In [1]:
```
# This notebook prepares the dataframes for analysis and runs
# the VAR on the combined model (economic and sentiment features) and
# the economic (only) model.
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.vector_ar.var_model import VAR
from sklearn.metrics import r2_score
from statsmodels.stats.stattools import durbin_watson
from scipy.stats.distributions import chi2
import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [5]:
```
# Quick look at the first two lines of the econ CSV
with open('data/econ_vars.csv') as f:
    print(f.readline())
    next(f)
    print(f.readline())
```

```
month_date,ur,pi_gr,nyse_gr,gdp_log_diff,tb_sqrt_diff,ics_log_diff,pce_gr_diff

2011-01-01,9.1,0.002028679786119758,0.016528644263536174,-
0.002411316330585225,0.013132595943347536,-0.004034975212179326,-
0.004803188602163777
```

In [6]:
```
# Set data types for the econ features
dts = {"month_date": str, "ur": np.float64
    , "pi_gr": np.float64, "nyse_gr": np.float64
    , "gdp_log_diff": np.float64, "tb_sqrt_diff": np.float64
    , "ics_log_diff": np.float64, "pce_gr_diff": np.float64}
```

In [7]:
```
# Import econ data
data_raw = pd.read_csv("data/econ_vars.csv"
                   , sep=","
                   , skiprows=0
                   , dtype=dts)
```

In [8]:
```
# Check the shape of the raw econ data
data_raw.shape
```

Out[8]:
```
(91, 8)
```

In [9]:
```
# Copy raw data
data = data_raw.copy()
```

In [10]:
```
# Sort descending and reset index
data.sort_index(ascending=False, inplace=True)
data.reset_index(drop=True, inplace=True)
```

In [11]:
```
# Check the dataframe
data.head()
```

Out[11]:

	month_date	ur	pi_gr	nyse_gr	gdp_log_diff	tb_sqrt_diff	ics_log_diff	pce_gr_diff
0	2018-06-01	4.0	0.003863	-0.001827	0.000000	0.014587	0.002039	-0.001110
1	2018-05-01	3.8	0.003728	0.000941	0.010188	0.037168	-0.008130	0.008168
2	2018-04-01	3.9	-0.004827	-0.002903	0.010188	0.022809	-0.025975	-0.008891
3	2018-03-01	4.1	0.003938	-0.015846	0.000000	0.050844	0.016907	0.007407
4	2018-02-01	4.1	0.003282	-0.053517	0.005479	0.065562	0.040947	0.001663

In [12]:
```
# Quick look at the first two lines of the sentiment CSV
with open('data/rev_means_vars_stationary.csv') as f:
    print(f.readline())
    next(f)
    print(f.readline())
```

```
date,perc_pos_rev_weighted,perc_neg_rev_weighted,perc_uncert_rev_weighted,perc_l
itig_rev_weighted,perc_modal_wk_rev_weighted,perc_modal_mod_rev_weighted,perc_co
nstrain_rev_weighted,perc_modal_str_rev_weighted_diff

2011-03-01,1.9607048390000001,0.979452392,0.957424913,0.073102265,0.387478896,0.
560045154,0.142131806,-0.0495581759999999
```

In [13]:
```
# Import sentiment data
sent_raw = pd.read_csv("data/rev_means_vars_stationary.csv"
    , sep=","
    , skiprows=0
    #, dtype=dts
    , usecols=[0,1,2,3,4,5,6,7,8]
    )
```

In [14]:
```
# Check the sentiment data types
sent_raw.dtypes
```

Out[14]:
```
date                                  object
perc_pos_rev_weighted                float64
perc_neg_rev_weighted                float64
perc_uncert_rev_weighted             float64
perc_litig_rev_weighted              float64
perc_modal_wk_rev_weighted           float64
perc_modal_mod_rev_weighted          float64
perc_constrain_rev_weighted          float64
perc_modal_str_rev_weighted_diff     float64
dtype: object
```

In [15]:
```
# Drop any observations with Nan
sent_raw.dropna(inplace=True)
```

In [16]:
```
#sort and reset the index
sent_raw.sort_values(by='date', ascending=False, inplace=True)
sent_raw.reset_index(drop=True, inplace=True)
```

In [17]:
```
# Confirm the shape of the dataframe
sent_raw.shape
```

Out[17]: (89, 9)

In [18]:
```
# Check the dataframe
sent_raw.head()
```

Out[18]:

	date	perc_pos_rev_weighted	perc_neg_rev_weighted	perc_uncert_rev_weighted	perc_litig
0	2018-06-01	2.043836	0.890239	0.815790	0.094844
1	2018-05-01	1.915107	1.074677	0.803269	0.082565
2	2018-04-01	1.926137	1.068873	0.777739	0.095691
3	2018-03-01	2.016781	0.979256	0.800769	0.102810
4	2018-02-01	2.004129	0.963727	0.815629	0.098500

In [19]:
```
# Copy sentiment data
sent = sent_raw.copy()
```

In [20]:
```
# Filter econ observations to match the sentiment dataframe
data = data.iloc[0:89,:]
```

In [21]:
```
# Combine the econ and sentiment dataframes
data = pd.concat([data,sent], axis=1)
```

In [22]:
```
# Replace the integer index with the date column
data = data.set_index("date")
```

In [23]:
```
# Drop the month_date column and the gdp column
data = data.drop(data.columns[[0,4]], axis=1)
```

In [24]:
```
# Check the dataframe
data.head()
```

Out[24]:

	ur	pi_gr	nyse_gr	tb_sqrt_diff	ics_log_diff	pce_gr_diff	perc_pos_rev_weighted	p
date								
2018-06-01	4.0	0.003863	-0.001827	0.014587	0.002039	-0.001110	2.043836	0
2018-05-01	3.8	0.003728	0.000941	0.037168	-0.008130	0.008168	1.915107	1
2018-04-01	3.9	-0.004827	-0.002903	0.022809	-0.025975	-0.008891	1.926137	1
2018-03-01	4.1	0.003938	-0.015846	0.050844	0.016907	0.007407	2.016781	0
2018-02-01	4.1	0.003282	-0.053517	0.065562	0.040947	0.001663	2.004129	0

In [25]:
```
# Copy econ features to econ dataframe and drop all sentiment variables
econ = data.drop(data.columns[[6,7,8,9,10,11,12,13]], axis=1)
```

In [26]:
```
# Check the dataframe
econ.head()
```

Out[26]:

	ur	pi_gr	nyse_gr	tb_sqrt_diff	ics_log_diff	pce_gr_diff
date						
2018-06-01	4.0	0.003863	-0.001827	0.014587	0.002039	-0.001110
2018-05-01	3.8	0.003728	0.000941	0.037168	-0.008130	0.008168
2018-04-01	3.9	-0.004827	-0.002903	0.022809	-0.025975	-0.008891
2018-03-01	4.1	0.003938	-0.015846	0.050844	0.016907	0.007407
2018-02-01	4.1	0.003282	-0.053517	0.065562	0.040947	0.001663

In [27]:
```
# Instantiate the VAR model using the combined econ and sentiment variables
comb_model = VAR(data)
```

In [28]:
```
# Fit the model to the combined variables
comb_fitted = comb_model.fit(maxlags=2, ic='bic', verbose=True, trend='c')
```

```
                 VAR Order Selection
=====================================================
           aic          bic          fpe         hqic
-----------------------------------------------------
0       -91.97       -91.57    1.141e-40       -91.81
1      -99.89*      -93.94*   4.360e-44*      -97.49*
2       -98.88       -87.37    1.649e-43       -94.25
=====================================================
* Minimum

Using 1 based on bic criterion
```

In [29]:
```
# Print a summary of the models
comb_fitted.summary()
```

Out[29]:
```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Tue, 18, Jun, 2019
Time:                     17:21:51
--------------------------------------------------------------------
No. of Equations:         14.0000    BIC:                   -93.7304
Nobs:                     88.0000    HQIC:                  -97.2605
Log likelihood:           2846.13    FPE:                5.57649e-44
AIC:                     -99.6422    Det(Omega_mle):     6.15740e-45
--------------------------------------------------------------------
Results for equation ur
================================================================================
======================
                                         coefficient       std. error
t-stat            prob
--------------------------------------------------------------------------------
----------------------
const                                      -0.827841         0.471726
-1.755           0.083
L1.ur                                       1.011513         0.011859
85.292           0.000
L1.pi_gr                                    1.511155         1.787599
0.845           0.401
L1.nyse_gr                                  0.130882         0.435033
0.301           0.764
L1.tb_sqrt_diff                             0.289768         0.323562
0.896           0.373
L1.ics_log_diff                             1.056703         0.304881
3.466           0.001
L1.pce_gr_diff                             -1.561460         2.838331
-0.550           0.584
L1.perc_pos_rev_weighted                    0.085887         0.103277
0.832           0.408
L1.perc_neg_rev_weighted                    0.197955         0.232477
0.852           0.397
L1.perc_uncert_rev_weighted                 0.367334         0.491891
0.747           0.458
L1.perc_litig_rev_weighted                  0.080162         0.983426
0.082           0.935
L1.perc_modal_wk_rev_weighted              -0.371585         0.736801
-0.504           0.616
L1.perc_modal_mod_rev_weighted              0.439772         0.461822
0.952           0.344
L1.perc_constrain_rev_weighted             -0.172817         0.992679
-0.174           0.862
L1.perc_modal_str_rev_weighted_diff        -0.557680         0.349682
-1.595           0.115
================================================================================
======================

Results for equation pi_gr
================================================================================
======================
                                         coefficient       std. error
t-stat            prob
--------------------------------------------------------------------------------
----------------------
const                                       0.023636         0.029573
0.799           0.427
L1.ur                                      -0.000248         0.000743
-0.333           0.740
L1.pi_gr                                   -0.191584         0.112067
-1.710           0.092
```

In [30]:
```
# Calculate the combined r squared
y_true = data['pce_gr_diff'].iloc[1:89]
y_pred = comb_fitted.resid['pce_gr_diff'] + y_true
print ("Combined model R^2: {}".format(r2_score(y_true, y_pred)))
```

```
Combined model R^2: 0.3033409524231241
```

In [31]:
```
# Calculate the mean squared error of the combined model
comb_mse = np.mean(comb_fitted.resid['pce_gr_diff']**2)
print ("Combined model MSE: {}".format(comb_mse))
```

```
Combined model MSE: 1.746804553927309e-05
```

In [32]:
```
# Run the Durbin Watson test
print ("Combined model Durbin-Watson test: {}".format(durbin_watson(comb_fitted.re
sid['pce_gr_diff'])))
```

```
Combined model Durbin-Watson test: 2.5328634408873234
```

In [33]:
```
# Instantiate the VAR model using the econ variables
econ_model = VAR(econ)
```

In [34]:
```
# Fit the model to the econ data
econ_fitted = econ_model.fit(maxlags=2, ic='bic', verbose=True, trend='c')
```

```
                 VAR Order Selection
=====================================================
           aic          bic          fpe         hqic
-----------------------------------------------------
0       -38.34       -38.17    2.228e-17       -38.27
1       -43.66      -42.47*    1.100e-19       -43.18
2      -44.13*       -41.92   6.936e-20*      -43.24*
=====================================================
* Minimum

Using 1 based on bic criterion
```

In [35]:
```
# Print a summary of the models
econ_fitted.summary()
```

Out[35]:
```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Tue, 18, Jun, 2019
Time:                     17:21:51
--------------------------------------------------------------------
No. of Equations:         6.00000    BIC:                   -42.4596
Nobs:                     88.0000    HQIC:                  -43.1656
Log likelihood:           1213.05    FPE:                1.11537e-19
AIC:                     -43.6420    Det(Omega_mle):     7.04649e-20
--------------------------------------------------------------------
Results for equation ur
================================================================================
==
                     coefficient       std. error           t-stat            pr
ob
--------------------------------------------------------------------------------
--
const                   0.009825         0.056289            0.175
0.862
L1.ur                   1.006361         0.008557          117.611
0.000
L1.pi_gr                1.210979         1.696079            0.714
0.477
L1.nyse_gr              0.131370         0.409557            0.321
0.749
L1.tb_sqrt_diff         0.201771         0.293805            0.687
0.494
L1.ics_log_diff         0.960370         0.287296            3.343
0.001
L1.pce_gr_diff         -1.438920         2.749622           -0.523
0.602
================================================================================
==

Results for equation pi_gr
================================================================================
==
                     coefficient       std. error           t-stat            pr
ob
--------------------------------------------------------------------------------
--
const                   0.004612         0.003593            1.284
0.203
L1.ur                  -0.000387         0.000546           -0.708
0.481
L1.pi_gr               -0.138910         0.108260           -1.283
0.203
L1.nyse_gr              0.009626         0.026142            0.368
0.714
L1.tb_sqrt_diff        -0.012863         0.018753           -0.686
0.495
L1.ics_log_diff        -0.022995         0.018338           -1.254
0.213
L1.pce_gr_diff         -0.257348         0.175507           -1.466
0.146
================================================================================
==

Results for equation nyse_gr
================================================================================
==
                     coefficient       std. error           t-stat            pr
ob
```

In [36]:
```
# Calculate the combined r squared
y_true = econ['pce_gr_diff'].iloc[1:89]
y_pred = econ_fitted.resid['pce_gr_diff'] + y_true
print ("Economic model R^2: {}".format(r2_score(y_true, y_pred)))
```

```
Economic model R^2: 0.28749324079974536
```

In [37]:
```
# Calculate the mean squared error of the econ model
econ_mse = np.mean(econ_fitted.resid['pce_gr_diff']**2)
print ("Economic model MSE: {}".format(econ_mse))
```

```
Economic model MSE: 1.7865411437689707e-05
```

In [38]:
```
# Run the Durbin Watson test
print ("Economic model Durbin-Watson test: {}".format (durbin_watson(econ_fitted.r
esid['pce_gr_diff'])))
```

```
Economic model Durbin-Watson test: 2.5575996229732354
```

In [39]:
```
# View the log likelihood values
print ("Combined model Log Likelihood: {}".format(comb_fitted.llf))
print ("Economic model Log Likelihood: {}".format(econ_fitted.llf))
```

```
Combined model Log Likelihood: 2846.125874607882
Economic model Log Likelihood: 1213.0463632573587
```

In [40]:
```
# Create a function that calculates the likelihood ratio test
def likelihood_ratio(ll_model_1, ll_model_2):  # Model 1 is the more restrictive e
con model
    return (2*(ll_model_2 - ll_model_1))
```

In [41]:
```
import math
# Calculate the likelihood ratio and run the likelihood ratio test on a chi square
distribution
LR = likelihood_ratio(econ_fitted.llf, comb_fitted.llf)
pval = chi2.sf(LR, 8) # Combined model +8 variables 8 d.f.
print ("Log Likelihood Ratio test p-value: {:4.03f}".format((pval)))
```

```
Log Likelihood Ratio test p-value: 0.000
```