Time Series Forecasting Using Statistical and Machine Learning Models

Jeffrey Yau
Chief Data Scientist, AllianceBernstein, L.P.
Lecturer, UC Berkeley Masters of Information Data Science

Forecasting: Problem Formulation

- Forecasting: predicting the **future values** of the series using **current information set**

- **Current information set** consists of current and past values of the series of interest and perhaps other “exogenous” series

Time Series Forecasting Requires Models

$$\hat{y}_{t+H|t} = f(\text{current information set})$$
$$= f(y_t, y_{t-1}, y_{t-2}, \ldots, y_1, \mathbf{X}_t, \mathbf{X}_{t-1}, \ldots, \mathbf{X}_1)$$

Forecast horizon: H

A statistical model or a machine learning algorithm

Information Set:

$$\Omega_t = \{y_t, y_{t-1}, y_{t-2}, \ldots, y_1, \mathbf{X}_t, \mathbf{X}_{t-1}, \ldots, \mathbf{X}_1\}$$

[redacted]

Autoregressive Moving Average Model (ARMA)

$$\boxed{\hat{y}_{t+H|t} = f(\text{current information set})}$$

$$y_t = \mu + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \omega_t + \theta_1 \omega_{t-1} + \theta_2 \omega_{t-2} + \cdots + +\theta_q \omega_{t-q}$$

More common notations:

$$(1 - \phi_1 L - \phi_2 L^2 - \cdots - \phi_p L^p)\tilde{y}_t = (1 - \theta_1 L - \theta_2 L^2 - \cdots - \theta_q L^q)\omega_t$$

$$\tilde{y}_t = y_t - \mu$$

$$\phi(L)\tilde{y}_t = \theta(L)\omega_t$$

[redacted]

ARIMA

A time series y_t follows an **ARIMA(p,d,q)** process if the dth differences of the y_t series is an **ARMA(p,q)** process

$$\phi_p(L)(1-L)^d \tilde{y}_t = \theta_q(L)\omega_t$$

From statsmodels' documentation:

The SARIMA model is specified $(p, d, q) \times (P, D, Q)_s$

$$\phi_p(L)\tilde{\phi}_P(L^s)\Delta^d\Delta_s^D y_t = A(t) + \theta_q(L)\tilde{\theta}_Q(L^s)\zeta_t$$

https://www.statsmodels.org/dev/generated/statsmodels.tsa.statespace.sarimax.SARIMAX.html

General Steps to Build SARIMA Model

1. Ingest the series
2. Train/validation/test split the series
3. Conduct exploratory time series data analysis on the training set
4. Determine if the series are stationary
5. Transform the series
6. Build a model on the transformed series
7. Model diagnostic
8. Model selection (based on some pre-defined criterion)
9. Conduct forecast using the final, chosen model
10. Inverse-transform the forecast
11. Conduct forecast evaluation

Iterative

[redacted]

Example: National Liquor Sales

United States Census Bureau

Topics — Population, Economy | Geography — Maps, Products | Library — Infographics, Publications | Data — Tools, Developers | Surveys/Programs — Respond, Survey Data | Newsroom — News, Blogs | About Us — Our Research

You are here: Census.gov › Business & Industry › Time Series / Trend Charts

Business and Industry

Main | About the Surveys | Data by Geography | Data by Survey | FAQs

In This Section:

- Technical Document [PDF, 825KB]
- Chart Instructions [PDF, 658KB]
- Download Data Sets New

Data by Sector:

- Economy-Wide
- Construction
- Governments
- International Trade
- Manufacturing
- Retail Trade
- Services
- Wholesale Trade
- Other Sectors

Contact Us:

Email

TIME SERIES / TREND CHARTS

Please follow the numbers in order.

1 Select the report/survey from which you wish to retrieve data: Monthly Retail Trade and Food Services

2 Select a date range: Start: 1992 End: 2017

3 Select Industry or Category: 4453: Beer, Wine, and Liquor Stores

4 Select one Item : Sales - Monthly

5 Select Geographical Level: U.S. Total

Select as available:

- ☐ Seasonally Adjusted
- ☑ Not Seasonally Adjusted
- ☐ Show Estimates of Sampling Variability

GET DATA

Download all data for this report/survey New

Save this search

[redacted]

Time Series EDA

[redacted]

Examine Autocorrelation of the Series

Time Series EDA

```
def tsplot2(y, title, lags=None, figsize=(12, 8)):
    '''Examine the patterns of ACF and PACF, along with the time series plot and histogram.
    '''
    fig = plt.figure(figsize=figsize)
    layout = (2, 2)
    ts_ax   = plt.subplot2grid(layout, (0, 0))
    hist_ax = plt.subplot2grid(layout, (0, 1))
    acf_ax  = plt.subplot2grid(layout, (1, 0))
    pacf_ax = plt.subplot2grid(layout, (1, 1))

    y.plot(ax=ts_ax)
    ts_ax.set_title(title, fontsize=14, fontweight='bold')
    y.plot(ax=hist_ax, kind='hist', bins=25)
    hist_ax.set_title('Histogram')
    smt.graphics.plot_acf(y, lags=lags, ax=acf_ax)
    smt.graphics.plot_pacf(y, lags=lags, ax=pacf_ax)
    [ax.set_xlim(0) for ax in [acf_ax, pacf_ax]]
    sns.despine()
    plt.tight_layout()
    return ts_ax, acf_ax, pacf_ax
```

Time Series EDA

Series Transformation

Transformation

Applying **first differencing** or **seasonal differencing** to the **log of the series** should make the-above two series stationary:

$$log(y_t) - log(y_{t-l}) = log(\frac{y_t}{y_{t-l}})$$

where l is some lag.

```
liquor_train_transformed = np.log(liquor_train).diff(12)
```

1992-01-01	1509	nan
1992-02-01	1541	nan
1992-03-01	1597	nan
1992-04-01	1675	nan
1992-05-01	1822	nan
1992-06-01	1775	nan
1992-07-01	1912	nan
1992-08-01	1862	nan
1992-09-01	1770	nan
1992-10-01	1882	nan
1992-11-01	1831	nan
1992-12-01	2511	nan
1993-01-01	1614	0.06727
1993-02-01	1529	-0.00782
1993-03-01	1678	0.04948

Time Series EDA (After Transformation)

[redacted]

Model Estimation

```
mod = sm.tsa.statespace.SARIMAX(liquor_train, order=(3,1,3), seasonal_order=(0,1,0,12))
sarima_fit2 = mod.fit()
print(sarima_fit2.summary())
```

```
                                 Statespace Model Results
==========================================================================================
Dep. Variable:                                 Value   No. Observations:                  235
Model:             SARIMAX(3, 1, 3)x(0, 1, 0, 12)   Log Likelihood               -1248.761
Date:                            Wed, 17 Oct 2018   AIC                           2511.521
Time:                                    11:15:59   BIC                           2535.738
Sample:                                01-01-1992   HQIC                          2521.285
                                     - 07-01-2011
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -1.2758      0.150     -8.496      0.000      -1.570      -0.981
ar.L2         -1.1413      0.172     -6.633      0.000      -1.479      -0.804
ar.L3         -0.1280      0.147     -0.871      0.384      -0.416       0.160
ma.L1          0.5647      0.138      4.091      0.000       0.294       0.835
ma.L2          0.4267      0.147      2.910      0.004       0.139       0.714
ma.L3         -0.5176      0.144     -3.602      0.000      -0.799      -0.236
sigma2      4772.3432    603.667      7.906      0.000    3589.178    5955.509
===================================================================================
Ljung-Box (Q):                      101.72   Jarque-Bera (JB):                 2.06
Prob(Q):                              0.00   Prob(JB):                         0.36
Heteroskedasticity (H):               1.41   Skew:                             0.11
Prob(H) (two-sided):                  0.14   Kurtosis:                         3.41
===================================================================================
```

Model Diagnostic

```
sarima_fit2.plot_diagnostics(figsize=(14, 8))
```

[redacted]

Forecasting

Multivariate Time Series Modeling

$$
\begin{aligned}
y_{1,t} &= f_1(y_{1,t-1}, \ldots, y_{K,t-1} \ldots y_{1,t-p}, \ldots, y_{K,t-p} \ldots \mathbf{X_{t-1}}, \mathbf{X_{t-2}} \ldots) \\
&\vdots \\
y_{K,t} &= f_K(y_{1,t-1}, \ldots, y_{K,t-1} \ldots y_{1,t-p}, \ldots, y_{K,t-p} \ldots \mathbf{X_{t-1}}, \mathbf{X_{t-2}} \ldots)
\end{aligned}
$$

[redacted]

Joint Modeling of Multiple Time Series

[redacted]

Vector Autoregressive (VAR) Models

- a system of linear equations of the K series being modeled
- only applies to stationary series
- non-stationary series can be transformed into stationary ones using simple differencing (note: if the series are not co-integrated, then we can still apply VAR ("VAR in differences"))

Vector Autoregressive (VAR) Model of Order 1

$$
\begin{aligned}
y_{1,t} &= c_1 + \phi_{11} y_{1,t-1} + \phi_{K2,1} y_{2,t-1} + \cdots + \phi_{1K} y_{K,t-1} + u_{1,t} \\
y_{2,t} &= c_2 + \phi_{21} y_{1,t-1} + \phi_{K2,1} y_{2,t-1} + \cdots + \phi_{2K} y_{K,t-1} + u_{2,t} \\
&\vdots \qquad\qquad\qquad\qquad \vdots \\
y_{K,t} &= c_K + \phi_{K1} y_{1,t-1} + \phi_{K2,1} y_{2,t-1} + \cdots + \phi_{KK} y_{K,t-1} + u_{K,t}
\end{aligned}
$$

General Steps to Build VAR Model

1. Ingest the series
2. Train/validation/test split the series
3. Conduct exploratory time series data analysis on the training set
4. Determine if the series are stationary
5. Transform the series
6. Build a model on the transformed series
7. Model diagnostic
8. Model selection (based on some pre-defined criterion)
9. Conduct forecast using the final, chosen model
10. Inverse-transform the forecast
11. Conduct forecast evaluation

Iterative

[redacted]

Index of Consumer Sentiment

[redacted]

Series Transformation

Transformation

Applying **first differencing** or **seasonal differencing** to the **log of the series** should make the-above two series stationary:

$$log(y_t) - log(y_{t-l}) = log(\frac{y_t}{y_{t-l}})$$

where l is some lag.

- UMCSENT series: $l = 1$
- beer series: $l = 12$

[redacted]

Transforming the Series

Take the simple-difference of the natural logarithmic transformation of the series

```
series_transformed['UMCSENT'] = np.log(series.iloc[:,0]).diff(1)
series_transformed['beer']    = np.log(series.iloc[:,1]).diff(12)
```

	UMCSENT	beer	UMCSENT	beer
DATE				
1992-01-01	67.5	1509.0	NaN	NaN
1992-02-01	68.8	1541.0	0.019076	NaN
1992-03-01	76.0	1597.0	0.099530	NaN
1992-04-01	77.2	1675.0	0.015666	NaN
1992-05-01	79.2	1822.0	0.025577	NaN
1992-06-01	80.4	1775.0	0.015038	NaN
1992-07-01	76.6	1912.0	-0.048417	NaN
1992-08-01	76.1	1862.0	-0.006549	NaN
1992-09-01	75.6	1770.0	-0.006592	NaN
1992-10-01	73.3	1882.0	-0.030896	NaN
1992-11-01	85.3	1831.0	0.151614	NaN
1992-12-01	91.0	2511.0	0.064685	NaN
1993-01-01	89.3	1614.0	-0.018858	0.067268
1993-02-01	86.6	1529.0	-0.030702	-0.007818
1993-03-01	85.9	1678.0	-0.008116	0.049476

note: difference-transformation generates missing values

Transformed Series

Consumer Sentiment

Beer Consumption

[redacted]

VAR Model Estimation and Output

```
model = sm.tsa.VARMAX(y_train, order=(3,0), trend='c')
model_result = model.fit(maxiter=1000, disp=False)
print(model_result.summary())
```

```
                              Statespace Model Results
==============================================================================
Dep. Variable:     ['UMCSENT', 'beer']   No. Observations:                  294
Model:                          VAR(3)   Log Likelihood                1124.934
                           + intercept   AIC                          -2215.867
Date:                  Thu, 30 Nov 2017   BIC                          -2153.246
Time:                          11:25:18   HQIC                         -2190.790
Sample:                      01-01-1993
                           - 06-01-2017
Covariance Type:                    opg
===================================================================================
Ljung-Box (Q):                52.72, 181.87   Jarque-Bera (JB):          21.96, 1.78
Prob(Q):                       0.09, 0.00     Prob(JB):                   0.00, 0.41
Heteroskedasticity (H):        2.26, 0.62     Skew:                      -0.38, -0.19
Prob(H) (two-sided):           0.00, 0.02     Kurtosis:                   4.11, 3.03
```

VAR Model Output - Estimated Coefficients

Results for equation UMCSENT

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0060	0.005	1.266	0.206	-0.003	0.015
L1.UMCSENT	-0.0551	0.051	-1.089	0.276	-0.154	0.044
L1.beer	-0.1338	0.105	-1.274	0.203	-0.340	0.072
L2.UMCSENT	-0.1654	0.060	-2.774	0.006	-0.282	-0.049
L2.beer	0.0174	0.096	0.182	0.856	-0.171	0.205
L3.UMCSENT	-0.1218	0.054	-2.247	0.025	-0.228	-0.016
L3.beer	-0.0398	0.089	-0.446	0.656	-0.215	0.135

Results for equation beer

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0097	0.003	3.377	0.001	0.004	0.015
L1.UMCSENT	0.0559	0.041	1.375	0.169	-0.024	0.136
L1.beer	0.1060	0.055	1.920	0.055	-0.002	0.214
L2.UMCSENT	0.0292	0.038	0.764	0.445	-0.046	0.104
L2.beer	0.2616	0.056	4.674	0.000	0.152	0.371
L3.UMCSENT	0.0254	0.036	0.711	0.477	-0.045	0.096
L3.beer	0.3698	0.057	6.507	0.000	0.258	0.481

VAR Model Output - Var-Covar Matrix

Error covariance matrix

	coef	std err	z	P>|z|	[0.025	0.975]
sqrt.var.UMCSENT	0.0462	0.002	27.616	0.000	0.043	0.049
sqrt.cov.UMCSENT.beer	0.0004	0.002	0.211	0.833	-0.003	0.004
sqrt.var.beer	0.0276	0.001	23.581	0.000	0.025	0.030

VAR Model Diagnostic

```
model = sm.tsa.VARMAX(y_train, order=(3,0), trend='c')
model_result = model.fit(maxiter=1000, disp=False)
model_result.plot_diagnostics()
```

UMCSE

Bee

VAR Model Selection

Model selection, in the case of VAR(p), is the choice of the order and the specification of each equation

Information criterion can be used for model selection:

```
aic = []
for i in range(5):
    i += 1
    model = sm.tsa.VARMAX(y_train, order=(i,0), trend='c')
    model_result = model.fit(maxiter=1000, disp=False)
    print('Order =', i)
    print('AIC: ', model_result.aic)
    print('BIC: ', model_result.bic)
    print('HQIC: ', model_result.hqic)
```

VAR Model - Inverse Transform

Don't forget to inverse-transform the forecasted series!

This is equivalent to $log(y_t) - log(y_{t-12}) = log(\frac{y_t}{y_{t-12}})$

Define $z = log(y_t) - log(y_{t-12})$

Then,

$$\boxed{\begin{aligned} log(y_t) &= z + log(y_{t-12}) \\ y_t &= e^{z+log(y_{t-12})} \\ &= e^z(y_{t-12}) \end{aligned}}$$

So, we have forecast

$$\boxed{y_{T+s} = e^z(y_{(t-12)+s})}$$

where $s > 1$

[redacted]

VAR Model Forecast

$$RMSE = \sqrt{\frac{1}{L}\sum_{l=1}^{L}\left(y_{T+l} - \hat{y}_{T+l}\right)^2}$$

where T is the last observation period and l is the lag

```
from math import sqrt
from sklearn.metrics import mean_squared_error

VAR_forecast_beer = np.exp(z['beer'])*series['beer'][-3:]
VAR_forecast_UMCSENT = np.exp(z['UMCSENT'])*series['UMCSENT'][-3:]

rmse_beer = sqrt(mean_squared_error(series['beer'][-3:], VAR_forecast_beer))
rmse_UMCSENT = sqrt(mean_squared_error(series['UMCSENT'][-3:], VAR_forecast_UMCSENT))
```

```
UMSCENT - Test RMSE: 0.210
Beer    - Test RMSE: 180.737
```

[redacted]

What do the result mean in this context?

Don’t forget to put the result in the existing context!

```
UMSCENT - Test RMSE: 0.210
Beer    - Test RMSE: 180.737
```

DATE	UMCSENT	beer
2017-07-01	93.4	4726.0
2017-08-01	96.8	4577.0
2017-09-01	95.1	4445.0

```
UMSCENT - Percentage Error relative to the mean: 0.22
Beer    - Percentage Error relative to the mean: 3.94
```