代码解读从1109工作日志往后接着写

第一部分：加载行数据

- 示例中从雅虎财经（Yahoo Finance)获取加拿大消费者物价指数的数据，我们这边无法访问，代码示例无法运行，但先尝试解读了解方法

```
'''
第一部分：加载行数据
'''
# 加载Canadian Consumer Price Index数据
# Load the Canadian Consumer Price Index data frm Statistics Canada
cpi_df=load_stats_can_data('10100106')
col='Consumer Price Index (CPI) inversely weighted by volatility and is adjusted to exclude the effect of changes in indirect taxes (CPIW) (year-over-year percent change)'
cpi_df=cpi_df[~cpi_df.index.duplicated(keep='first')][col]
# 从雅虎财经下载S&P/TSX综合指数历史每日数据,从1970年至今
# load the S&P/TSX Composite index historical daily data from Yahoo Finance
tsx=pd.read_csv('https://query1.finance.yahoo.com/v7/finance/download/%5EGSPTSE?'+\
                'period1=-315619200&period2={}&interval=1d&events=history'.format(
                (pd.Timestamp.today()-pd.Timestamp("1970-01-01"))//pd.Timedelta('1s')),
               parse_dates=['Date'],index_col=0)
```

第二部分：导出和处理数据

1. 填充空数据
   TSX数据只存在于非假日工作日，但月数据仍具有可比性,假设这些天的经济状况没有变化，那么我们可以在连续的每日日期范围内填充缺失的数据来解决这个问题.

首先创建一个空的数据框架，包含可用TSX数据的开始日期和结束日期之间的所有日期

```
'''
第二部分：导出和处理数据
'''
# the TSX data exists only for non-holiday weekdays but to keep all months' data comparable
# let's assume that there is no change in economic conditions on those days, so we
```

```
can forward fill missing data over a continuous daily date range to address this
issue by first creating an empty dataframe of all days in between the start and end
dates of the available TSX data
continuous_daterange=pd.date_range(tsx.index[0],tsx.index[-1]).to_frame().drop(0,ax
is=1)
# then concatenate this continuous date range with the TSX data and do a forward
fill
tsx=pd.concat([continuous_daterange,tsx],axis=1).ffill()
del continuous_daterange # delete the temporary continuous_daterange index
```

第三部分：探讨多项式加权方法

三种可能使用的the baseline weights for nlag observations多项式加权方法
还没看懂

第四部分:拟合MIDAS的回归模型

- 示例中筛选了至2019年截至的数据，进行模型的拟合

```
# Select training data up to end of 2019
x1 = tsx['Adj Close'].rename('TSX_7d_pct_change')['1983-11-01':'2019-12-
01'].pct_change(7).dropna()
x2 = tsx['Adj Close'].rename('TSX_14d_pct_change')['1983-11-01':'2019-12-
01'].pct_change(14).dropna()
x3 = tsx['Adj Close'].rename('TSX_21d_pct_change')['1983-11-01':'2019-12-
01'].pct_change(21).dropna()
x4 = tsx['Adj Close'].rename('TSX_28d_pct_change')['1983-11-01':'2019-12-
01'].pct_change(28).dropna()
y = cpi_df.rename('CPI_YoY')['1984-02-01':'2019-12-01']
model = MIDASRegressor(endog=y,exog=pd.concat([x1,x2,x3,x4],axis=1).loc['1983-11-
30':],
                       xlag=30,ylag=1,poly='beta')
fit = model.fit()
```

第五部分：评估模型的拟合

``

- 绘制预测与实际的拟合曲线图形：

```
pd.concat([fit.orig_endog,fit.predict()], axis=1)[-24:].plot()
```

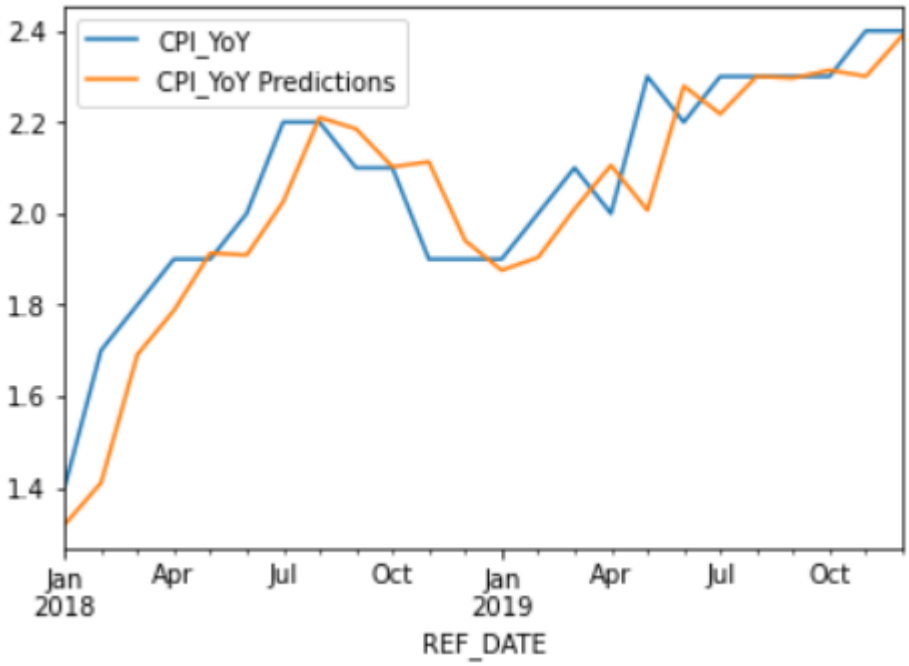
- 拟合分数：

```
fit.score()
```

- 一些指标：

```
fit.significance()
```

示例中给出的结果图，拟合程度看起来相当高：



第六部分：样本外预测

- 选取2020年到今天的数据，进行预测

```
# Select test data (2020-present)
x1_test = tsx['Adj Close'].rename('TSX_7d_pct_change')['2019-12-
01':].pct_change(7).dropna()
x2_test = tsx['Adj Close'].rename('TSX_14d_pct_change')['2019-12-
01':].pct_change(14).dropna()
x3_test = tsx['Adj Close'].rename('TSX_21d_pct_change')['2019-12-
01':].pct_change(21).dropna()
x4_test = tsx['Adj Close'].rename('TSX_28d_pct_change')['2019-12-
```

```
01':].pct_change(28).dropna()
y_test = cpi_df.rename('CPI_YoY')['2020-01-01':]
```

- 预测

```
# Plot predictions
preds =
fit.predict(endog=y_test,exog=pd.concat([x1_test,x2_test,x3_test,x4_test],axis=1))
preds.index = y_test.index[1:]
pd.concat([y_test[1:], preds], axis=1).plot()# 绘制比较图
```

示例中给出的样本外预测比较图：