ARIMA(Auroregressive Integrated Moving Average) Model

Model explanation

AR, MA, and ARMA Model

- Autoregressive (AR) Model

$$y_t = \beta_0 + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \beta_3 y_{t-3} + \cdots + \beta_p y_{t-p}$$

- Moving Average (MA) Model

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

- Autoregressive Moving Average (ARMA) Model

$$y_t = \beta_1 y_{t-1} + \beta_2 y_{t-2} + \beta_3 y_{t-3} + \cdots + \beta_p y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

- Auto Regression sub-model - AR term refers to the past values used for forecasting the next value. The AR term is defined by the parameter ‘p’ in arima.
- Integrated sub-model - This sub-model performs differencing to remove any non-stationarity in the time series.
- Moving Average sub-model - It uses past errors to make a prediction.

$$\text{ARIMA} \quad \underbrace{(p, d, q)}_{\uparrow \atop \left(\begin{array}{l}\text{Non-seasonal part}\\ \text{of the model}\end{array}\right)} \quad \underbrace{(P, D, Q)_m}_{\uparrow \atop \left(\begin{array}{l}\text{Seasonal part}\\ \text{of the model}\end{array}\right)}$$

Parameters

- p: It is the order of the Auto Regression (AR) sub-model. It refers to the number of past values that the model uses to make predictions.

- d: It is the number of differencing done to remove non-stationary components.
- q: It is the order of the Moving Average (MA) sub-model. It refers to the number of past errors that an ARIMA Model can have when making predictions.
- The function can either use the Grid Search technique, or Random Search technique to find the optimal parameter values. It tries multiple combinations of p,d, and q and then selects the optimal ones.
- In Auto ARIMA, the model itself will generate the optimal p, d, and q values which would be suitable for the data set to provide better forecasting.

Install package

! pip install pmdarima
from pmdarima.arima import auto_arima

Auto_arima document can be found here

Steps for ARIMA implementation

The general steps to implement an ARIMA model are –

- Load the data: The first step for model building is of course to load the dataset Preprocessing: Depending on the dataset, the steps of preprocessing will be defined. This will include creating timestamps, converting the dtype of date/time column, making the series univariate, etc.
- Make series stationary: In order to satisfy the assumption, it is necessary to make the series stationary. This would include checking the stationarity of the series and performing required transformations
- Determine d value: For making the series stationary, the number of times the difference operation was performed will be taken as the d value
- Create ACF and PACF plots: This is the most important step in ARIMA implementation. ACF PACF plots are used to determine the input parameters for our ARIMA model
- Determine the p and q values: Read the values of p and q from the plots in the previous step

- Fit ARIMA model: Using the processed data and parameter values we calculated from the previous steps, fit the ARIMA model
- Predict values on validation set: Predict the future values
- Calculate RMSE: To check the performance of the model, check the RMSE value using the predictions and actual values on the validation set

Fbprophet model

What is Facebook Prophet and how does it work?

Facebook Prophet is an open-source algorithm for generating time-series models that uses a few old ideas with some new twists. It is particularly good at modeling time series that have multiple seasonalities and doesn't face some of the above drawbacks of other algorithms. At its core is the sum of three functions of time plus an error term: growthg(t), seasonality s(t), holidays h(t) , and error e_t :

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

The Growth Function (and change points):

The growth function models the overall trend of the data. The new idea incorporated into Facebook prophet is that the growth trend can be present at all points in the data or can be altered at what Prophet calls "changepoints". Changepoints are moments in the data where the data shifts direction.

The growth function has three main options:

Linear Growth: This is the default setting for Prophet.

Logistic Growth: This setting is useful when your time series has a cap or a floor in which the values you are modeling becomes saturated and can't surpass a maximum or

minimum value (think carrying capacity).
Flat: Lastly, you can choose a flat trend when there is no growth over time (but there still may be seasonality).

The Seasonality Function:

The seasonality function is simply a Fourier Series as a function of time. You can also choose between additive and multiplicative seasonality.

The Holiday/Event Function:

The holiday function allows Facebook Prophet to adjust forecasting when a holiday or major event may change the forecast.It takes a list of dates (there are built-in dates of US holidays or you can define your own dates) and when each date is present in the forecast adds or subtracts value from the forecast from the growth and seasonality terms based on historical data on the identified holiday dates.

Demo code can be found here:
https://facebook.github.io/prophet/docs/installation.html#python

```
pip install prophet
import pandas as pd
from prophet import Prophet

# read data from NY times
df  = pd.read_csv("")

# create pandas time series
df.date = pd.to_datetime(df.date)
df.set_index('date',inplace=True)

# create timeseries readable by fbprophet
ts = pd.DataFrame({'ds':df.index,'y':df.new_cases})
#ts['cap'] = 30000 # unused in linear growth
#ts['floor'] = 0 # unused in linear growth
ts.head()
```