A Random Walk Through Time Series Techniques

Chris Hyland | 2018

Overview

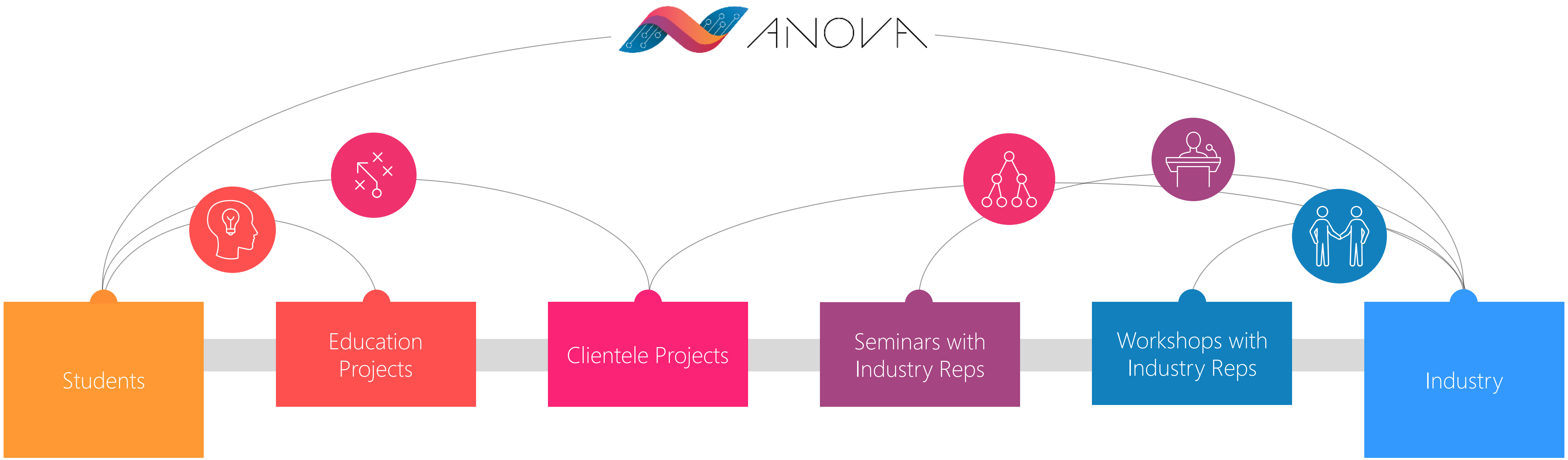
OVERVIEW

WHO WE ARE:

ANOVA is a society that aims to bridge the disconnect between academia and the technical industry through fostering a community of technically adept students via projects, workshops and talks

STORY TO DATE:

- Founded in December 2017
- Member base of over 300 students
- Worked with over 8 industry clients
- Ran industry seminars with BCG Digital Ventures, IBM, Accenture, Quantium, Minerva Collective
- 4 educational projects to upskills students in programming and data science



Outline

Intellify is a data science consultancy that works in the price optimization space

To optimize your price, you need to first be able to forecast your price

Our task was to forecast the demand component for Fast Moving Consumer Goods (FMCG)

Dataset contains 60 products in stores across America

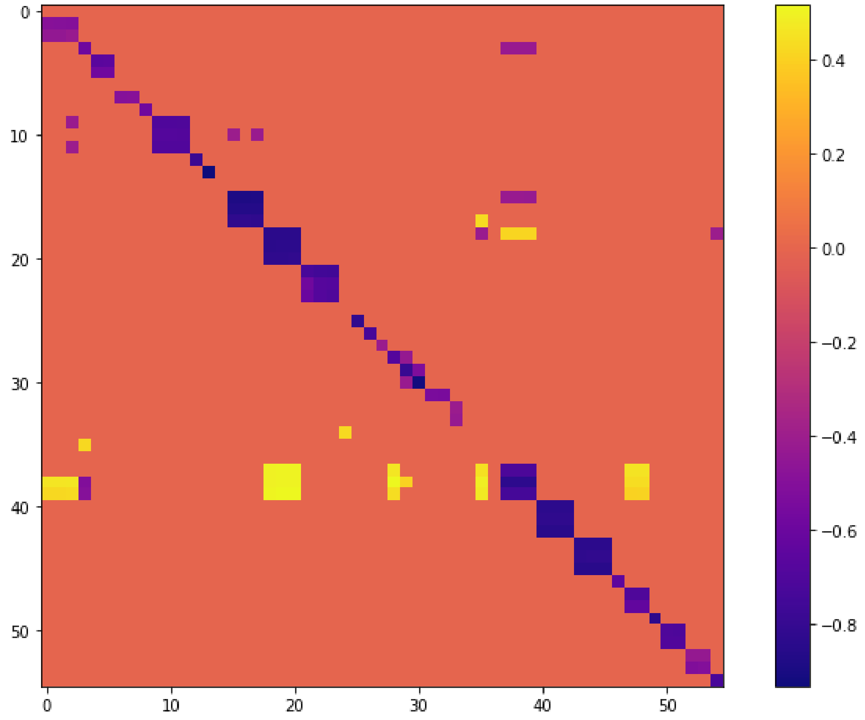
Setting Up

Initial Look into the Data

We built a correlation matrix of the products to see the correlation between quantities bought of each product on a week by week basis

Big issue is that a lot of the data was missing at sporadic moments

We focused our attention to a subset of products, specifically the pizza category of products, as they were the most correlated and one of the few products with complete data



MAPE

Important to be smart about the metric you use to evaluate your models

Since products were in different units, the forecast errors for one product can be larger than forecast errors for another product just because they are in different units

$$\mathrm{M} = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|,$$

where A_t is the actual value and F_t is the forecast value.

Hence, we use the **MAPE** as our metric for model evaluation

MAPE: Mean Absolute Percentage Error

- Scale Invariant
- Heavier penalties on negative errors (Actual < Forecast) underforecasts to overforecasts (an upper limit of 100% for underforecasts but no limit for overforecasts

Wold Decomposition

Stationarity of data

- Imposes a structure on the time series
- Certain statistical properties (constant mean, variance or the unconditional joint probability distribution does not change over time.

Wold Decomposition Theorem

- Powerful theorem stating that any stationary time series can be expressed as

$$Y_t = \sum_{j=0}^{\infty} b_j \varepsilon_{t-j} + \eta_t,$$

Y_t is the time series being considered,

ε_t is an uncorrelated sequence which is the innovation process to the process Y_t – that is, a white noise process that is input to the linear filter $\{b_j\}$.

b is the *possibly* infinite vector of moving average weights (coefficients or parameters)

η_t is a deterministic time series, such as one represented by a sine wave.

- This means that any stationary time series has a linear representation, so that a lot of the models we will apply have theoretical justifications!

Dickey Fuller Test

We can use the Dickey Fuller Test to test for stationarity of data

- Note: A small caveat as it is not the only way to test whether is a dataset nonstationary

Simple Example

$$y_t = \rho y_{t-1} + u_t$$

Subtract the lag of y from both sides

$$\Delta y_t = \delta y_{t-1} + u_t$$

Regress and test if the coefficient is 0 with the null hypothesis being the data is **not stationary**

Dickey Fuller Test

```
In [12]: from statsmodels.tsa.stattools import adfuller

         products = ["Prod_one", "Prod_two", "Prod_three"]

         """
         Run ADF test on each product and check test.
         """

         for prod in products:
             print("P-value for {} is {:.5f}".format(prod, adfuller((final_trans[prod]))[1]))

         P-value for Prod_one is 0.00000
         P-value for Prod_two is 0.00000
         P-value for Prod_three is 0.00000
```

We have stationarity!

With this, we operate under the assumption of stationarity and move on to apply time series models

Models

AR and MA Models

Autoregressive Model

How it works:

- Today's value is a linear combination of previous values

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t,$$

Notes:
- We can further extend this framework to more models

Autoregressive Moving Average Model

How it works:

- Today's value is a linear combination of past forecast errors which are assumed to be normally distributed

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t,$$

Notes:
- We can have further modifications such as ARIMA

Moving Average Model

How it works:

- Today's value is a linear combination of past forecast errors which are assumed to be normally distributed

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q},$$

Notes:
- Do not confuse with moving average smoothing

Choosing optimal lags

To prevent overfitting, we need to be careful about the parameters of our model

- We need to be careful of bias-variance tradeoff in our model
- Common strategy is to compute the Akaike Information Criterion (AIC)
- AIC = Likelihood of model + Penalty for number of parameters of model

A common model selection technique is to compute the AIC for different lags of the model and select the minimum model with minimum AIC

Code Example of Model Selection

```
In [26]: def aic_selection(data, p_lag, q_lag):
    '''
    Forward selection based on the AIC.

    Parameters
    ---------
    data: series
        The data to test for.
    p_lag: int scalar
        Maximum number of lags to test for AR.
    q_lag: int scalar
        Maximum number of lags to test for MA.

    Returns
    ---------
    results: tuple
        Tuple containing optimal p,q, and AIC score.
    '''
    best_score = np.inf
    opt_p, opt_q = 0, 0

    range_p, range_q= range(p_lag), range(q_lag)

    # Iterate through and fit different lags.
    for p in opt_p:
        for q in opt_q:
            try:
                model=sm.tsa.statespace.SARIMAX(y,  order=(i, 0, k)).fit()
                score=model.aic
                if (score<best_score):
                    best_score = model.aic
                    opt_p, opt_q = p, q
            except ValueError:
                pass

    return (opt_p, opt_q, best_score)
```

```
'''
Select best ARMA model based on AIC, MAPE for each product
'''
products = ["Prod_one", "Prod_two", "Prod_three"]
results = {}

for prod in products:
    y = data_time(train_data,prod)
    opt_p, opt_q, best_score = aic_selection(y, 3, 3)

    results.update({prod: [opt_p, opt_q, best_score.round(2)]})
```

	Product	Optimal P	Optimal Q	AIC
0	Prod_one	1	0	1912.13
1	Prod_two	1	0	1840.05
2	Prod_three	1	2	1856.30

Vector Autoregressive Models

Vector Autoregressive Models (VAR) can capture linear interdependencies between time series

- This can be quite useful for when we think one product's demand can help forecast another product's demand
- This is common when the products are **substitutes** or **complements** to each other

1) First we define a **structural VAR**

$$y_t = \gamma_{10} - \beta_{12} z_t + \gamma_{11} y_{t-1} + \gamma_{12} z_{t-1} + \epsilon_{yt}$$

$$z_t = \gamma_{20} - \beta_{21} y_t + \gamma_{21} y_{t-1} + \gamma_{22} z_{t-1} + \epsilon_{zt}$$

Vector Autoregressive Models

1) First we define a **structural VAR**. We have an **identification issue** or **simultaneous equation bias** (y causes z and z causes y)

$$y_t = \gamma_{10} - \beta_{12} z_t + \gamma_{11} y_{t-1} + \gamma_{12} z_{t-1} + \epsilon_{yt}$$

$$z_t = \gamma_{20} - \beta_{21} y_t + \gamma_{21} y_{t-1} + \gamma_{22} z_{t-1} + \epsilon_{zt}$$

2) We re-express it like so

$$\underbrace{\begin{bmatrix} \beta_{11} & \beta_{12} \\ \beta_{21} & \beta_{22} \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} y_t \\ z_t \end{bmatrix}}_{x_t} = \underbrace{\begin{bmatrix} \gamma_{10} \\ \gamma_{20} \end{bmatrix}}_{\Gamma_0} + \underbrace{\begin{bmatrix} \gamma_{11} & \gamma_{12} \\ \gamma_{21} & \gamma_{22} \end{bmatrix}}_{\Gamma_1} \underbrace{\begin{bmatrix} y_{t-1} \\ z_{t-1} \end{bmatrix}}_{x_{t-1}} + \underbrace{\begin{bmatrix} \varepsilon_{yt} \\ \varepsilon_{zt} \end{bmatrix}}_{\varepsilon_t}$$

$$Bx_t = \gamma_0 + \gamma_1 x_{t-1} + \epsilon_t$$

3) We now have a **reduced form VAR** after taking the inverse of B matrix. $x_t = A_0 + A_1 x_{t-1} + e_t$
4) We can estimate as usual via OLS

$$y_t = \alpha_{10} + \alpha_{11} y_{t-1} + \alpha_{12} z_{t-1} + e_{yt}$$

$$z_t = \alpha_{20} + \alpha_{21} y_{t-1} + \alpha_{22} z_{t-1} + e_{zt}$$

Vector Autoregressive Models

ANOVA

VAR(1) Model for 3 variables

```
model.summary()
```

```
Method:                       OLS
Date:           Tue, 11, Dec, 2018
Time:                    04:46:57
--------------------------------------------------------------------
No. of Equations:         3.00000    BIC:                  31.0664
Nobs:                     155.000    HQIC:                 30.9265
Log likelihood:          -3037.19    FPE:              2.45272e+13
AIC:                      30.8308    Det(Omega_mle):   2.27223e+13
--------------------------------------------------------------------
Results for equation Prod_one
================================================================================
                   coefficient       std. error           t-stat            prob
--------------------------------------------------------------------------------
const               288.619685        77.322744            3.733           0.000
L1.Prod_one           0.227743         0.306026            0.744           0.457
L1.Prod_two          -0.313328         0.436219           -0.718           0.473
L1.Prod_three         0.958356         0.257581            3.721           0.000
================================================================================

Results for equation Prod_two
================================================================================
                   coefficient       std. error           t-stat            prob
--------------------------------------------------------------------------------
const               252.519949        57.670685            4.379           0.000
L1.Prod_one          -0.001106         0.228247           -0.005           0.996
L1.Prod_two          -0.048135         0.325351           -0.148           0.882
L1.Prod_three         0.751965         0.192115            3.914           0.000
================================================================================

Results for equation Prod_three
================================================================================
                   coefficient       std. error           t-stat            prob
--------------------------------------------------------------------------------
const               307.512077        63.591147            4.836           0.000
L1.Prod_one          -0.051573         0.251679           -0.205           0.838
L1.Prod_two          -0.084623         0.358751           -0.236           0.814
L1.Prod_three         0.811113         0.211837            3.829           0.000
================================================================================

Correlation matrix of residuals
              Prod_one  Prod_two  Prod_three
Prod_one      1.000000  0.965009    0.914308
Prod_two      0.965009  1.000000    0.922331
Prod_three    0.914308  0.922331    1.000000
```

Extensions

2 Tricks to Improve Forecasts

ANOVA

STATE CLUSTERING:

- We noticed that the behaviour of the demand for products had **locality effects**
- We segmented the dataset so that we only forecasted demand for products for each state

MODEL COMBINATION:

- Combine forecasts from multiple models
- Weaknesses of one model can be offset by strengths of another model

RESULTS:

- Models improved by over 50% with MAPE scores of <15%

Model Combination

Recall that if we had a 1-step ahead forecast from 2 different models, which we denote as $\hat{y}_{t+1}^{(1)}$ and $\hat{y}_{t+1}^{(2)}$. Furthermore, we can compute the forecast errors which we denote as $e_{t+1}^{(1)}$ and $e_{t+1}^{(2)}$. From this, letting λ be the weight parameter, we denote the combined forecast as

$$\hat{y}_{t+1}^{c} = (1-\lambda)\hat{y}_{t+1}^{(1)} + \lambda\hat{y}_{t+1}^{(2)}$$

From this, the variance of the combined forecast error is

$$Var(e_{t+1}^{c}) = (1-\lambda)^2\sigma_1^2 + \lambda^2\sigma_2^2 + 2\lambda(1-\lambda)\rho\sigma_1\sigma_2$$

We can then optimise λ to minimise the variance, which gives us

$$\lambda^* = \frac{\sigma_1^2 - \rho\sigma_1\sigma_2}{\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2}$$

However, we need to use estimates so we have that

$$\hat{\lambda}^* = \frac{\hat{\sigma}_1^2 - \hat{\sigma}_{12}}{\hat{\sigma}_1^2 + \hat{\sigma}_2^2 - 2\hat{\sigma}_{12}}.$$

where we use $\hat{\sigma}$ as an estimator for σ,to be the residuals in our estimated model.

Conclusions

Results and Considerations

- Lots of other models we tried out like exponential smoothing, Holt Winters

- This is all even before adding in regressors where we used dimensionality reduction

- Machine Learning techniques could be an option

- Models and techniques discussed today returned a MAPE score of <15, which beats the benchmark MAPE of 25

Questions