Andreas Dunn

May 5th, 2022

Healthcare Industry Dataset: 10,000 rows, 50 features

In the medical industry, readmission of patients is such a problem that an external organization penalizes hospitals for excessive readmissions (Centers for Medicare and Medicaid Services or CMS). When it comes to readmission penalties, studies show that many hospitals are overconfident and underprepared. The percentage of hospitals penalized for readmissions has increased each year since CMS began imposing penalties, and according to the CMS reporting, as much as 78 percent of hospitals were fined in fiscal year 2015. However, three-quarters of hospitals feel confident in their ability to reduce readmissions, and only 55 percent of them anticipate receiving a penalty this year. Given the historical trend and the addition of COPD and Hip and Knee replacement to the list of medical conditions measured, the percentage of hospitals penalized will likely be much higher than 55 percent. Additionally, although hospitals are applying various reduction strategies, fewer than 1 in 5 utilize technology that is specific to reducing their readmissions, so they may not be doing all that they can.

Time Series Analysis (ARIMA)

Using a time series model, can we make future profit predictions for the hospital? Predicting profit is an important analysis technique because it will allow the hospital to better understand their customers and the hospital can use a time series analysis in tandem with other analysis to reduce readmission rates. The goal of this analysis is to better understand the trend of profit over the course of two years so that we can make conclusions about our customer base in hopes of reducing readmission to the hospital. The scope of this analysis is to look at revenue streams over a two year period.

A times series is an observation or set or observations over time, like stock prices in a given year. However, since time is a continuous variable, the unit of analysis of time can be broken down into several different intervals of time (month, week, day, hour, minute, etc.) *Time series* specifically refers to the series of time that we want to analyze, like stock returns daily over a year, and *time series modeling* refers to the model we create to analyze that time series.

Time series modeling is a method of analyzing that time series in order to make predictions on the future, visualizing the trend over time, and examining the statistical properties of the series. In order to simplify the expression of time series for the purposes of study (and for our machine learning models to work), certain assumptions are made about the data. For instance, stationarity is an assumption of time series which says that the duration of the time series should have no effect on the statistical properties of the time series. Put more simply, this means that our model needs to work with data that have a constant mean and variance over time.

Secondly, the assumption of autocorrelation is also important. Firstly, lets discuss how time series observations are correlated with one another. The stock on the closing time on Friday, for instance, is intrinsically linked with its value on the closing on Tuesday. It is also correlated with the closing price on Thursday, and therefore, any previous day. For time series, there are two different flavors of correlation: autocorrelation and partial autocorrelation.

Using stock prices as an example, autocorrelation means that the price of a stock today is both directly affected by the price of the stock yesterday *and* indirectly affected by the price of the stock the day before yesterday (since yesterday's price is correlated with the day before yesterday). Autocorrelation considers both direct and indirect effects. Partial autocorrelation refers specifically and only to the effects of yesterday. The lag factor of the correlation is how far in the past you wish to examine.

The data set has a daily time step that runs consequently without gaps from Day 1 to Day 730; 2 years of time series data in total. There are no missing or null values in the dataset as demonstrated in

the Jupyter notebook attached with this document. In order to varify that our data are stationary, we can visually examine the overall trendline to try and notice patterns ourselves or we can confirm with a statistical test. One of the most popular and widely used tests is the Augmented Dickey-Fuller test. In the Augmented Dickey-Fuller test we test a null-hypothesis that the data are non-stationary and test against it using a p-value with an alpha level. (just like is done with a regression task).

The evaluation of the stationarity is shown in the graph below along with the results from our Dickey-Fuller test:

```
Results of Dickey-Fuller Test:
p-value = 0.1997. The series is likely non-stationary.
Test Statistic                  -2.218319
p-value                          0.199664
#Lags Used                       1.000000
Number of Observations Used    729.000000
Critical Value (1%)             -3.439352
Critical Value (5%)             -2.865513
Critical Value (10%)            -2.568886
dtype: float64
```

Our next step is to separate the data into train and test sets so that we can evaluate our model before putting it into use. Technically, since our model is a time series, we do not necessarily need to split up the data into training and test sets because our observations are *not* independent (they are correlated as discussed above). However, since time series is a *supervised* machine learning task it is consider to be a best practice to evaluate the model using a train/test split.

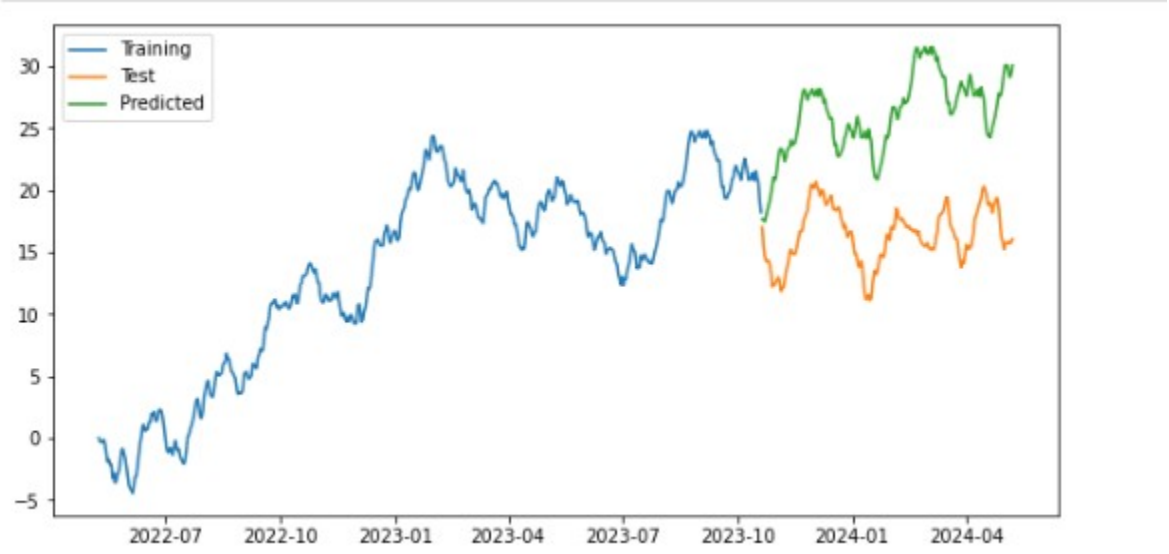
We will be using the ARIMA model to make our predictions for this time series task. In order for an ARIMA model to function correctly, we need to have a dataset that is stationary and has no seasonal components. The reason that we need to make our data stationary is that it allows us to make predictions; it essentially turns a times series into a regression problem. In order for regression to work, the variables need to be independent of one another. In other words, the variables cannot be correlated with one another and therefore they cannot share their dependency on time (not a function of time). Stationarizing the data will remove the auto-correlation problem that time series inevitably invokes which is done through the auto_arima method in Python.

The ARIMA model is a model that integrates (the I term) two other models. The AR term of ARIMA stands for auto-regressive and it simply means that this model forecasts future values by using the previous values, or lags, as regressors. The AR model then integrates with the MA model which stands for Moving Average which seeks to predict new values by using the errors, or residuals, of the previous values. You can think of them as two sides to the same coin, and are similar to an OLS

regression prediction task where AR uses the previous values and MA uses the errors. Together they make the AR-I-MA or ARIMA model.

We make the data stationary by differencing the data using python and then examine a plot of the data to see if the data is stationary. With our ARIMA results you can see that only 1 differencing was needed to get stationary data.

The AR part of our model will use the PACF plot to determine the correct value for P. The MA part of the model will use the ACF plot to determine to correct number for Q, where d is the number of differencing required to make the data stationary. However for this model we will call the auto_arima method which will calculate the ideal values for p, d, q based on the AIC score and do the rest automatically.



We chose an ARIMA model that had the highest AIC score which allowed us to incorporate a model with p,d,q, of 1,1,0 as shown in the model summary attached to this paper. The prediction interval was 90 days ahead into the future which represents a standard business quarter. The business quarter aligns with our original research question and therefore is a justified length.

```
[35]: #import necessary libraries
      from sklearn.metrics import mean_squared_error
      from math import sqrt

      #calculate RMSE with test set vs predicted revneue
      sqrt(mean_squared_error(test, prediction['Predicted Revenue']))
```

```
[35]: 10.259252716755139
```

As we can see from above, our RMSE for our differences between the test set and predicted values is 10.25 indicating that we have roughly a 10 unit point error between the predictions our model made and the actual values from the test set. Considering our mean values for the entire data frame being just 14.17, I would say that this error metric is high which indicates that our model that does not predict well based on the training set, this is further verified by visually inspecting the predicted and test trendline.

As a recommended course of action for this analysis, it is clear that our prediction is way off from our test set and therefore more analysis may be needed. As shown in the annotated visualization of the next quarter, the data trend line does not predict well based on the training date and therefore I recommend getting a new training set to train the model.