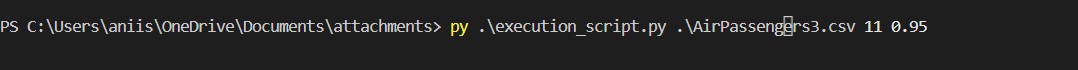
**DEMO:**

The process will require the user to supply the python file name called execution\_script.py, a path to the data file with one column with name ‘y’, and the number of predictions to be generated. The user can also specify a fourth argument, which is optional, for the confidence interval. This must be a value between 0 and 1, else the model will use 0.95 as default and generate a 95% confidence interval.

The images below describe the process:

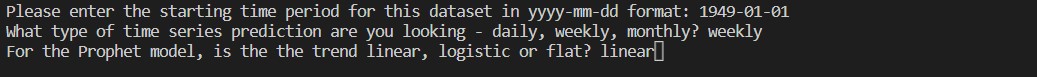
In the image below, the user is calling the python file containing the code to execute the algorithms. Here, the code file name is execution\_script.py, data file is Airpassengers3.csv, number of future predictions is 11 and confidence interval is 0.95



Once the user hits enter, the script will prompt the user to enter the starting date or date when the first observation was measured in “yyyy-mm-dd” format



Once, the user hits enter, the script will ask for what type of time series data is in the .csv file – daily, weekly, monthly. The user must enter one of the three options, and the string has to match exactly one of the options. This will be followed by another question related to the Prophet algorithm. The user can specify if the growth trend is linear, logistic or flat, and the model will fit the data accordingly. The input must match exactly one of the options, else a while loop will force the system to go back to the previous question, and the process has to be repeated again. The funnel, here, is relatively small. Hence, I did not work on optimizing it, but it is certainly something I would work as a part of the improvement.



The looping above was to ensure that the script does not crash after the user has been interacting with script through various layers. Instead of closing the script and having the user to repeat the process from the beginning, I felt that it was better if the above set of three questions can force the user to enter the correct option without making them exit forcefully.

If the time series type is weekly or monthly, the next question is about which type of ARIMA model to use – ARIMA or ARIMA with seasonality (SARIMA) – If the user types ‘y’ it is SARIMA, else, it will execute an ARIMA model. Note that, if the type of time series is daily, then only the Prophet algorithm will be executed. The script will not deploy the ARIMA/SARIMA model in this case because it does not work well on daily data.



If the trend for the Prophet model is logistic, then the below two questions will also be asked of the user. They are the estimated minimum and maximum value for the time series. In the example being discussed, we know that air passengers will always have a minimum of zero. For the maximum, the user can use his experience to decide what a practical maximum value is



After the above set of questions have been answered correctly, and if the data file exists in the correct format, and there are no other errors, the output will be generated as below.

Graphical user interface, text

Description automatically generated

The Seasonal ARIMA model can take time for weekly predictions relative to the Prophet algorithm. This is because of the seasonality parameter, which is 52 and equal to the number of weeks in a year. The ARIMA model does not have that problem.

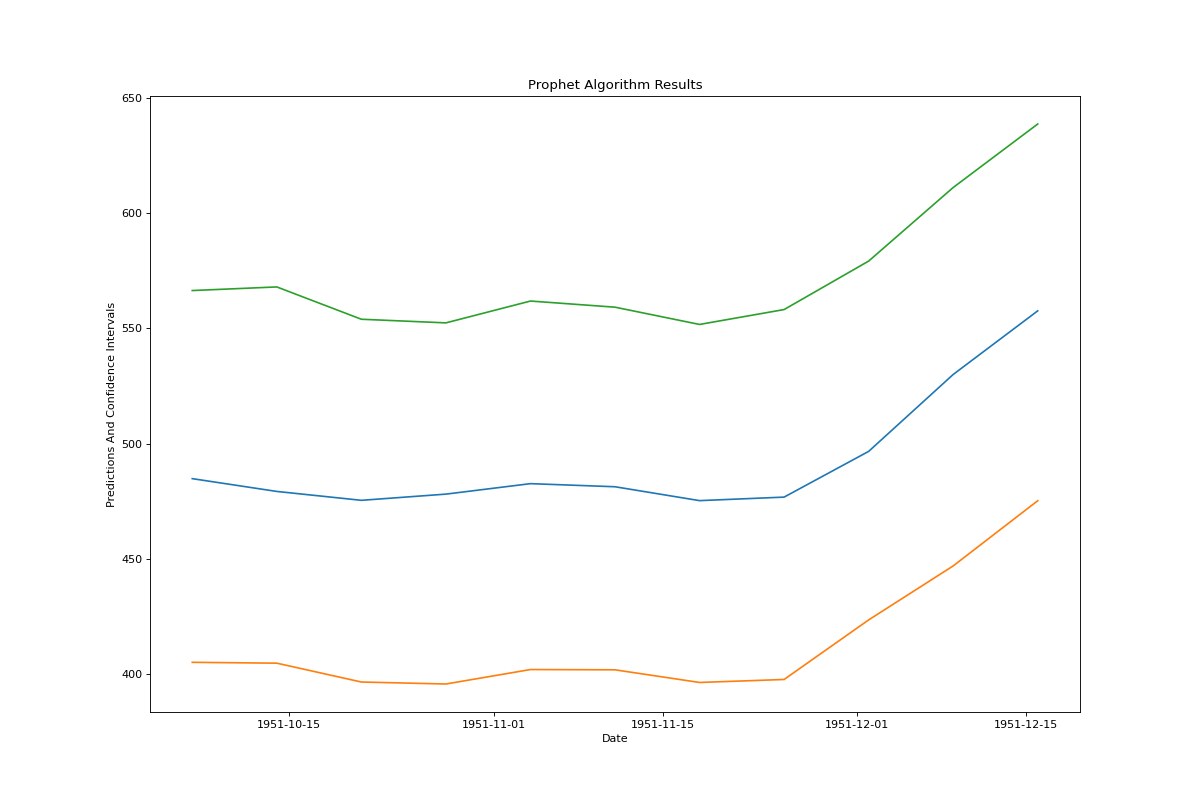
If the model is for weekly or monthly predictions, then both the Prophet and ARIMA/SARIMA algorithm will be used to generate predictions. Two csv files, two line charts and a diagnostic report analysing the quality of the ARIMA/SARIMA model will be generated and reported to the user.

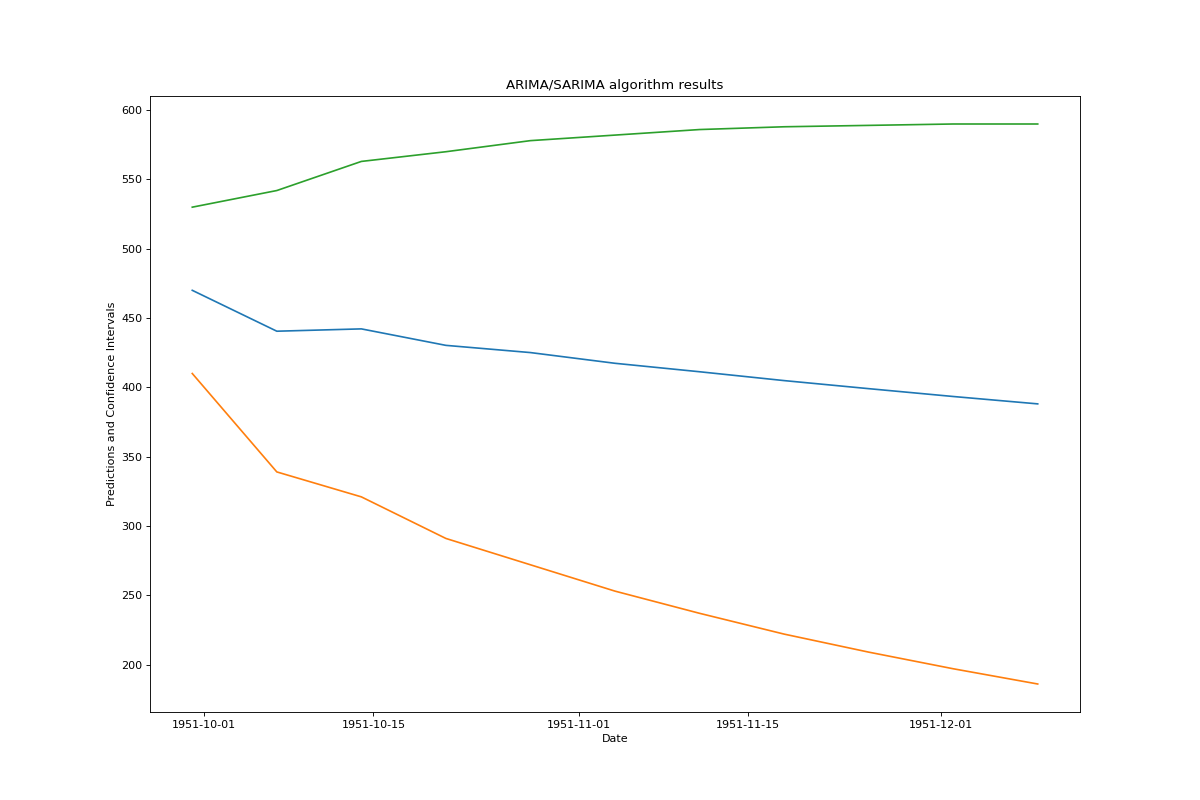
The csv files – Prophet\_TS.csv, ARIMA\_SARIMA\_Model.csv will be saved into the user’s working directory with four columns which include date stamps, expected value, lower and upper confidence intervals. The plots attached below are the diagnostics for the ARIMA/SARIMA model. They allow the user to investigate if the errors incurred by the model are systematic or random. If they are random, that means the model is capturing a significant amount of signal in the data.

Chart

Description automatically generated

The plots attached below correspond to the Prophet time series algorithm and the ARIMA/SARIMA process respectively

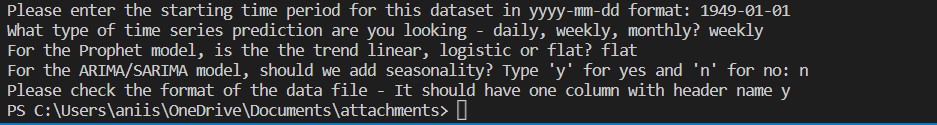




The script has also has some try-except blocks to catch any error incurred by the user. These errors are mostly related to the input at the beginning, and may also catch any errors related to the data file or the algorithm itself.

In the snapshot below, the user is entering AirPassengers.csv, which does not have the ‘y’ column. The script will stop and report the error to the user. Note that both Airpassengers.csv and Airpassengers3.csv have been included in the submission folder. The latter is the correct file and the former is the one with the incorrect format.





In the scenario below, the user, for the first question, has entered none. The script will not crash, but will continue; And once the user answers the last question in the box, it will revert back to the first question, which is about the trend. Hence, until all questions are not answered correctly and match to one of the options, this loop will continue



The goal of developing a prediction methodology using the Prophet algorithm designed by Facebook was achieved. In addition, I also managed to achieve the extendable goals of ARIMA/SARIMA model. The script generates the predictions, charts and saves them in the user’s working directory.

**References:**

<https://towardsdatascience.com/time-series-forecasting-with-arima-sarima-and-sarimax-ee61099e78f6>

<https://towardsdatascience.com/using-open-source-prophet-package-to-make-future-predictions-in-r-ece585b73687>