Contents

Process Outline (Flowchart)

1. **Load the data.**
2. **Pre-processing:** Creating timestamps, converting the datatype of date/time column, making the series univariate, etc.
3. **Check and make the series stationary:** Most time series methods assume the data is stationary. Therefore, we need to check that using tests like ADF and perform transformations like differencing to make the series stationary.
4. **Determine d value:** The number of times a difference operation needs to be performed to make the series stationary is the **d** value.
5. **Create ACF and PACF plots:** ACF and PACF plots are used to determine the input parameters **p and q** for our ARIMA model
6. **Determine the p and q values:** Read the values of p and q from the plots in the previous step.
7. **Fit ARIMA model:** Using the processed data and parameter values we calculated from the previous steps to fit the ARIMA model.
8. **Predict values on validation set:** Predict the future values.
9. **Calculate RMSE:** To check the performance of the model, check the RMSE value using the predictions and actual values on the validation set.

https://www.analyticsvidhya.com/blog/2018/08/auto-arima-time-series-modeling-python-r/

Preprocessing

Timestamps

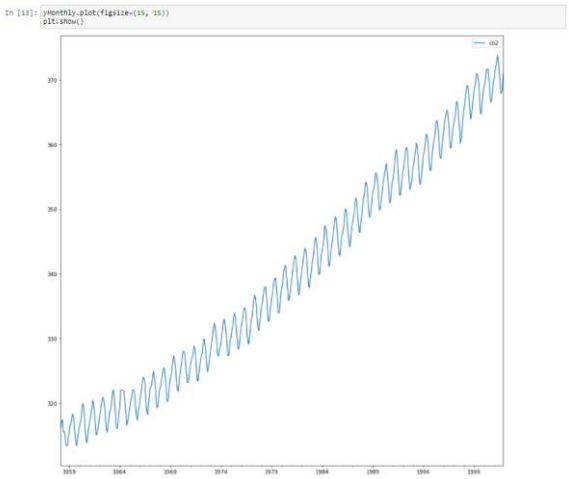
Perth data

Missing values

Visualizing Data

The first step in Time Series Analysis is visualizing the data. A simple plot can give us the basic information.

```
dataframe.plot(figsize = (15, 15))
plt.show()
```



Decomposing a Timeseries

A simple plot like above may not always be easy to understand. A time series can be thought of as a combination of **Level, Trend, Seasonality and Noise**

Level – The average value of the series

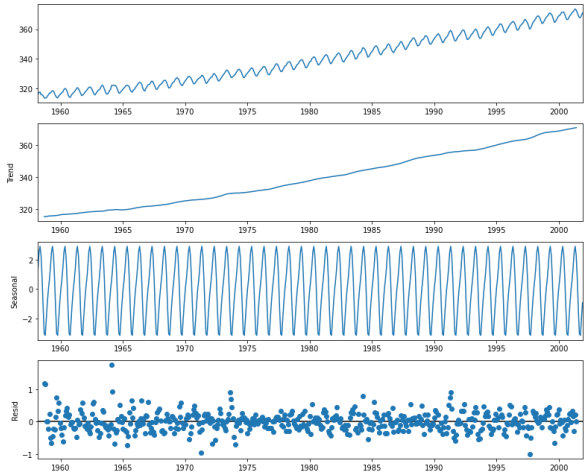
Trend - The increasing or decreasing value in the series

Seasonality – Repeating cycles in the series

Noise – Random variation in the series

We can decompose the given time series in as shown below.

```
decomposition = sm.tsa.seasonal_decompose(yMonthly,
model='additive')
fig = decomposition.plot()
plt.show()
```



Stationarity Tests

A stationary time series is one whose statistical properties like mean and variance are constant over time. Most statistical forecasting methods assume that the time series data is stationary. Therefore, it is important to check if the data is non-stationary.

One of the most popular methods for checking stationarity is **Augmented Dickey-Fuller test.**

Null Hypothesis (H0): If failed to be rejected, it suggests the time series is non-stationary. It has some time dependent structure.

Alternate Hypothesis (H1): The null hypothesis is rejected; it suggests the time series is stationary. It does not have time-dependent structure.

We interpret this result using the **p-value** from the test. A p-value below a threshold (such as 5% or 1%) suggests we reject the null hypothesis (stationary), otherwise a p-value above the threshold suggests we fail to reject the null hypothesis (non-stationary).

Below code shows a function that accepts a time series and does a stationarity test

```
def adf_test(ts, signif=0.05):
    dftest = adfuller(ts, autolag='AIC')
    adf = pd.Series(dftest[0:4], index=['Test Statistic','p-value','# Lags','# Observations'])

    for key,value in dftest[4].items():
       adf['Critical Value (%s)'%key] = value
    print (adf)

    p = adf['p-value']
    if p <= signif:
        print(f" Series is Stationary")
    else:
        print(f" Series is Non-Stationary")
```

Output of the test

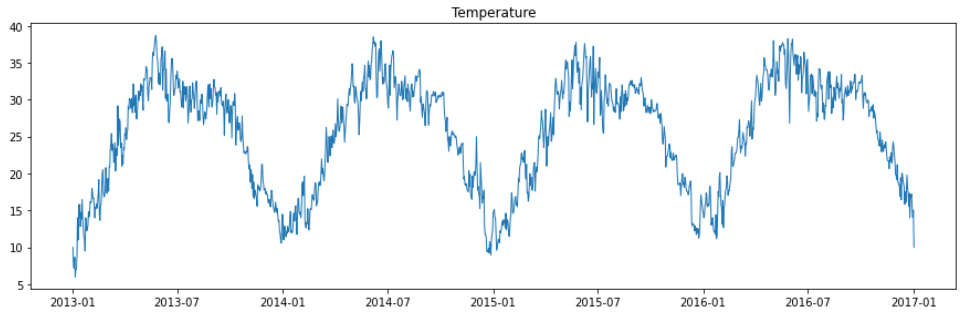
```
Test Statistic            2.359810
p-value                   0.998990
# Lags                   14.000000
# Observations          511.000000
Critical Value (1%)      -3.443212
Critical Value (5%)      -2.867213
Critical Value (10%)     -2.569791
dtype: float64
 Series is Non-Stationary
```

How to make a time series stationary?

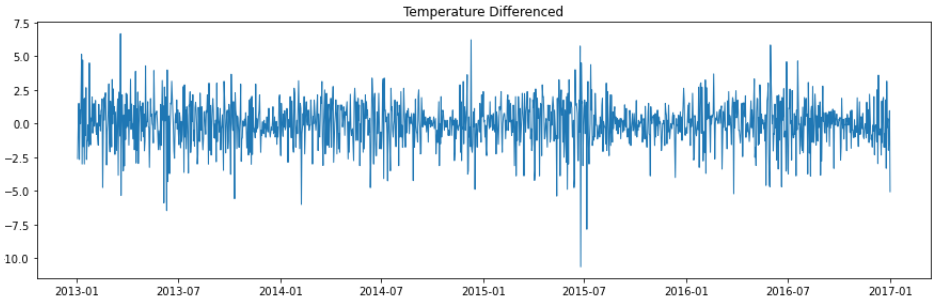
Differencing

The first difference of a time series is the series of changes from one period to the next.

The image below shows the original values of a time series.



The image below shows the first difference of the above time series.



Univariate Analysis

ARIMA Models

ARIMA stands for Auto Regressive Integrated Moving Average. It has 3 main parameters. **p, q, d**

p = Periods to lag (AR Terms)
if P = 3 then we will use the three previous periods of our time series in the autoregressive portion of the calculation

q = This is the number of MA terms. These are lagged forecast errors. If q is 5, the predictors for x(t) will be e(t-1)….e(t-5) where e(i) is the difference between the moving average at ith instant and actual value. This variable denotes the lag of the error component, where error component is a part of the time series not explained by trend or seasonality

d = d refers to the number of differencing transformations required by the time series to get stationarity.

Determining the Parameters

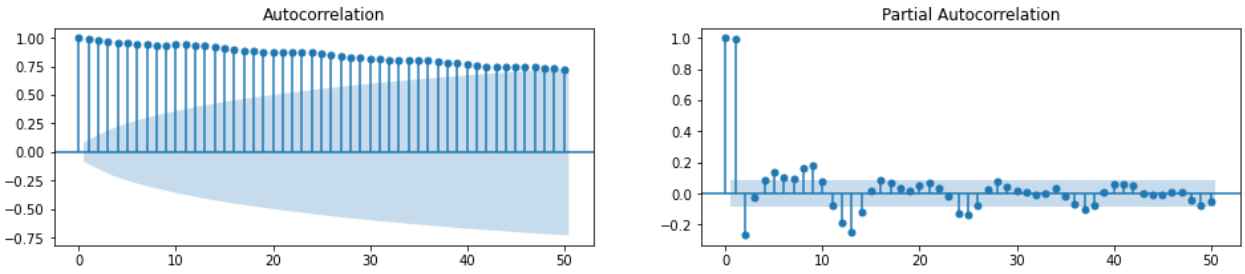
We use ACF and PACF plots to determine the values for p, q and d

https://arauto.readthedocs.io/en/latest/how_to_choose_terms.html

http://people.duke.edu/~rnau/411arim3.htm

Autocorrelation and Partial Autocorrelation Plots

```
fig, axes = plt.subplots(1,2,figsize=(16,3))
plot_acf(yMonthly.co2.tolist(), lags=50, ax=axes[0])
plot_pacf(yMonthly.co2.tolist(), lags=50, ax=axes[1])
```



ACF and PACF plots explained. How to read them and determine p q and d

Auto ARIMA and Seasonal ARIMA

Determining the parameters from ACF and PACF plots is time taking and confusing. We can use python packages for doing that automatically. The code below uses auto_arima to determine the best parameters.

```
model = pm.auto_arima(train, start_p=1, start_q=1,
                      test='adf',       # use adftest to find optimal 'd'
                      max_p=3, max_q=3, # maximum p and q
                      m=1,              # frequency of series
                      d=None,           # let model determine 'd'
                      seasonal=False,   # No Seasonality
                      start_P=0,
                      D=0,
                      trace=True,
                      error_action='ignore',
                      suppress_warnings=True,
                      stepwise=True)

print(model.summary())
```

Output

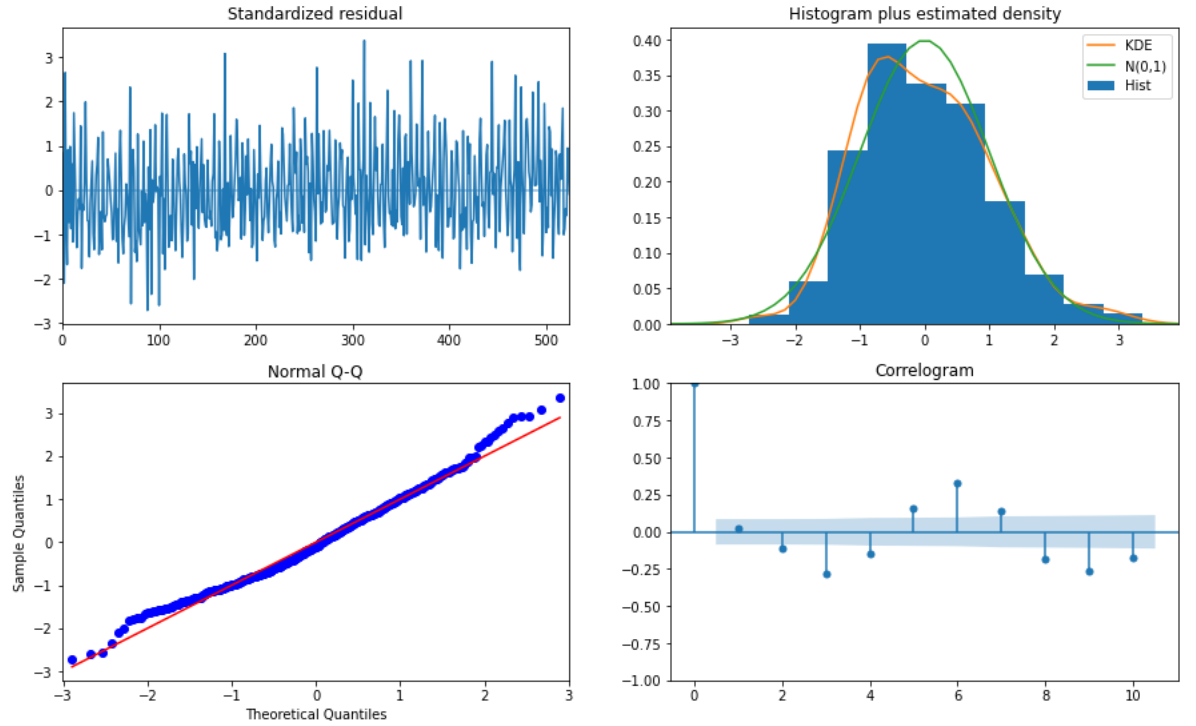
```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=1086.163, Time=0.08 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=1427.684, Time=0.01 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=1131.425, Time=0.05 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=1177.579, Time=0.07 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=1428.664, Time=0.02 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=862.819, Time=0.28 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=1035.934, Time=0.06 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=860.049, Time=0.37 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=981.827, Time=0.09 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=861.659, Time=0.58 sec
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=857.822, Time=0.50 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=1042.048, Time=0.12 sec
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AIC=999.332, Time=0.43 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=1023.975, Time=0.21 sec
 ARIMA(3,1,3)(0,0,0)[0] intercept   : AIC=999.725, Time=0.67 sec
 ARIMA(2,1,2)(0,0,0)[0]             : AIC=916.513, Time=0.09 sec

Best model:  ARIMA(2,1,2)(0,0,0)[0] intercept
Total fit time: 3.645 seconds
```

The best values for p, d, q are: 2, 1, 2

Analyzing Residuals

Analyzing the results of ARIMA. Residual plot diagnostics.

```
model.plot_diagnostics(figsize=(15,9))
plt.show()
```



Seasonal ARIMA

Holt-Winters Model

Another popular model for univariate time series analysis

https://medium.com/analytics-vidhya/python-code-on-holt-winters-forecasting-3843808a9873

https://towardsdatascience.com/holt-winters-exponential-smoothing-d703072c0572

https://machinelearningmastery.com/how-to-grid-search-triple-exponential-smoothing-for-time-series-forecasting-in-python/

Multivariate Analysis

Granger Causality

https://medium.com/swlh/using-granger-causality-test-to-know-if-one-time-series-is-impacting-in-predicting-another-6285b9fd2d1c

Granger causality test is used to determine if one time series will be useful to forecast another variable by investigating causality between two variables in a time series.

It is based on the idea that if X causes Y, then the forecast of Y based on previous values of Y AND the previous values of X should best result in the forecast of Y based on previous values of Y alone.

https://towardsdatascience.com/granger-causality-and-vector-auto-regressive-model-for-time-series-forecasting-3226a64889a6

https://www.machinelearningplus.com/time-series/time-series-analysis-python/

VAR

VARMAX

“Convergence warnings in VARMAX are common, because vector autoregressions typically have poor identification for many of the parameters.”

Unless you know you want VARMAX for some reason, it is more robust to stick with VAR.

Prophet (By Facebook)

Official documentation

https://facebook.github.io/prophet/docs/quick_start.html#python-api

Univariate

https://machinelearningmastery.com/time-series-forecasting-with-prophet-in-python/#:~:text=The%20Prophet%20library%20is%20an,and%20seasonal%20structure%20by%20default.

To use Prophet for forecasting, first, a Prophet() object is defined and configured, then it is fit on the dataset by calling the fit() function and passing the data.

The Prophet() object takes arguments to configure the type of model you want, such as the type of growth, the type of seasonality, and more. By default, the model will work hard to figure out almost everything automatically.

The fit() function takes a DataFrame of time series data. The DataFrame must have a specific format. The first column must have the name 'ds' and contain the date-times. The second column must have the name 'y' and contain the observations.

This means we change the column names in the dataset. It also requires that the first column be converted to date-time objects, if they are not already (e.g. this can be down as part of loading the dataset with the right arguments to read_csv).

Multivariate (Kinda)

https://www.youtube.com/watch?v=XZhPO043lqU&ab_channel=AIEngineering

LSTM

DTW