**Econometric Approach to Time Series Analysis — Seasonal ARIMA**

At this post, we will talk about the analysis of time series data with Trend and Seasonal components. An econometric approach will be followed to model the statistical properties of the data. The business objective here is forecasting. We attempted to explain various concepts involved in time series modelling, such as time series components, serial correlation, model fitting, metrics, etc. We will use SARIMAX model provided by statsmodels library to model both, seasonality and trend in the data. SARIMA (Seasonal ARIMA) is capable of modelling seasonality and trend together, unlike ARIMA which can only model trend.

**Contents:**

1. Definition of time series data
2. Introduction to the project and data
3. Seasonal decomposition and Time series components: Trend, Seasonality, Cycles, Residuals
4. Stationarity in time series data and why it is important
5. Autocorrelation and partial autocorrelation
6. Data transformation: Log transformation and differencing
7. Model Selection and Fitting
8. Conclusion

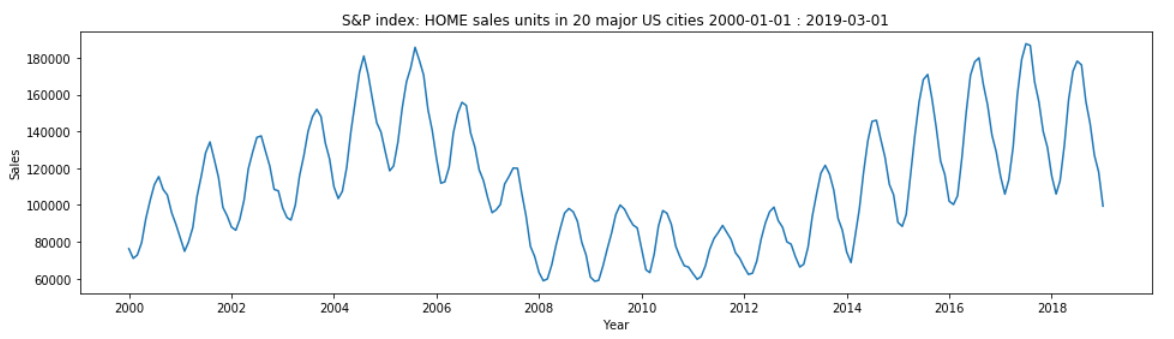
**1. Definition of time series data?**

Time series data is a sequence of data points measured over time intervals. In other words, data is a function of time f(t) = y.  
Data points can be measured hourly, daily, weekly, monthly, quarterly, yearly and also with smaller or larger time scales such as seconds or decades.

**2. Introduction to the project and data**

The data we are using in this article is a monthly home sales index for 20 major US cities between the years 2000 and 2019. (<https://fred.stlouisfed.org/series/SPCS20RPSNSA>). You can freely download many different economic time series data representing the US economy from this source. You may see 2 different versions of the same data, seasonally-adjusted and non-seasonally-adjusted. Seasonally adjusted data is time series data without a seasonal component. The version used at this post is not seasonally adjusted as we want to model the seasonality as well as trend. You may ask why people want to use seasonally adjusted data in the industry. Well, sometimes businesses may want to know the true effect of economic events on a particular data, which may overlap with a season. In that case, seasonality may hide or underestimate/overestimate the effect of an economic event. For instance: Heating oil producers may want to study the impact of declining petrol prices on heating oil prices. However, heating oil prices are increasing in winter, which is surprising as heating oil is petrol substance. The decrease in petrol prices should be reflected in the decrease in heating oil prices. However, in winter there is a big demand for heating, which causes a slight increase in prices. By removing the seasonal effect from the time series data, you may see that heating oil price actually follows a decreasing trend. The slight increase in the price was the seasonal effect. In Section 3, we will talk about seasonal decomposition in more detail.

If we look at the time series plot of the data we can observe an increasing trend in 2000–2006, a decreasing trend in home sales starting from 2007 till 2012 due to big financial crisis and increasing trend again till 2019. We can also observe seasonality in the data as usually the housing market is not active at the beginning of a year and sales usually go high in mid-year and again sales getting lower by the end of the year. Seems like warmer seasons, especially summer is the good season for the American housing market.



**3. Seasonal decomposition and Time series components: Trend, Seasonality, Cycles, Residuals**

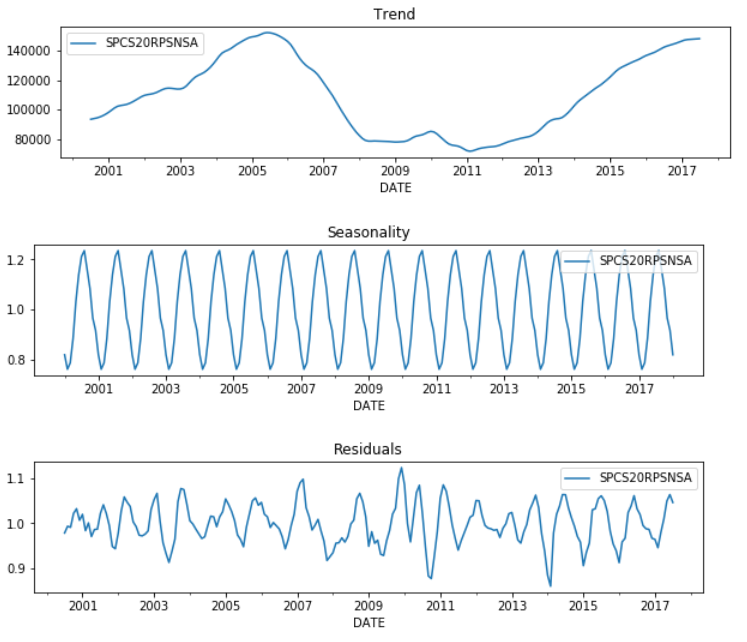
Time series data Y is composed of a combination of Trend, Cycles, Seasonality and Residuals. Obviously, you may come across with time series where it doesn’t have a Trend, Cycles or Seasonality. So, it is your task to identify the components of Y. Definition of the terms are given below:

**Trend —**long-term upward or downward movement.   
**Cycle —**periodic variation due to economic movements. It is different from seasonal variation. The cycle is the variation of autoregressive component of time series data. Cycles occur within longer time intervals such as every 6–10 years, whereas seasonal variation occurs in shorter time intervals.   
**Seasonality** **—**variation in data caused by seasonal effects. Ice cream sales are high in summer, heating oil sales are high in winter but low in summer.  
**Residuals —**a component that is left after other components have been calculated and removed from time series data. It is randomly, identiclly and independently distributed (i.i.d). Follows ~ N(0,1)

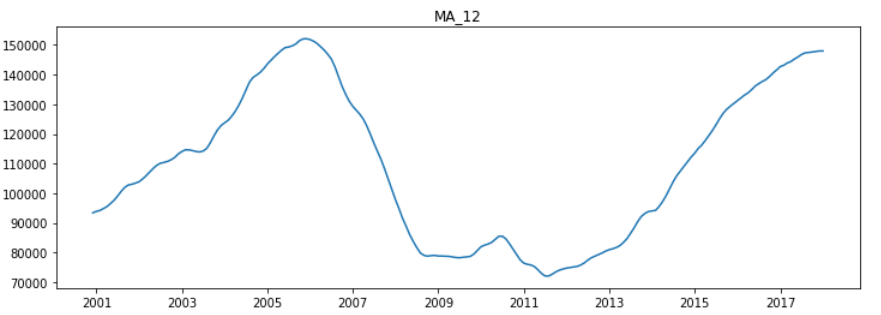
Statsmodels library has a function called seasonal\_decompose, which decomposes Y into Trend, Seasonality and Residuals. Although, it is a naive decomposition algorithm, in practice it is very intuitive and works well for time series data where T, S and R are obvious. Before explaining the below graphs I would like to talk about the interaction among these components.   
Time series data Y can take either an additive or a multiplicative form. In additive form, time series Y is formed by the sum of time series components, namely, T, S, C, R:   
***Y = T + C + S + R***

In multiplicative form time series Y is formed by the product of time series components:  
***Y = T \* C \* S \* R***

We decompose time series data into Trend, Seasonal component and Residuals using seasonal\_decompose() function from statsmodels. Don’t be surprised if the function returns all 3 components even though you assume that they do not exist for a particular time series data. In reality, these components are generated by a simple algorithm involving simple math, that’s why the decomposition function cannot say a component doesn’t exist, despite the calculated value is not significant. So, you will see these three components for any time series data. You have to know how to read the results and decide which model (ARIMA or SARIMA) to fit the data.

Plot of seasonal decomposition from statsmodels library

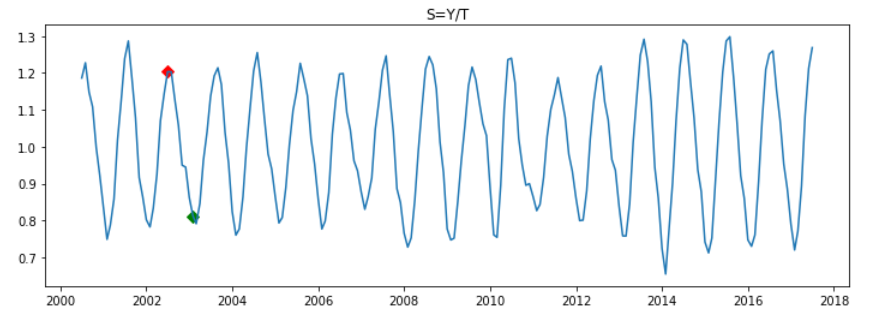
We will try to decompose the data into components ourselves to better understand their derivation and usage. The trend can be calculated taking moving average with window size= 12. The below plot is very similar to the Trend generated by stats model library.

Moving average with window size=12Plot of Moving average window size=12

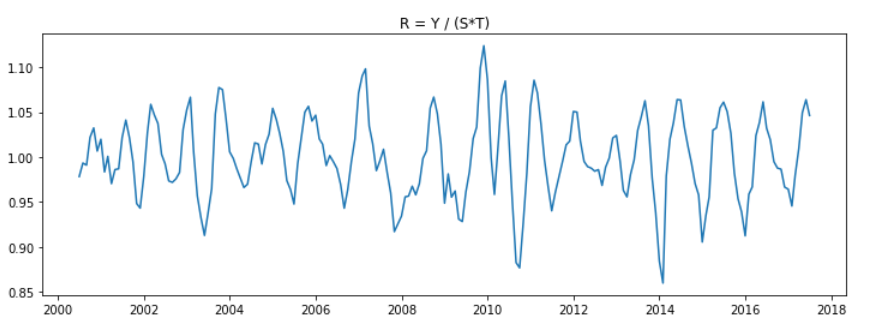
Remember the multiplicative model, ***Y = T \*C\* S\* R***? Dividing ***Y*** by **T** may give you ***Y/T=C\*S\*R*** and we assume that Residuals are too small for this data as the time series plot looks smooth. We have a very small data, thus we cannot detect an economic cycle. We get a seasonal component when we divide time series by trend, ***S=Y/T***.

Code snippet for seasonality decomposition

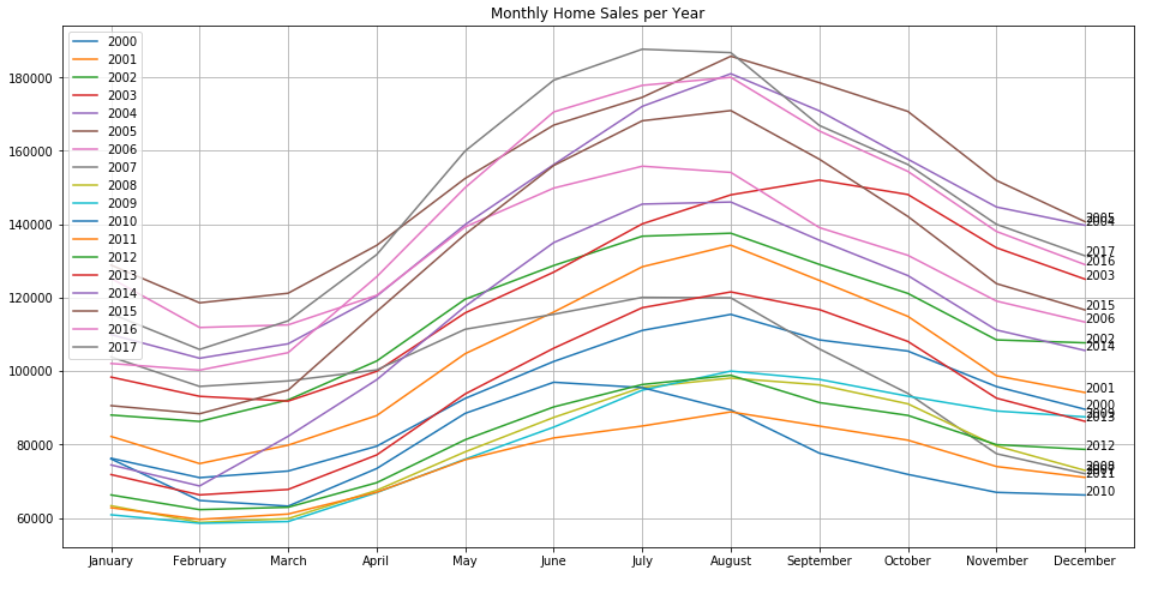
Graph of seasonality is a little bit harder to understand, however, the explanation given here is sufficient. *y=1.2(red marker)* means there were *20%* more sales in July-2000. In other words, June has a seasonal effect of *20%* or 1.2 \*T. *y=0.8(green marker)* in Feb-2003 shows a *20%* decrease in sales due to the winter season. So, for some time series data not at this particular case, you may see the seasonal effect is very small i.e *y=0.0001*. It shows a very small seasonal effect, which shouldn’t even be considered significant.

Seasonal Component

Residuals can be computed as ***R = Y/(S\*T)***

Code snippet for residuals decompositionResiudals component

There are some other ways to detect seasonality. In the below graph, monthly home sales for each year is plotted and as you can see every year follows pretty much the same pattern with a slight difference. House sales are high in summer, lower in winter months.

Code snippet for the monthly home sales graphMonthly sales per year

**4. Stationarity in time series data and why it is important**

When we have trend and/or seasonality in a time series data we call it non-staionary.

Stationarity means the statistical properties of data, such as mean, variance and standard deviation remain constant over time. Stationary data should be i.i.d from normal distribution~N(0,1). In simpler language, every data point should be independent from the previous data point. The histogram of data points should look like a bell curve.

Why do we want the statistical properties to remain the same over time? Well, because we make statistical assumptions (a good example could be OLS assumptions) about the sample data in due course of model building and the model will only be capable of performing under those assumptions. When the statistical properties of the data changes, the model is no longer capable of representing the true nature of data as data properties have been changed. That’s why our forecasting/prediction results will no longer be valid. Changing mean/variance will require us to fit another model and this model may be valid for a short period of time and we have to abandon it again and fit a new model. See, how inefficient and unreliable this process looks like. We have to make time series data stationary before fitting a model. We can make time series stationary by transforming the data. Usually, differencing is used to make the data stationary. We will talk about it in Section 6, below.

So, how can we test whether a time series data is stationary or not? The first is just eyeballing the time series plot and identify trend or seasonality. Secondly, you may divide the data into 3 different sets and calculate mean and variance for each set and confirm whether mean and variance for each set is substantially different or not. The third option is to use one of the statistical tests provided in statsmodels library.

Augmented Dickey-Fuller test is the most popular amongst others, where the null hypothesis, H\_0 = data is not stationary. ADF test result provides test statistic and P value. P value >= 0.05 means the data is not stationary, otherwise, we reject the null hypothesis and say data is stationary.

 I assume you know what hypothesis testing is and what P value means. If you are not quite familiar with these terms, then look at the p-value and if it is smaller than 0.05 (p-value < 0.05) then data is stationary if p-value >= 0.05 data is not stationary. ADF test confirms that the original time series data is not stationary with *p-value =0.0803366374517756*

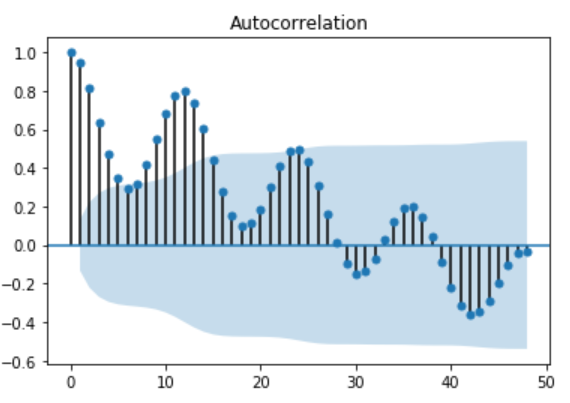
Code snippet for ADF test

Time series data is not stationary. Adfuller test pvalue=0.0803366374517756

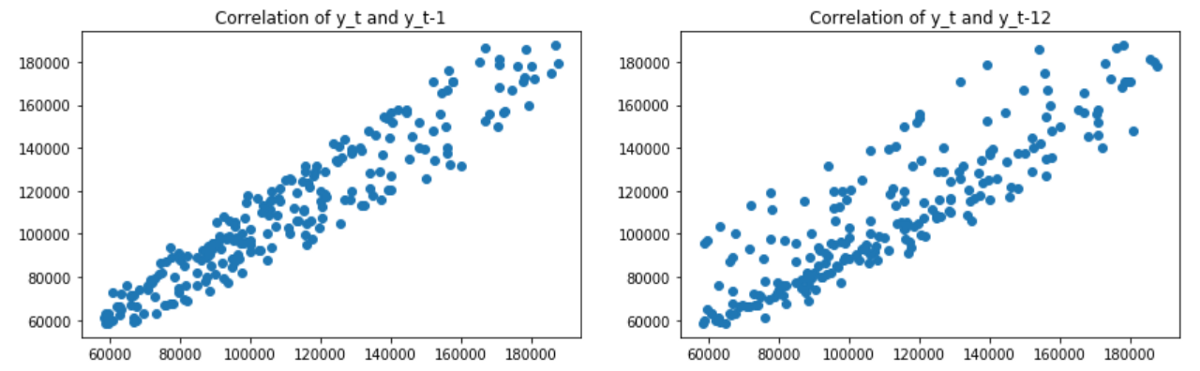
**5. Autocorrelation and partial autocorrelation**

We have to take a look at ACF and PACF plots, before making the data stationary as we will use these plots a lot from now on.

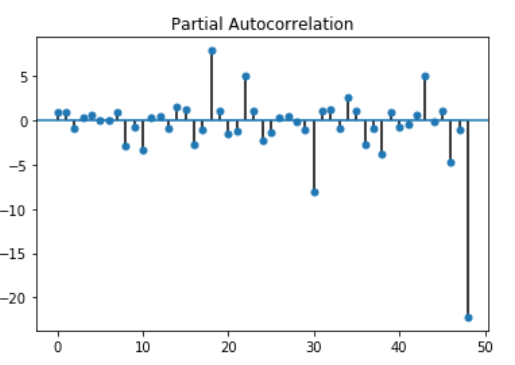
Autocorrelation plot shows the correlation of time series data with its own lagged values. For example, autocorrelation at lag=1 shows the correlation between y\_t and y\_t-1. At lag=2, corr(y\_t, y\_t-2). At lag=12 corr(y\_t, y\_t-12). Every data point at time t having a high correlation with a data point at time t-12 denotes seasonality.

Autocorrelation plot of original home sales index data

The below code snippet and scatter plots may help you to better understand what correlation between lagged data, namely, autocorrelation.

Code snippet for correlation of lagsCorrelation between lag values

Back to ACF plot, blue shaded area at the autocorrelation plot shows significance level. So, correlation coefficients within the shaded area show weak correlation at those lags and we don’t consider them significant in the analysis.  
The partial autocorrelation function (PACF) gives the partial correlation of a stationary time series with its own lagged values.

Partial autocorrelation plot of original home sales index data

PACF removes the correlation contribution of other lags and gives the pure correlation between two lags without the effect of others.

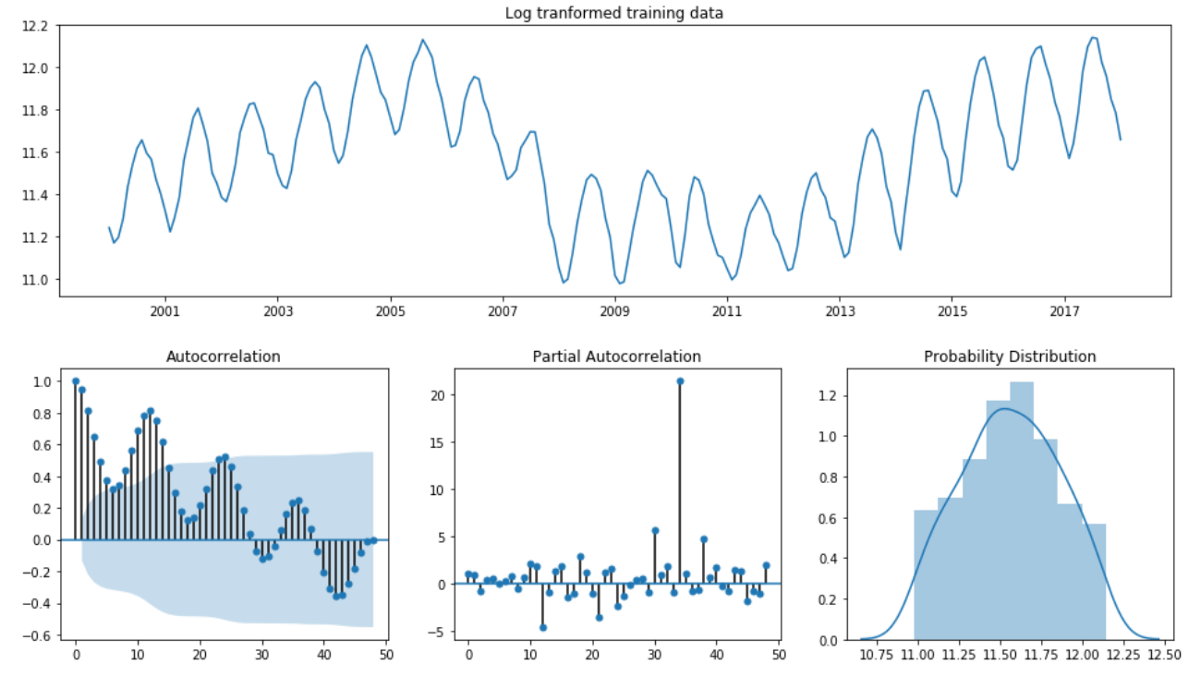
We use ACF and PACF to choose a correct order for AR(p) and MA(q) values of an ARIMA model. For AR order p look at PACF plot and choose a lag value which has a significant correlation factor before correlations get insignificant. For MA order q look at ACF plot and do the same. Don’t forget you should only get these values from the ACF and PACF plots of stationary time series, not the above plots. The ACF and PACF plot given above are the plots of original data, which is non-stationary.

**6. Data transformation: Log transformation and differencing**

So, let’s transform the data to make it stationary. We split the original data into training and test data. Training data will contain US home sales data from 2000 to 2008 and test data will contain data from 2018 to 2019. Don’t forget that you cannot do random sampling as you are doing for cross-sectional data. We have to keep the temporal behaviour (dependence on time) of time series data.

Home sales index data can be formulated as a multiplicative model where Y= T\*S\*R. I am ignoring Cycles, as it is not actually present in this data. (S)ARIMA models are linear models, like Linear Regression. We can not fit a linear model SARIMA to data generated by a process Y = T\*S\*R. We have to make Y linear before fitting a linear model. As you are aware of from math ***Log(a\*b) = log(a) + log(b)***. We have to log-transform the data to make it linear. ***log(Y) = log(T) + log(S) + log(R).*** Log transformation makes data linear and smoother.

log\_transformed\_data = np.log(training\_data)  
plot\_data\_properties(log\_transformed\_data, ‘Log tranformed training data’)

Properties of log transformed data

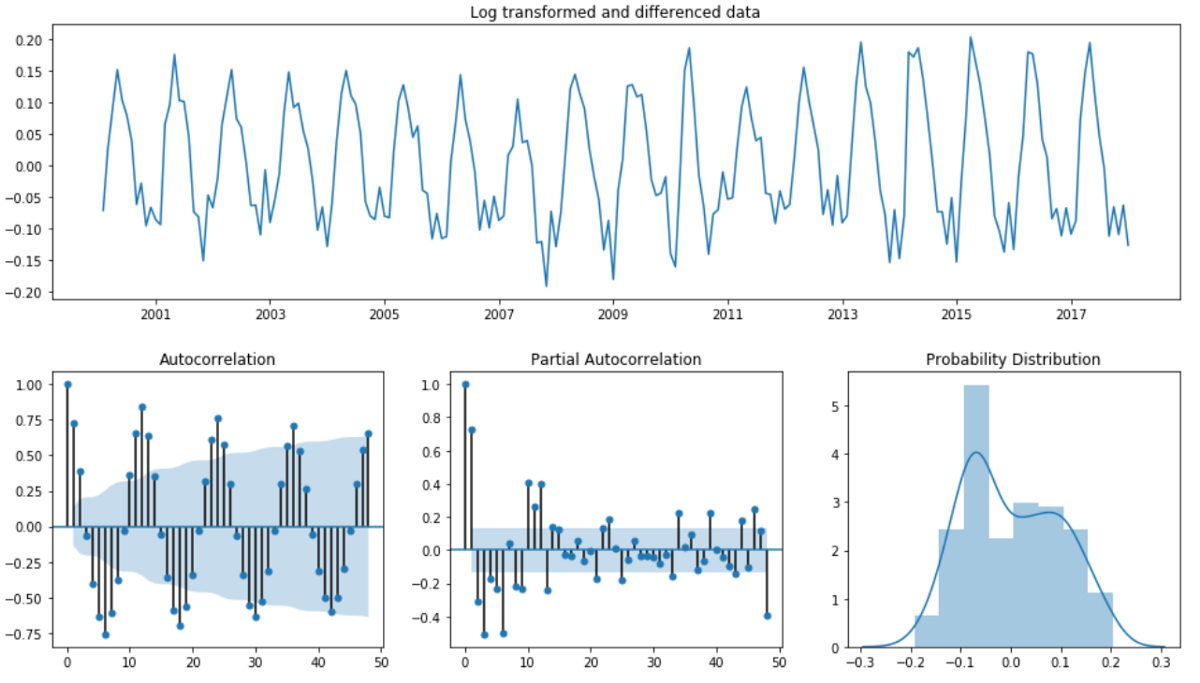
Sometimes, log transformation on itself can make data stationary, but it is not the case here.

test\_stationarity(log\_transformed\_data)  
Time series data is not stationary. Adfuller test pvalue=0.22522944188413385

Differencing is a basic operation or data transformation. It is the difference between y at time=t and y at time=t-x. ***diff\_1 = y\_t — y\_t-1***

Differencing makes the data stationary as it removes time series components from the data and you are left with changes between time periods. Notice, first order differencing took away only Trend not Seasonality. Data is still not stationary as it contains seasonal effects.

logged\_diffed\_data = log\_transformed\_data.diff()[1:]  
plot\_data\_properties(logged\_diffed\_data, 'Log transformed and differenced data')

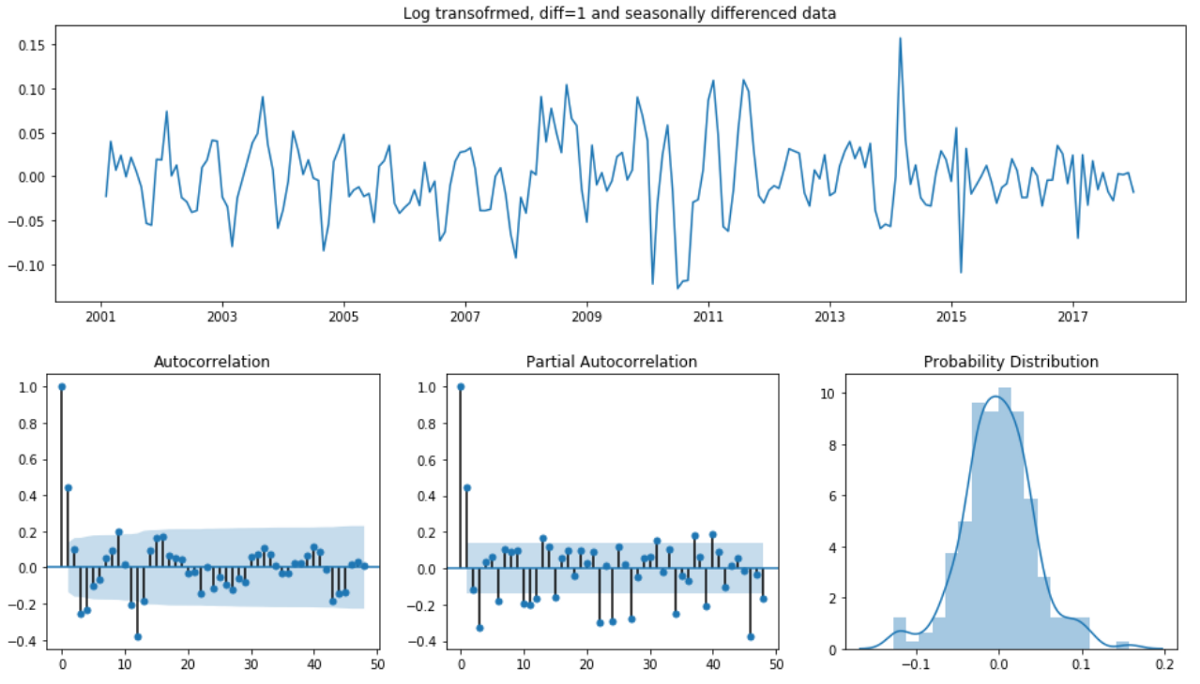
Log transformed and 1st order integrated: non-stationary

test\_stationarity(logged\_diffed\_data)  
Time series data is not stationary. Adfuller test pvalue=0.20261733702504936

We have to take 12th order difference to remove seasonality. You may ask how I decided to take 12th order difference not 6th or 8th or other order. We have discussed in seasonality section. Monthly data has seasonality at lag=12, weekly data has at lag=4 and daily has at lag=30. Or you can derive it from the ACF plot. The12th, 24th, 36th lags are highly correlated.

Data is stationary now. If you look at the histogram of data it looks like Normal bell curve. Stationary data is randomly i.i.d distributed and the plot looks like white noise. White noise is just an example of stationary time series data.

seasonally\_diffed\_data = logged\_diffed\_data.diff(12)[12:]  
plot\_data\_properties(seasonally\_diffed\_data, 'Log transofrmed, diff=1 and seasonally differenced data')

Stationary data

test\_stationarity(seasonally\_diffed\_data)  
Time series data is stationary. Adfuller test pvalue=0.0006264163287311492

I have used Shapiro normality test below to confirm that the data is normally distributed. It is one of the properties of stationary data. So, why differencing makes the data stationary? By differencing you only get the small changes of y between time steps. These changes usually are normally distributed with constant mean and variance, which is a property of stationary data.

Normality Test — Shapiro

shapiro\_normality\_test(seasonally\_diffed\_data.SPCS20RPSNSA)

Data follows normal distribution: X~N(-0.0, 0.043)  
Shapiro test p\_value=0.006

**7. Model Selection and Fitting**

As transformed data is stationary now we can proceed to model fitting phase. We had a brief chat about SARIMA before. I want to elaborate on this particular model. SARIMA, Seasonal ARIMA is a special member of ARIMA family which can model seasonal component of time series data. Just to recap what ARIMA means:

AR — Auto-Regressive model means time series data is regressed on its lagged values. Lagged values become independent variables, whereas time series itself becomes the dependent variable.

**y = a\_0 + a\_1\*y\_t-1 + a\_2\*y\_t-2, ….., a\_k\*y\_t-k.**

The main task here is to choose how many time steps to be used as independent variables. Do not let the word time series or lagged values to confuse you, they are just independent variables. In linear Regression, you could look at the correlation between independent and dependent variables and choose highly correlated variables as your features. Here you should do the same. But, you don’t have to calculate the correlation between lagged values and target variable, because you can use PACF to determine how many lags to use. PACF of **stationary** data has significant autocorrelation at lag=1 and the next autocorrelation at Lag=2 becomes insignificant. Ideally, AR order p should be 1. However, we will see in the model selection process that, we have to do parameter search on p to find the optimal value. An initial guess will help to define which values to use for a grid search. In this case, p = [0–2] would be sufficient.

I — order of integration: Basically, how many times you have differenced the data. We had it once d=1. Do not forget to fit the model to not differenced data when you set parameter d=1, as the algorithm will differentiate it. If you fit model to stationary data, thenyou don't need differencing anymore. You can leave d=0. We need differencing just to make the data stationary.

MA — Moving Average model: Time series y is regressed on residuals w.

**y = a\_0 + a\_1\*w1 + a\_2\*w2 + …. + a\_k\*wk**

Look at ACF plot to determine MA order (q) of the ARIMA model. ACF suggests order q=1 for MA part of the ARIMA model. However, we should do a grid search to find an optimal model. I suggest looking at values q=[0–2]

Seasonal model — Seasonal features have to be added to the model together with AR and MA and it has 4 parameters (P, D, Q, s).   
Think of P, D and Q parameters being similar to AR, I and MA parameters, but only for a seasonal component of the series.

Choose P by looking at PACF and Q by looking at ACF. The number of seasonal differences has been taken is D. Frequency of seasonal effect is defined by s.

P = 1 — because we have significant correlation at lag=12, however, they are not strong enough and we may not need to have an AR variable in the model. That’s why we should grid search on P = [0–2]

D=1 — we differenced for seasonality once

Q=1 — as we have strong correlation at lag=12 according to ACF plot. Let’s perform grid search on parameter Q=[0–2], too.

s=12 — seasonality frequency, every 12 months

best\_sarima\_model function below performs grid search on (p,d,q) and (P,D,Q,s) parameters and finds the best model taking statistical metrics AIC, BIC, HQIC as evaluation criteria. Lower AIC, BIC, HQIC means better model. These metrics reward goodness-of-fit (log-likelihood) and penalises overfitting. In our case having many lagged features leads to overfitting. AIC, BIC and HQIC balance the tradeoff between likelihood and degrees of freedom. You can see this property in their formula. I will not get into the details of other metrics, but will give an example below for supporting my point of using AIC:

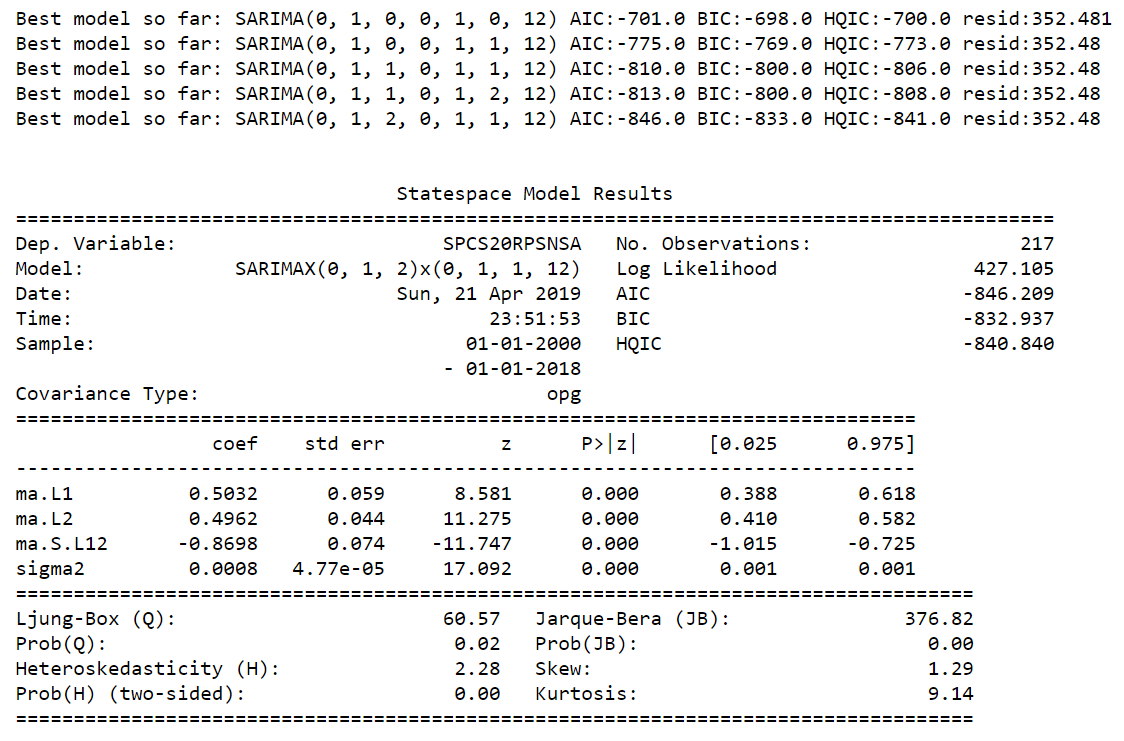
-k is number of estimated parameters in the model, in other words, number of features (lag terms).   
-L is the maximum of the likelihood function.  
**AIC = 2k — 2ln(L)**

I have seen many examples in the industry, using only one of those metrics as a model selection criteria, but you may come across the cases where AIC of a model can be lower than another, while BIC is higher. That’s why try to choose a model over the other if 2 of the 3 metrics are lower.

Code snippet for model selection

Note, we are fitting a model to log-transformed data and as we have set d=1 and D=1 parameters and the model will do the differencing for us. We evaluated SARIMA models with the parameters we have identified above. The summary below shows the best model or in other words lowest AIC, BIC, HQIC. The best model suggests that we don’t need to have AR features, but only MA and seasonal MA features.

best\_model, models = best\_sarima\_model(train\_data=log\_transformed\_data,p=range(3),q=range(3),P=range(3),Q=range(3))

Model Summary of the best Sarima model

ARIMA or SARIMA models are OLS based models, that’s why all OLS assumptions are applicable to this family of models. I don't want to elaborate on these assumptions here. It is a topic of another article. However, we have to confirm that our model aligns with those assumptions. P values of coefficients are <= 0.05. Residuals follow a normal distribution, highly concentrated around 0. Residuals are stationary and homoscedastic. There is no serial correlation among residuals.

shapiro\_normality\_test(best\_model.resid[1:])  
Data follows normal distribution: X~N(-0.026, 0.389)  
Shapiro test p\_value=0.0

We will predict home sales from 2018–01–01 to 2019–01–01. I will use MAPE — mean absolute percentage error to evaluate the model performance. Best model we have got is SARIMA(order=(0,1,2),seasonal\_order=(0,1,1,12). I prefer MAPE error metric in time series analysis as it is more intuitive. Sklearn doesn’t provide MAPE metric, that’s why we have to code it ourselves. Formula:

https://cdn-images-1.medium.com/max/1200/1*IX9aOt_e6vTXl5naOhWVTg.jpegMAPE formula

Code snippet for MAPE

When you use predict function, there are some nuances to be careful about its parameters:

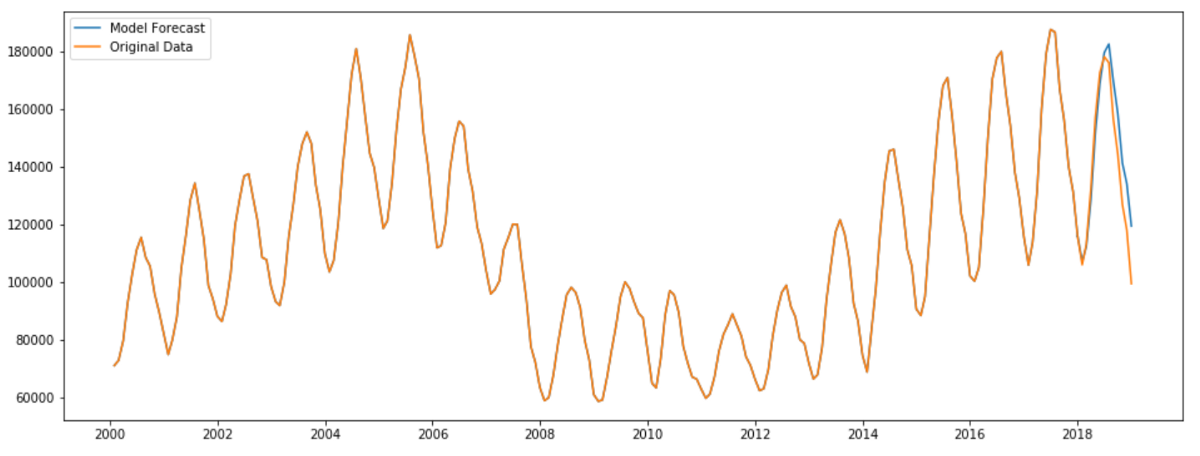
1. type = ‘levels’ means predicted values will be at the same level with endog/training values, in our case they were log transformed and not diffed at all. Then, if you notice we take np.exp() to scale the predicted values to original data. Remember, np.exp(np.log(a)) = a. So, np.exp(np.log(original data)) = original data
2. dynamics = True, then use the predicted value for time = t as a predictor for time = t+1.

preds\_best=np.exp(best\_model.predict(start=test\_start\_date,end='2019-01-01', dynamic=True, typ='levels'))  
print("MAPE{}%".format(np.round(mean\_abs\_pct\_error(test\_data,preds\_best),2)))

MAPE:6.05%

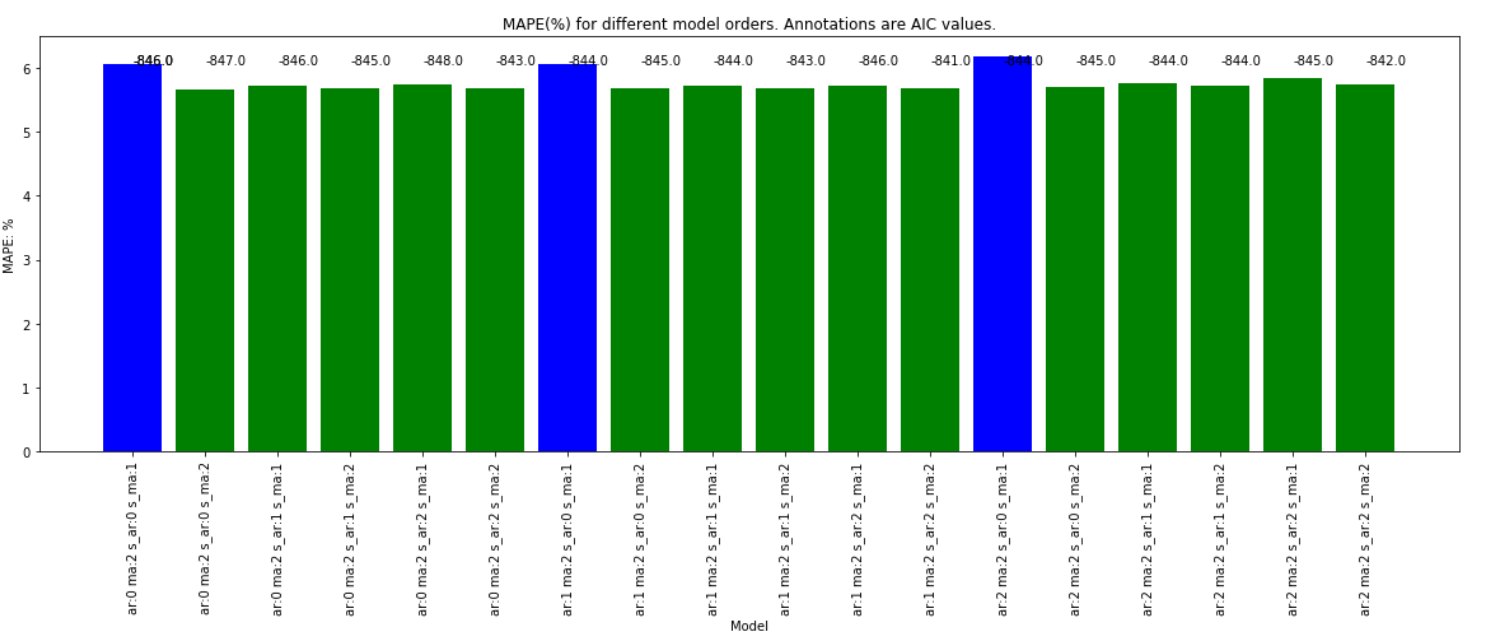
We make approx. 6% error in our prediction. It doesn’t mean the model will underperform at 6% of the time. Rather it translates as the predicted value will be offset from the real value 6% on average.

Plot the predicted values with original data and see the results. What can we infer from the below plot? Well, a lot! The model can successfully capture the seasonal effect, however, cannot do the same with the trend. Home sales go downward trend, however, the model cannot capture it well. It knows that sales go down but due to seasonal effect, however, there is a downward trend after 2018 which it struggles to predict. This is due to small training data we have got.

Forecasted data

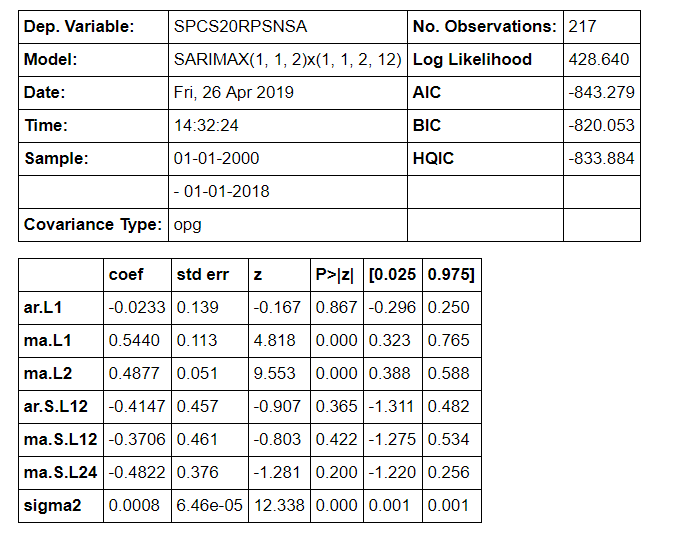
 If we had a larger data set, we could identify an economic cycle, and model it. Possibly, every 6–7 years of housing sales follow a reduction. Or if this downward trend continues in 2019 our 2020 prediction would definitely capture the trend.

Another option to capture trend quicker is to add an AR term to the model. If we add 1 or 2 AR terms to the model it could react to the trend quicker and have less MAPE. The below plot displays MAPE for each model. Models performing better than the best model in terms of test MAPE are in green.



We have added AR terms to the model and we have got improvement in test metrics.

agile\_model = SARIMAX(endog=log\_transformed\_data,order=(1,1,2), seasonal\_order=(1,1,2,12),enforce\_invertibility=False).fit()  
agile\_model.summary()

Model summary

Test MAPE now is 5.67%, improved from 6.05%, which is the test MAPE of the optimal model.

agile\_model\_pred = np.exp(agile\_model.predict(start=test\_start\_date,end=’2019–01–01', dynamic=True, typ=’levels’))  
print(“MAPE{}%”.format(np.round(mean\_abs\_pct\_error(test\_data,agile\_model\_pred),2)))

MAPE:5.67%

However, if you look at AIC, BIC and HQIC we get higher values, which means we traded off the model generality. We know that we have few data points roughly 300 and having 6 features in a linear model may lead to overfitting. If you take a look at the model summary above, P values of feature coefficients ar.L1, ma.L2, ar.S.L12, ma.S.L12 and ma.S.L24 are higher than 0.05%.

**8. Conclusion**

We have talked about many different concepts above, which are used during the analysis and model building phase. The below steps summarize the approach we have taken and can be used as a guidance or framework for a similar project:

1. Identify if the model is multiplicative or additive
2. Identify time series components: Trend, Cycle, Seasonality, Residuals
3. Transform data to make it linear
4. Make data stationary if it is not
5. Based on step 2, choose ARIMA or SARIMA model
6. Define order parameters for each model variable/feature
7. Do grid search and choose an optimal model based on AIC, BIC, HQIC
8. Forecast and calculating forecasting error: MAPE, MAE, etc.