
# Create the lagged variable
merged['tanh_normalized_compound_lag_1'] = merged['tanh_normalized_compound'].
 ↪shift(1)

merged['M1_YOY_MON_lag_12'] = merged['M1_YOY_MON'].shift(12)
merged['M1_YOY_MON_lag_24'] = merged['M1_YOY_MON'].shift(24)
merged.dropna(inplace=True)




# Define the dependent variable after dropping NaN values
dependent_var = merged['HICP_YOY_MON']

# Define the independent variables
independent_vars = merged[['tanh_normalized_compound', 'M1_YOY_MON_lag_12',
```

```
                                   'tanh_normalized_compound_lag_1',␣
 ↪'M1_YOY_MON_lag_24']]
merged.set_index('year_month', inplace=True)


# Add a constant to the independent variables
independent_vars = sm.add_constant(independent_vars)

# Create the model
model = sm.OLS(dependent_var, independent_vars)

# Fit the model
results = model.fit()

# Summary of the results
results_summary = results.summary()
results_summary
```

[ ]:

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:            HICP_YOY_MON   R-squared:                       0.103
Model:                             OLS   Adj. R-squared:                  0.091
Method:                  Least Squares   F-statistic:                     8.250
Date:                 Sat, 18 Nov 2023   Prob (F-statistic):           2.60e-06
Time:                         10:50:39   Log-Likelihood:                -578.55
No. Observations:                  292   AIC:                             1167.
Df Residuals:                      287   BIC:                             1185.
Df Model:                            4
Covariance Type:             nonrobust
================================================================================
==================
                                     coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
------------------
const                              2.3679      0.299      7.921      0.000
1.780       2.956
tanh_normalized_compound          -7.5744      3.965     -1.911      0.057
-15.378       0.229
M1_YOY_MON_lag_12                  0.0299      0.010      3.065      0.002
0.011       0.049
tanh_normalized_compound_lag_1    -7.9915      3.961     -2.017      0.045
-15.789      -0.194
M1_YOY_MON_lag_24                  0.0401      0.009      4.251      0.000
0.022       0.059
```

```
==============================================================================
Omnibus:                      116.298   Durbin-Watson:                   0.067
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              389.295
Skew:                           1.760   Prob(JB):                     2.92e-85
Kurtosis:                       7.428   Cond. No.                         711.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```