data_exploration_rendered

January 29, 2025

1 Data exploration in advance of model selection

Thanks to neptune.ai for their great tutorials in visualization; they were quite helpful here. Much of the visualization in this section was described helpfully here.

1.0.1 Import Libraries

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly
import plotly.express as px
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.tsa.stattools import adfuller
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

1.0.2 Import data

The CSV here is an extract of the most recent Google BigQuery run of 'weekly_variables_flattened.sql'. My intent was to do this analysis in Google Colab, but as it turns out, activating that feature in my personal cloud instance would cost me about $400 on the low end, so instead I'll be doing it here.

[2]:
```
# import data extracted from last GCP run of 'weekly_variables_flattened'
df = pd.read_csv("weekly_variables_flattened.csv")
df['week_start'] = pd.to_datetime(df['week_start'])
df = df.sort_values(by='week_start', ascending=True)
df.set_index('week_start', inplace=True)
```

[3]:
```
print(df.head())
print(len(df))
# df = df.dropna()
print(len(df))
```

```
            approving  disapproving  unsure_or_no_data  BusinessApplications  \
week_start
```

```
2004-07-04       48.0     48.333333           3.666667                159034
2004-07-11       48.0     48.333333           3.666667                159034
2004-07-18       48.0     48.333333           3.666667                159034
2004-07-25       48.0     48.333333           3.666667                159034
2004-08-01       50.0     46.500000           3.500000                191673

            ConstructionSpending  DurableGoodsNewOrders  \
week_start
2004-07-04             1006184.0              186835.0
2004-07-11             1006184.0              186835.0
2004-07-18             1006184.0              186835.0
2004-07-25             1006184.0              186835.0
2004-08-01             1013616.0              183728.0

            InternationalTrade_Exports  ManuInventories  ManuNewOrders  \
week_start
2004-07-04                     96907.0         426260.0       356710.0
2004-07-11                     96907.0         426260.0       356710.0
2004-07-18                     96907.0         426260.0       356710.0
2004-07-25                     96907.0         426260.0       356710.0
2004-08-01                     97295.0         429820.0       355629.0

            NewHomesForSale  NewHomesSold  ResConstPermits  \
week_start
2004-07-04            400.0        1088.0             2112
2004-07-11            400.0        1088.0             2112
2004-07-18            400.0        1088.0             2112
2004-07-25            400.0        1088.0             2112
2004-08-01            405.0        1175.0             2056

            ResConstUnitsCompleted  ResConstUnitsStarted  RetailInventories  \
week_start
2004-07-04                    1881                  2002           454032.0
2004-07-11                    1881                  2002           454032.0
2004-07-18                    1881                  2002           454032.0
2004-07-25                    1881                  2002           454032.0
2004-08-01                    1911                  2024           457271.0

            SalesForRetailAndFood  WholesaleInventories  orders_outcome_var
week_start
2004-07-04                 318549              326753.0                   3
2004-07-11                 318549              326753.0                   6
2004-07-18                 318549              326753.0                   0
2004-07-25                 318549              326753.0                  12
2004-08-01                 318977              330102.0                   0
1070
1070
```

1.0.3 Review NA values

We know that there are instances where census data lags, or some components of the economic indicators were not included in certain time periods. These will need to be examined prior to analysis:

[4]:
```
null_rows = df[df.isnull().any(axis=1)]

print(null_rows)
print(f'{len(null_rows)} rows with null values found: {len(null_rows)} out of␣
  ↪{len(df)} total')
```

```
            approving  disapproving  unsure_or_no_data  BusinessApplications  \
week_start
2021-12-05        NaN           NaN                NaN                428051
2021-12-12        NaN           NaN                NaN                428051
2021-12-19        NaN           NaN                NaN                428051
2021-12-26        NaN           NaN                NaN                428051
2022-12-04        NaN           NaN                NaN                423977
2022-12-11        NaN           NaN                NaN                423977
2022-12-18        NaN           NaN                NaN                423977
2022-12-25        NaN           NaN                NaN                423977
2024-12-01       39.0          56.0                5.0                457544
2024-12-08       39.0          56.0                5.0                457544
2024-12-15       39.0          56.0                5.0                457544
2024-12-22       39.0          56.0                5.0                457544
2024-12-29       39.0          56.0                5.0                457544

            ConstructionSpending  DurableGoodsNewOrders  \
week_start
2021-12-05             1757320.0                266894.0
2021-12-12             1757320.0                266894.0
2021-12-19             1757320.0                266894.0
2021-12-26             1757320.0                266894.0
2022-12-04             1922389.0                278589.0
2022-12-11             1922389.0                278589.0
2022-12-18             1922389.0                278589.0
2022-12-25             1922389.0                278589.0
2024-12-01                   NaN                     NaN
2024-12-08                   NaN                     NaN
2024-12-15                   NaN                     NaN
2024-12-22                   NaN                     NaN
2024-12-29                   NaN                     NaN

            InternationalTrade_Exports  ManuInventories  ManuNewOrders  \
week_start
2021-12-05                    235613.0         808491.0       549680.0
2021-12-12                    235613.0         808491.0       549680.0
2021-12-19                    235613.0         808491.0       549680.0
```

2021-12-26	235613.0	808491.0	549680.0
2022-12-04	252504.0	859100.0	576183.0
2022-12-11	252504.0	859100.0	576183.0
2022-12-18	252504.0	859100.0	576183.0
2022-12-25	252504.0	859100.0	576183.0
2024-12-01	NaN	NaN	NaN
2024-12-08	NaN	NaN	NaN
2024-12-15	NaN	NaN	NaN
2024-12-22	NaN	NaN	NaN
2024-12-29	NaN	NaN	NaN

week_start	NewHomesForSale	NewHomesSold	ResConstPermits	\
2021-12-05	386.0	834.0	1913	
2021-12-12	386.0	834.0	1913	
2021-12-19	386.0	834.0	1913	
2021-12-26	386.0	834.0	1913	
2022-12-04	451.0	632.0	1400	
2022-12-11	451.0	632.0	1400	
2022-12-18	451.0	632.0	1400	
2022-12-25	451.0	632.0	1400	
2024-12-01	NaN	NaN	1483	
2024-12-08	NaN	NaN	1483	
2024-12-15	NaN	NaN	1483	
2024-12-22	NaN	NaN	1483	
2024-12-29	NaN	NaN	1483	

week_start	ResConstUnitsCompleted	ResConstUnitsStarted	RetailInventories	\
2021-12-05	1326	1757	663693.0	
2021-12-12	1326	1757	663693.0	
2021-12-19	1326	1757	663693.0	
2021-12-26	1326	1757	663693.0	
2022-12-04	1386	1340	747278.0	
2022-12-11	1386	1340	747278.0	
2022-12-18	1386	1340	747278.0	
2022-12-25	1386	1340	747278.0	
2024-12-01	1544	1499	NaN	
2024-12-08	1544	1499	NaN	
2024-12-15	1544	1499	NaN	
2024-12-22	1544	1499	NaN	
2024-12-29	1544	1499	NaN	

week_start	SalesForRetailAndFood	WholesaleInventories	orders_outcome_var
2021-12-05	632515	785340.0	0
2021-12-12	632515	785340.0	3
2021-12-19	632515	785340.0	1

```
2021-12-26                632515              785340.0                   1
2022-12-04                666734              922988.0                   0
2022-12-11                666734              922988.0                   0
2022-12-18                666734              922988.0                   1
2022-12-25                666734              922988.0                   1
2024-12-01                729191                   NaN                   0
2024-12-08                729191                   NaN                   0
2024-12-15                729191                   NaN                   0
2024-12-22                729191                   NaN                   3
2024-12-29                729191                   NaN                   2
13 rows with null values found: 13 out of 1070 total
```

Given the relatively small number of NAs in the data - just over 1% of the total - we will proceed with exploration, and interpolate these prior to model building.

1.1 Data visualizations: The Dependent (Outcome) Variable

It is unfortunate that we do not have sufficient data at the current time to build a model through the current administration; at the time of this writing, the full tenure of the Biden presidency is not reflected in this data.

That said, in addition to the quality checks done in previous instances, the visualization in the first figure below can be compared to prior work to see that the structure of the data is as would be expected.

1.1.1 Seasonality

First, a simple trend of the outcome variable 'orders_outcome_var': we find some clear outliers that may need to be addressed prior to model implementation.

[5]:
```
fig1 = px.line(df, x=df.index, y='orders_outcome_var')
fig1.update_layout(
    xaxis_title = 'Date',
    yaxis_title = 'No. Exec. Orders',
    title = 'Executive Orders by Date'
)
fig1.show()
```

We see below that there is a clear seasonal component to the data: this will require differencing (and suggests a SARIMA regression model be used).

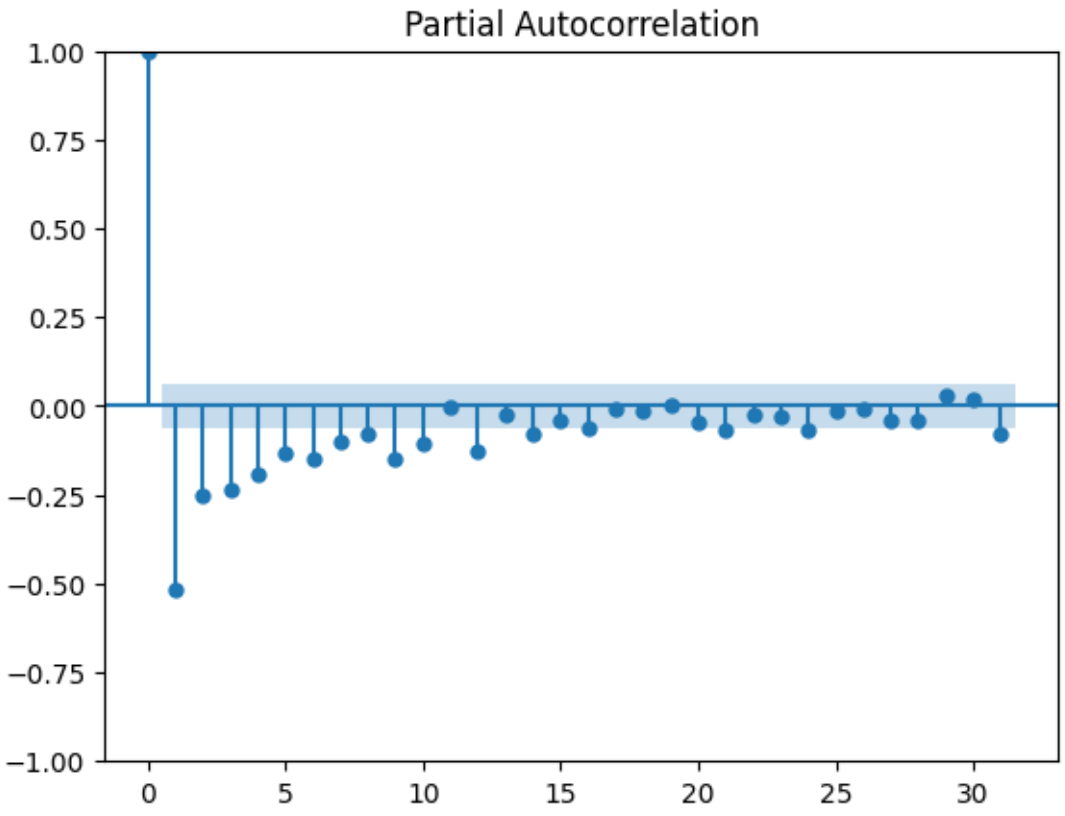
[6]:
```
fig2 = seasonal_decompose(x=df['orders_outcome_var'], model='additive')

fig2 = fig2.plot()

fig2.tight_layout(pad = 3)

fig2 = plotly.tools.mpl_to_plotly(fig2)
fig2.update_layout(
    title = 'Seasonal Decomposition showing strong seasonality',
```

```
    title_x = .1,
    title_y = .97
)
fig2.show();
```

1.1.2 Autocorrelation

Prior to looking for autocorrelation, we will difference the data based on the seasonal period of one year. Plotting *partial autocorrection correlation* indicates that there is a strong autocorrelation between high values in one period and a high value in the following time period.

[7]:
```
df['orders_outcome_var_diff']=df['orders_outcome_var']-df['orders_outcome_var'].
↪shift()
plot_pacf(df['orders_outcome_var_diff'].dropna());
```



1.1.3 Stationarity

Given the wobbly trend line in the figure above, we will use the Dickey-Fuller test to examine stationarity. This is performed on the differenced data.

The results below indicate that the differenced data is now stationary ($p < 0.05$, by no small amount.)

[8]:
```
adf, pval, usedlag, nobs, crit_vals, icbest =␣
↪adfuller(df['orders_outcome_var_diff'].dropna().values)

print('ADF test statistic:', adf)
print('ADF p-value:', pval)
print('Number of lags used:', usedlag)
print('Number of observations:', nobs)
print('Critical values:', crit_vals)
print('Best information criterion:', icbest)
```

```
ADF test statistic: -13.51328357251598
ADF p-value: 2.826005937992845e-25
Number of lags used: 15
Number of observations: 1053
Critical values: {'1%': np.float64(-3.4365753682419133), '5%':
np.float64(-2.8642886771163396), '10%': np.float64(-2.568233502009814)}
Best information criterion: 5778.721360947314
```

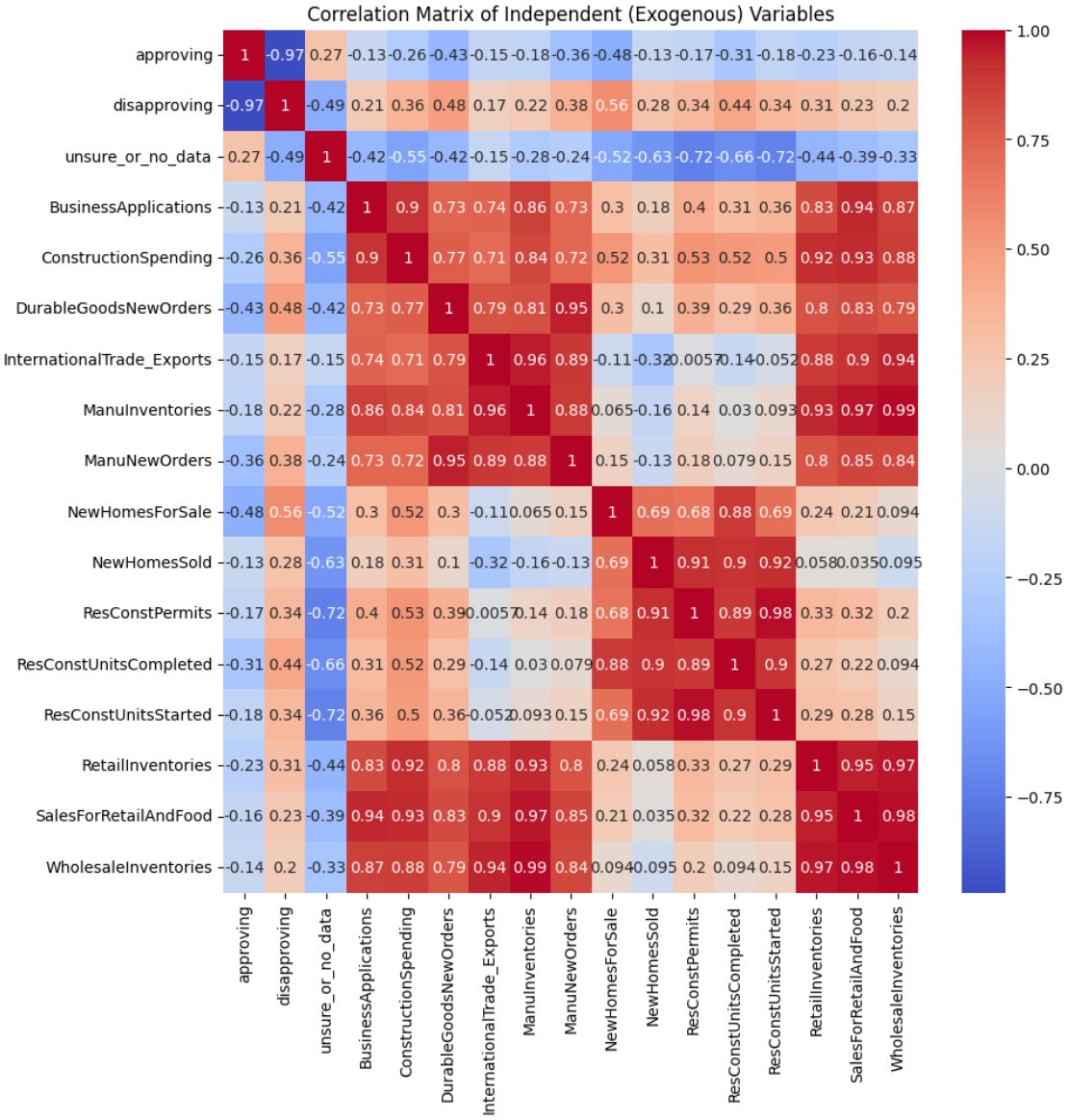
Given these results, we will use a **Seasonal Auto-Regressive Integrated Moving Average with Exogenous Variable** (SARIMAX) model to attempt to predict future counts of executive orders by week.

1.2 Data Visualizations: The Independent 'Exogenous' Variables

In a SARIMAX model, the exogenous variables are treated as ordinary least squares regressors and need to be examined for multicollinearity.

Below, significant multicollnearity (> 0.8) can be seen in many of the exogenous variables; referencing work with similarly multicollinear characteristics, we will - rather than attempting to use stepwise or exhaustive feature analysis - perform principal component analysis (PCA) on the exogenous variable set and use those principal components as 'exog' input in the SARIMAX model.

[9]:
```
df_exog_vars = df.drop(['orders_outcome_var', 'orders_outcome_var_diff'], axis␣
↪= 1)
correlation_matrix = df_exog_vars.corr()
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', ax=ax).set(title =␣
↪'Correlation Matrix of Independent (Exogenous) Variables')

plt.show()
```



1.3 Missing Values in Exogenous Variables

There are some time periods where the census data is missing certain economic indicator values, and we will use spline interpolation to fill those values (as neither PCA nor SARIMAX allows for missing values). Line graphs before and after are reviewed to confirm that interpolation does not result in visible trend change.

[10]:
```
fig4 = px.line(df_exog_vars)
fig4.update_layout(
    xaxis_title = 'Date',
    yaxis_title = 'No. Exec. Orders',
    title = 'Exogenous Variables as Line Graph'
```

```
)
fig4.show()
null_rows = df_exog_vars[df_exog_vars.isnull().any(axis=1)]
print(f'{len(null_rows)} rows with null values found: {len(null_rows)} out of␣
 ↪{len(df_exog_vars)} total')
```

```
13 rows with null values found: 13 out of 1070 total
```

[11]:
```
df_exog_interpolated = df_exog_vars.interpolate(method='spline', order = 3) #␣
 ↪order 3 is cubic
fig5 = px.line(df_exog_interpolated)
fig5.update_layout(
    xaxis_title = 'Date',
    yaxis_title = 'No. Exec. Orders',
    title = 'Exogenous Variables (Interpolated) as Line Graph'
)
fig4.show()
null_rows = df_exog_interpolated[df_exog_interpolated.isnull().any(axis=1)]
print(f'{len(null_rows)} rows with null values found: {len(null_rows)} out of␣
 ↪{len(df_exog_interpolated)} total')
```

```
c:\Users\rtbra\OneDrive\Documents\WGU\D610\executive_order_prediction\.venv\Lib\
site-packages\pandas\core\missing.py:604: UserWarning:


The maximal number of iterations maxit (set to 20 by the program)
allowed for finding a smoothing spline with fp=s has been reached: s
too small.
There is an approximation returned but the corresponding weighted sum
of squared residuals does not satisfy the condition abs(fp-s)/s < tol.


0 rows with null values found: 0 out of 1070 total
```

1.4 Principal Component Analysis

1.4.1 Standardization of exogenous variables

[12]:
```
scaler = StandardScaler()
df_to_standardize = df_exog_interpolated.drop(['disapproving'], axis=1)
df_standardized = scaler.fit_transform(df_to_standardize)

# create matrix of all principal components

pca = PCA()
pca.fit(df_standardized)

print(pca.components_)
```

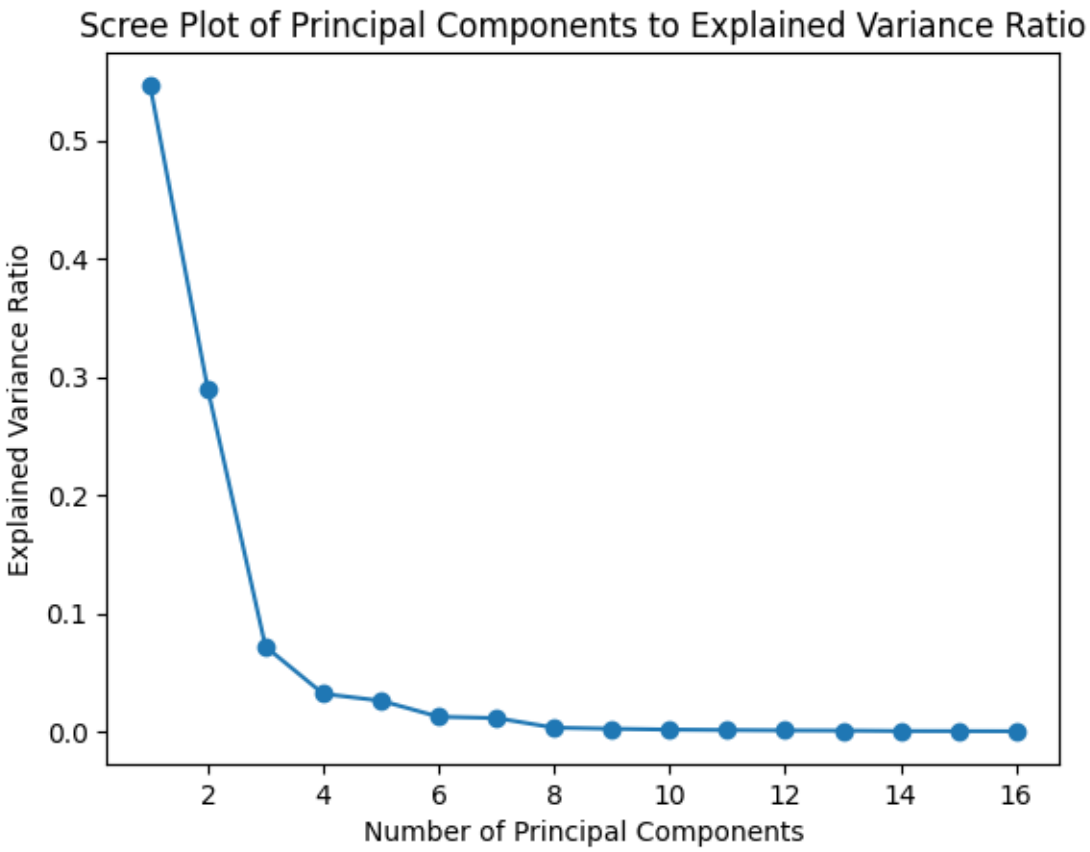
```
[[-1.13977988e-01 -1.96907127e-01  3.04171609e-01  3.26251029e-01
   3.00935542e-01  2.64852728e-01  2.99036565e-01  2.84493117e-01
   1.53551675e-01  9.17831260e-02  1.86262429e-01  1.60883541e-01
   1.74190836e-01  3.15797918e-01  3.20240115e-01  3.04965838e-01]
 [-7.99896334e-02 -2.50282028e-01 -5.83337368e-02  1.78150719e-02
  -6.56578238e-02 -2.73938111e-01 -2.07773556e-01 -1.70135064e-01
   3.28171312e-01  4.28004529e-01  3.60704822e-01  3.94084303e-01
   3.73578393e-01 -1.04931343e-01 -1.26427411e-01 -1.81442961e-01]
 [ 8.29643632e-01 -6.78048692e-02  1.77038740e-01  6.17763669e-02
  -2.06796835e-01  1.48935325e-02  4.99723710e-02 -2.03512558e-01
  -2.93662916e-01  1.46758728e-01  1.62804146e-01 -3.24177399e-02
   1.42841242e-01  6.18042117e-02  1.18456997e-01  1.06985926e-01]
 [ 1.07243094e-01  5.65760538e-01  2.35838105e-01  2.97986707e-01
  -2.66335131e-01 -1.29134658e-01  2.86563645e-02 -1.76685437e-01
   5.31072664e-01 -3.38562435e-02 -1.75201218e-01  2.03418417e-01
  -1.81892134e-01  4.80499249e-02  1.07073833e-01  4.72425903e-02]
 [ 1.76927104e-01  6.45017728e-01 -4.53625788e-02 -1.56731117e-01
   4.03036885e-01  3.25490771e-02 -6.63644307e-02  4.07972910e-01
  -7.38493368e-02  1.20947260e-01  2.36488650e-01  3.63682355e-02
   2.49826723e-01 -1.88138817e-01 -3.01627075e-02 -1.25025310e-01]
 [ 4.85590544e-01 -3.49269618e-01 -2.05232323e-01  1.09619001e-02
   3.17820669e-01 -9.00078544e-02 -6.81698700e-02  2.89323443e-01
   4.59738556e-01 -3.14016020e-01 -1.99071762e-01  9.68389096e-02
  -1.35056606e-01 -9.52445140e-02 -7.09468749e-02 -1.05770062e-01]
 [-6.07095992e-02 -1.69047627e-01  7.35815268e-01 -4.04711961e-02
   9.20401040e-02 -2.27796915e-01 -5.24637105e-04  9.64628907e-02
  -9.19864629e-04  5.77009197e-02 -6.45047011e-02 -2.28919569e-01
  -8.02744946e-02 -4.98601399e-01  7.99936390e-02 -1.83481439e-01]
 [ 5.16034799e-02 -8.26279717e-03  9.06094231e-02 -2.75214703e-02
   3.26928183e-01 -9.20037329e-03 -1.88227807e-01 -3.21066862e-02
  -2.38790978e-01  5.26701385e-01 -4.03476485e-01  3.41486624e-01
  -4.03796128e-01  2.40057908e-01 -4.24137703e-02 -9.24887548e-02]
 [ 5.31617221e-02 -7.02627719e-02 -1.18256635e-01 -1.47527066e-01
  -4.15318248e-01  5.55845167e-01  1.38403190e-01  3.14755557e-01
   2.25016557e-01  4.28531294e-01 -3.19924565e-02 -5.90826430e-02
  -1.91099984e-01 -2.77827600e-01  1.61791241e-02 -6.36676288e-02]
 [-3.66476900e-02 -2.66076691e-02  2.05809649e-01 -1.14351406e-01
  -1.83567769e-01  3.39669848e-01 -2.10241638e-01  4.63135741e-02
  -2.30314675e-01 -4.29214326e-01 -7.50722041e-02  6.64164890e-01
   1.24572174e-01 -1.59558823e-01  8.65190313e-02 -1.20882118e-01]
 [-2.55563561e-02 -9.39041875e-03 -2.30887707e-01  1.11654310e-01
  -1.32013982e-01 -4.54972343e-01  5.15720070e-01  2.31158048e-01
  -2.61932443e-01  3.67549306e-02 -9.65358672e-03  3.54632276e-01
  -1.60258260e-01 -3.30559289e-01  3.04930287e-02  2.40739209e-01]
 [ 7.37856408e-03 -5.37649940e-03  8.17142840e-02 -9.98809046e-02
   2.48907275e-02 -2.26662904e-04 -1.07726235e-01 -6.55105741e-04
   1.11560223e-02 -1.30758690e-01  7.07560814e-01  8.13290773e-02
  -6.60584868e-01  8.93075199e-02 -4.70178952e-02 -6.00011690e-04]
```

```
[-4.07519766e-02 -2.66733339e-02 -2.23676121e-01  6.73806005e-01
 -1.08405213e-01 -3.45501710e-02 -3.98338156e-01  2.10011108e-01
 -2.04449760e-01  2.13314711e-02  5.53151790e-02 -1.30287433e-01
 -7.27434134e-02 -1.08468578e-01  3.47826062e-01 -2.78441353e-01]
[ 3.28248912e-03 -2.46647742e-02  2.11997301e-01 -5.89645191e-02
 -3.98832899e-01 -2.74332761e-01 -2.50765666e-01  5.90887918e-01
 -4.45144615e-02 -6.65058352e-03 -4.93360523e-02 -4.37622361e-02
  8.13703619e-02  4.03547564e-01 -3.35028338e-01  1.28500586e-01]
[-2.83258591e-02 -1.12168537e-02 -1.07348584e-01 -4.18656149e-01
 -3.62642012e-02 -1.85352050e-01 -3.78977533e-01  3.93773008e-02
  1.00766399e-01  6.05106383e-02 -5.03379078e-02 -2.58608918e-02
 -1.20169490e-02 -6.63055726e-02  6.39624991e-01  4.48052519e-01]
[-6.66207524e-04  3.73430230e-03  2.18186116e-02  2.88968266e-01
  1.27504174e-01  1.86090110e-01 -3.36894208e-01 -8.68475256e-02
 -1.63914124e-02  3.53836118e-02 -8.03326065e-03 -1.11092480e-02
 -1.75990673e-02 -3.65415762e-01 -4.34268873e-01  6.48030377e-01]]
```

1.4.2 Using a scree plot and cumulative variance ratio of explained variance ratio to select principal components for use in SARIMAX as exogenous variables:

The scree plot elbow at 4 principal components, and the cumulative explained variance of the first four principal components at ~94%, reveal that four principal components of the exogenous variables are sufficient for use in the SARIMAX model.

[13]:
```
# Calculate explained variance ratio
explained_variance_ratio = pca.explained_variance_ratio_

# Scree plot using explained variance ratio
plt.plot(range(1,
               len(pca.explained_variance_ratio_) + 1),
         pca.explained_variance_ratio_,
         'o-')
plt.xlabel('Number of Principal Components')
plt.ylabel('Explained Variance Ratio')
plt.title('Scree Plot of Principal Components to Explained Variance Ratio')
plt.show()

print(pca.explained_variance_ratio_.cumsum())
```




```
[0.54660966 0.83642271 0.90816453 0.93994265 0.96590585 0.97831245
 0.98958994 0.99274269 0.99497806 0.9964964  0.99766848 0.99853465
 0.99923728 0.99957836 0.99984189 1.        ]
```