JGP DS test

July 22, 2024

JGP Macro Data Scientist Internship Selection Process Test

Author: Pedro Silva dos Santos

1 Question 1

Obtain the U.S. inflation series (CPI) from the BLS (Bureau of Labor Statistics) — https://www.bls.gov/ — via API, process the data, and format it so that the columns correspond to the series and each row represents an observation over time. Save the result in a CSV file. Specifically, obtain the following nationallevel monthly series: - CPI All items, seasonally adjusted - CPI All items, less food and energy, seasonally adjusted - CPI Gasoline (all types), seasonally adjusted

1.0.1 Solution:

[1]:
```
import requests
import pandas as pd
import json
```

[2]:
```
# Define the series IDs for the data I want to retrieve
# Series ID as specified in Data Tools > Help & Tutorials at bls.gov
series_ids = {
    'CPI All items, seasonally adjusted': 'CUSR0000SA0',
    'CPI All items, less food and energy, seasonally adjusted':␣
 ↪'CUSR0000SA0L1E',
    'CPI Gasoline (all types), seasonally adjusted': 'CUSR0000SETB01'
}

# Define the payload for the API request
payload = json.dumps({
    "seriesid": list(series_ids.values()),
    "startyear": "2003",
    "endyear": "2023",
    "registrationkey": '263823f3339346b7978c67e2b0c5d8d4'
})

# Define the URL for the BLS API request
url = 'https://api.bls.gov/publicAPI/v2/timeseries/data/'
```

```
# Make the API request
response = requests.post(url, headers={'Content-Type': 'application/json'},␣
 ↪data=payload)

# Check if the request was successful
if response.status_code == 200:
    data = response.json()

    # Initialize an empty DataFrame
    df = pd.DataFrame()

    # Process each series
    for series in data['Results']['series']:
        series_id = series['seriesID']
        series_name = [key for key, value in series_ids.items() if value ==␣
 ↪series_id][0]
        series_data = series['data']

        # Create a DataFrame for the current series
        series_df = pd.DataFrame(series_data)
        series_df['date'] = pd.to_datetime(series_df['year'] + '-' +␣
 ↪series_df['period'].str.replace('M', '') + '-01')
        series_df = series_df[['date', 'value']]
        series_df.set_index('date', inplace=True)
        series_df.columns = [series_name]

        # Merge the current series DataFrame with the main DataFrame
        if df.empty:
            df = series_df
        else:
            df = df.merge(series_df, on='date', how='outer')

    # Save the DataFrame to a CSV file
    df.to_csv('bls_cpi_data.csv')
else:
    print(f"Failed to retrieve data: {response.status_code}")

print("Data retrieval and processing complete. The data has been saved to␣
 ↪'bls_cpi_data.csv'.")
```

```
Data retrieval and processing complete. The data has been saved to
'bls_cpi_data.csv'.
```

2 Question 2

Using Plotly in Python, develop a chart displaying the All items, less food and energy, seasonally adjusted price series with year-over-year percentage variation using the monthly data from 2019 to the present. Keep the frequency monthly in the chart.

2.0.1 Solution:

[3]:
```
# Reading the previously fetched data
df = pd.read_csv('bls_cpi_data.csv')
df.tail()
```

[3]:

	date	CPI All items, seasonally adjusted	CPI All items, less food and energy, seasonally adjusted	CPI Gasoline (all types), seasonally adjusted
235	2022-08-01	295.209	296.569	339.037
236	2022-09-01	296.341	298.284	322.713
237	2022-10-01	297.863	299.351	335.141
238	2022-11-01	298.648	300.292	333.999
239	2022-12-01	298.812	301.423	308.222

Calculating year-over-year percentage variation using the monthly data from 2019:

[4]:
```
# Calculate the year-over-year percentage variation
df['yoy_pct_variation'] = (df['CPI All items, less food and energy, seasonally␣
 ↪adjusted'].pct_change(periods=12) * 100)

# Filter the data from 2019 to the present
df['date'] = pd.to_datetime(df['date'])
df_filtered = df[df['date'] >= '2019-01-01']

# Remove other data
df_filtered = df_filtered.drop(columns=['CPI All items, seasonally adjusted',␣
 ↪'CPI Gasoline (all types), seasonally adjusted'])
df_filtered.head()
```

[4]:
```
           date  CPI All items, less food and energy, seasonally adjusted  \
192 2019-01-01                                                    260.766
193 2019-02-01                                                    261.186
194 2019-03-01                                                    261.567
195 2019-04-01                                                    261.997
196 2019-05-01                                                    262.217

     yoy_pct_variation
192           2.179433
193           2.141089
194           2.066562
195           2.091338
196           1.972428
```

Plotting the data with plotly:

[5]:
```
import plotly.express as px

# Plot the data using Plotly
fig = px.line(df_filtered, x=df_filtered['date'], y='yoy_pct_variation',␣
 ↪title='CPI All items, less food and energy, seasonally adjusted -␣
 ↪Year-over-Year Percentage Variation')

fig.update_layout(
    xaxis_title='Date',
    yaxis_title='Year-over-Year Variation (%)'
)

fig.show()
```

3 Question 3

Describe in words how you would automate the process of extracting the data.

3.0.1 Solution:

I would develop and deploy a data pipeline using Meltano and Apache Airflow.

- Meltano Setup: I would define and configure Meltano to extract data from the BLS API and load it into the desired storage (csv for example).
- Airflow Scheduling: I would create an Airflow DAG to schedule the Meltano extraction process and set the frequency (monthly for this case) based on how often we need to retrieve data.

Airflow will automatically trigger the Meltano extraction process according to the defined schedule and we can also use Airflow to monitor the execution, review logs, and handle any issues that arise.

By combining a data movement tool (Meltano) and a data scheduler tool (Airflow), we can automate the process of data extraction (ETL) with scheduling and monitoring capabilities.

4 Question 4

Explain how you would relate the price series (All items) with the Gasoline (Gasoline) price series. BONUS: Implement this solution.

4.0.1 Solution:

Plotting a graph to visually inspect how the gasoline prices follow or deviate from the overall CPI.

[6]:
```
import plotly.express as px

fig = px.line(df, x=df['date'], y=['CPI All items, seasonally adjusted', 'CPI␣
 ↪Gasoline (all types), seasonally adjusted'],
              labels={'value': 'Index Value', 'date': 'Date'},
              title='Comparison of CPI All Items and Gasoline Prices')
fig.update_layout(yaxis_title='Index Value')
fig.show()
```

[7]:
```
correlation = df['CPI All items, seasonally adjusted'].corr(df['CPI Gasoline␣
 ↪(all types), seasonally adjusted'])
print(f"Correlation coefficient: {correlation}")
```

```
Correlation coefficient: 0.5160930109950101
```

The overall Pearson correlation indicates the variables are moderately correlated.

Let's peform a regression analysis:

[8]:
```
import statsmodels.api as sm

# Prepare the data
X = df['CPI All items, seasonally adjusted']
y = df['CPI Gasoline (all types), seasonally adjusted']

# Add a constant to the independent variable matrix
X = sm.add_constant(X)

# Fit the regression model
model = sm.OLS(y, X).fit()

# Print the summary
print(model.summary())
```

```
                                            OLS Regression Results
==============================================================================
==========================
Dep. Variable:     CPI Gasoline (all types), seasonally adjusted   R-squared:
0.266
Model:                                                       OLS   Adj.
R-squared:                  0.263
```

```
Method:                                          Least Squares   F-statistic:
86.41
Date:                                         Mon, 22 Jul 2024   Prob
(F-statistic):           9.74e-18
Time:                                                23:50:04   Log-
Likelihood:                  -1273.5
No. Observations:                                         240   AIC:
2551.
Df Residuals:                                             238   BIC:
2558.
Df Model:                                                   1
Covariance Type:                                    nonrobust
================================================================================
======================
                                       coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
----------------------
const                                -8.5532     26.710     -0.320      0.749
-61.171      44.064
CPI All items, seasonally adjusted    1.0718      0.115      9.295      0.000
0.845       1.299
==============================================================================
Omnibus:                       43.948   Durbin-Watson:                   0.075
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               12.449
Skew:                           0.254   Prob(JB):                      0.00198
Kurtosis:                       2.006   Cond. No.                     1.96e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.96e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

- R-squared: 0.266
- Adj. R-squared: 0.263

This indicates that approximately 26.6% of the variance in the gasoline CPI can be explained by the CPI All items. This suggests a relatively weak relationship.

Adjusted R-squared is slightly lower than R-squared, accounting for the number of predictors. It is still relatively low, indicating that the model might not be a good fit.

- F-statistic: 86.41

This statistic tests whether the independent variable significantly contributes to explaining the dependent variable. A high F-statistic with a very low p-value (Prob (F-statistic): 9.74e-18) indicates that the model is statistically significant.

- Prob (F-statistic): 9.74e-18

This p-value is extremely low, suggesting that the overall regression model is statistically significant.

- CPI All items, seasonally adjusted: 1.0718

The coefficient indicates that for each one-unit increase in the CPI All items, the CPI for gasoline is expected to increase by approximately 1.072 units. This coefficient is statistically significant (p-value: 0.000), showing a strong relationship between the two series.

The model is statistically significant overall and the CPI All items coefficient is significant and suggests a positive relationship with gasoline prices. But the explanatory power (R-squared) is relatively low and there are indications of non-normal residuals, strong autocorrelation, and a large condition number, which could affect the reliability of the model.

Additionally we could do time series decomposition to understand trends and seasonal patterns and perform a Granger causality test to determine if one series can help predict the other.