Exploring the Seasonal ARIMA (SARIMA) Model for Forecasting: Differences from ARIMA

Jihyun (Jenny) Kim · Follow

6 min read · Apr 3, 2024

It's been about a week since the hectic Winter quarter ended, and I've finally had some time to review important concepts in machine learning and statistical modeling, applying them to my side projects. One such concept is the ARIMA model. Standing for AutoRegressive Integrated Moving Average, ARIMA is a forecasting technique that models time series data to analyze data points and predict future ones. It is widely used in finance, manufacturing, and economics to forecast future trends based on historical data.

ARIMA model?

First let's review what I've learned in last quarter. ARIMA models are characterized by three main parameters: p, d, and q, which are crucial to capturing the autocorrelation in time series data. To break down each component:

> ***AR (AutoRegressive), p:*** *This part of the model captures the relationship between a current observation and observations over previous periods. The parameter "p" denotes the order of the autoregressive part, which is the number of lag observations included in the model. Essentially, it assumes that the current value is a linear combination of the previous values.*
>
> ***I (Integrated), d:*** *Integration is a way to make the time series stationary, which is a critical step in time series forecasting. A stationary time series is one whose properties do not depend on the time at which the series is observed. Thus, it should exhibit constant mean and variance over time. The parameter "d" represents the degree of differencing required to make the series stationary.*

> *Differencing is the process of subtracting the current observation from the previous observation. If the time series is already stationary, then d = 0.*
>
> ***MA (Moving Average), q**: The moving average part models the error of the time series as a linear combination of error terms that occurred contemporaneously and at various times in the past. The parameter "q" specifies the order of the moving average part, which is the number of lagged forecast errors in the prediction equation. This component helps to smooth out the noise in the data.*

So, an ARIMA model is denoted as ARIMA(p,d,q) where:

> *p is the number of lag observations in the model (lag order).*
>
> *d is the degree of differencing (the number of times the data have had past values subtracted).*
>
> *q is the size of the moving average window (order of moving average).*

For the assignment, I utilized COVID-19 cases data to experiment with the ARIMA model, forecasting the last 41 days of COVID cases and comparing the predictions to the actual case numbers. This time, I've opted for a different dataset that I found on Kaggle: the Amtrak ridership dataset. The most significant difference between the COVID dataset and this one is the presence of seasonality. Also while we used R for this class, I decided to try Python for this project, as I'm more comfortable with it.

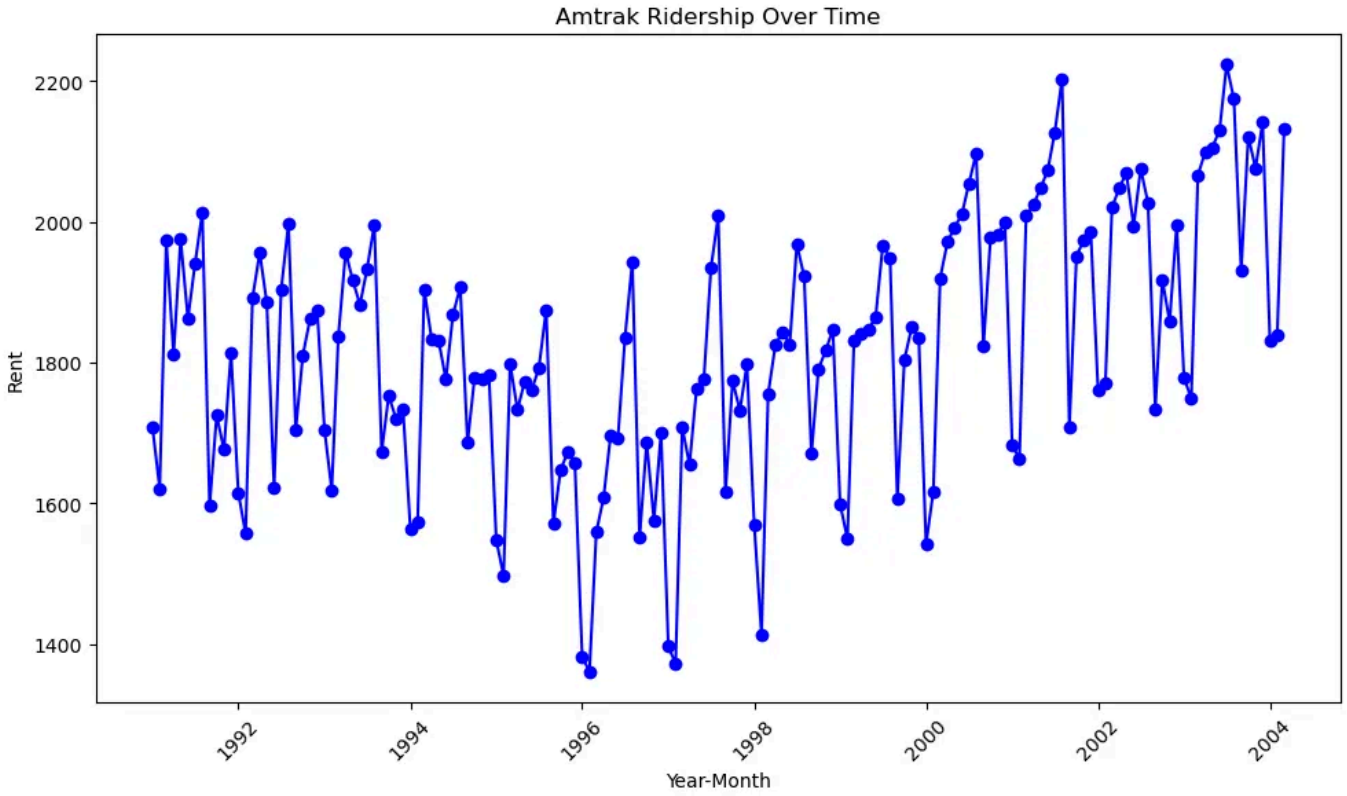
Dataset

First things first, let's examine the data. It is quite straightforward, containing four columns: Month, Ridership, t(stands for seasonal order,

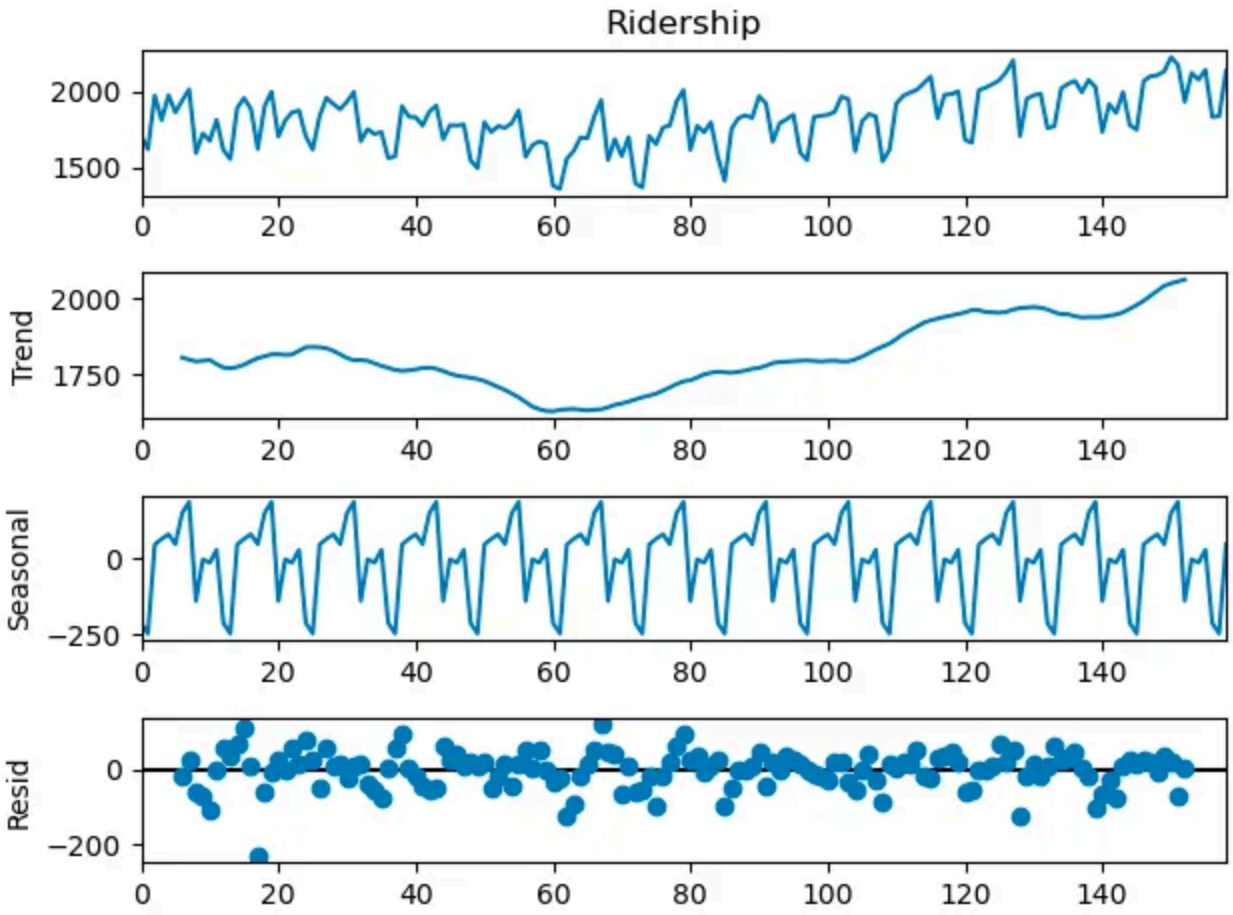
which we don't need now), and Season, with a total of 159 observations. The goal of this project is to forecast the number of Amtrak ridership for the last 12 months, so we will focus on the Month and Ridership columns.

Month	Ridership	t	Season
Jan-91	1709	1	Jan
Feb-91	1621	2	Feb
Mar-91	1973	3	Mar
Apr-91	1812	4	Apr
May-91	1975	5	May
Jun-91	1862	6	Jun
Jul-91	1940	7	Jul
Aug-91	2013	8	Aug

Now, let's create a line plot to check for any signs of seasonality within the dataset. Judging from the shape of the plot, I can subjectively observe some seasonal patterns.



In case the seasonality wasn't clear from the initial plot, I also decomposed the dataset, which can be easily done using the `seasonal_decompose` function in the `statsmodels` library. This decomposition confirmed the presence of seasonality in the Ridership numbers.



Fitting ARIMA model

Knowing that ARIMA model may not be the best approach when seasonality exists in the data, I just tried fitting ARIMA model for benchmark.

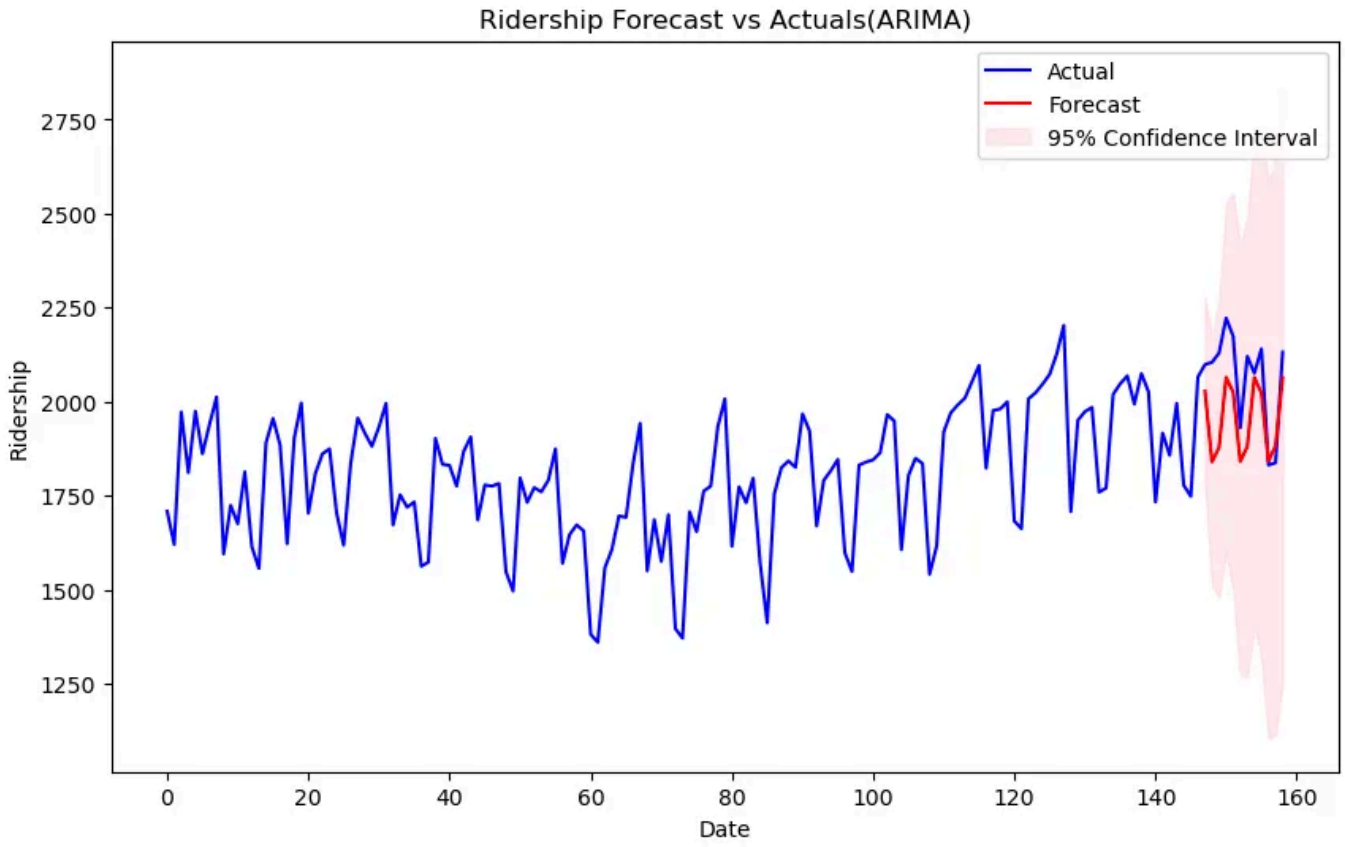
```python
train = df[:-12]
validation = df[-12:]

# Find the best p, d, q combination for the lowest AIC
p = d = q = range(0, 3)  # ranges for ARIMA parameters, 0 - 2

best_aic = float("inf")
best_params = None

for param in itertools.product(p, d, q):
    try:
        # ARIMA model does not use seasonal_order, so we exclude it
        model = ARIMA(train['Ridership'], order=param)
        results = model.fit()

        # Check if the current model AIC is lower than what we've seen before
        if results.aic < best_aic:
            best_aic = results.aic
            best_params = param

    except Exception as e:
        print(f"ARIMA{param} - AIC: None - Exception: {e}")
        continue  # Continue to the next iteration

# Print out the best parameters and AIC
if best_params is not None:
    print(f"Best ARIMA parameters: {best_params}")
    print(f"Best AIC: {best_aic}")
else:
    print("No suitable ARIMA model was found.")
```

✓ 0.7s

```
Best ARIMA parameters: (2, 1, 2)
Best AIC: 1843.7717475999127
```

Using AIC, obtained the best ARIMA parameters p,d,q = (2,1,2).

Let’s just jump to the good part:



Forecasted values vs. actual values, and the shaded area is 95% CI.

The results seem decent but not entirely satisfactory. It feels like there could be room for improvement — Seasonal ARIMA model.

SARIMA model

So what's SARIMA model? The Seasonal ARIMA (SARIMA) model is an extension of the ARIMA model, designed to better handle time series data with seasonal patterns. SARIMA is particularly useful for forecasting when data exhibit regular, predictable changes that recur every calendar cycle, such as daily, monthly, or quarterly fluctuations (in short, seasonality). The model is capable of capturing both the trends and seasonality in data to make accurate predictions. To break down each component:

$$SARIMA \underbrace{(p,d,q)}_{non-seasonal} \underbrace{(P,D,Q)_m}_{seasonal}$$

> ***AR (AutoRegressive), p****: The number of lag observations included in the model. It captures the relationship between an observation and a number of lagged observations.*
>
> ***I (Integrated), d****: The number of times the observations are differenced to make the series stationary, meaning the series does not show trends or seasonality and has a constant variance over time.*
>
> ***MA (Moving Average), q****: The size of the moving average window. It models the error term as a linear combination of the error terms at various times in the past.*
>
> ***Seasonal components (*****P,D,Q*****)****: These are similar to the non-seasonal components but are applied to the seasonal component of the series.* P, D*, and* Q *represent the seasonal autoregressive order, degree of seasonal differencing, and seasonal moving average order, respectively.*
>
> ***m****: The length of the seasonal period (e.g., 12 for monthly data with an annual cycle).*

The SARIMA model can be fitted using the `SARIMAX` class in the `statsmodels` library in Python.

```python
train = df[:-12]
validation = df[-12:]

# Find the best p,d,q combination gives lowest AIC
p = d = q = range(0, 3)  # ranges for ARIMA parameters, 0 - 2
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))] # 12 for 12 months

best_aic = float("inf")
best_params = None
best_seasonal_params = None

for param in itertools.product(p, d, q):
    for seasonal_param in seasonal_pdq:
        try:
            model = SARIMAX(train['Ridership'],
                            order=param,
                            seasonal_order=seasonal_param,
                            enforce_stationarity=False,
                            enforce_invertibility=False)
            results = model.fit(maxiter=200, disp=0)  # Set disp=0 to reduce output verbosity
        except Exception as e:
            print(f"ARIMA{param}x{seasonal_param} - AIC: None - Exception: {e}")
            continue  # Continue to the next iteration

        # Check if the current model AIC is lower than what we've seen before
        if results.aic < best_aic:
            best_aic = results.aic
            best_params = param
            best_seasonal_params = seasonal_param

# Print out the best parameters and AIC
if best_params is not None:
    print(f"Best SARIMA parameters: {best_params}")
    print(f"Best Seasonal parameters: {best_seasonal_params}")
    print(f"Best AIC: {best_aic}")
else:
    print("No suitable model was found.")
```

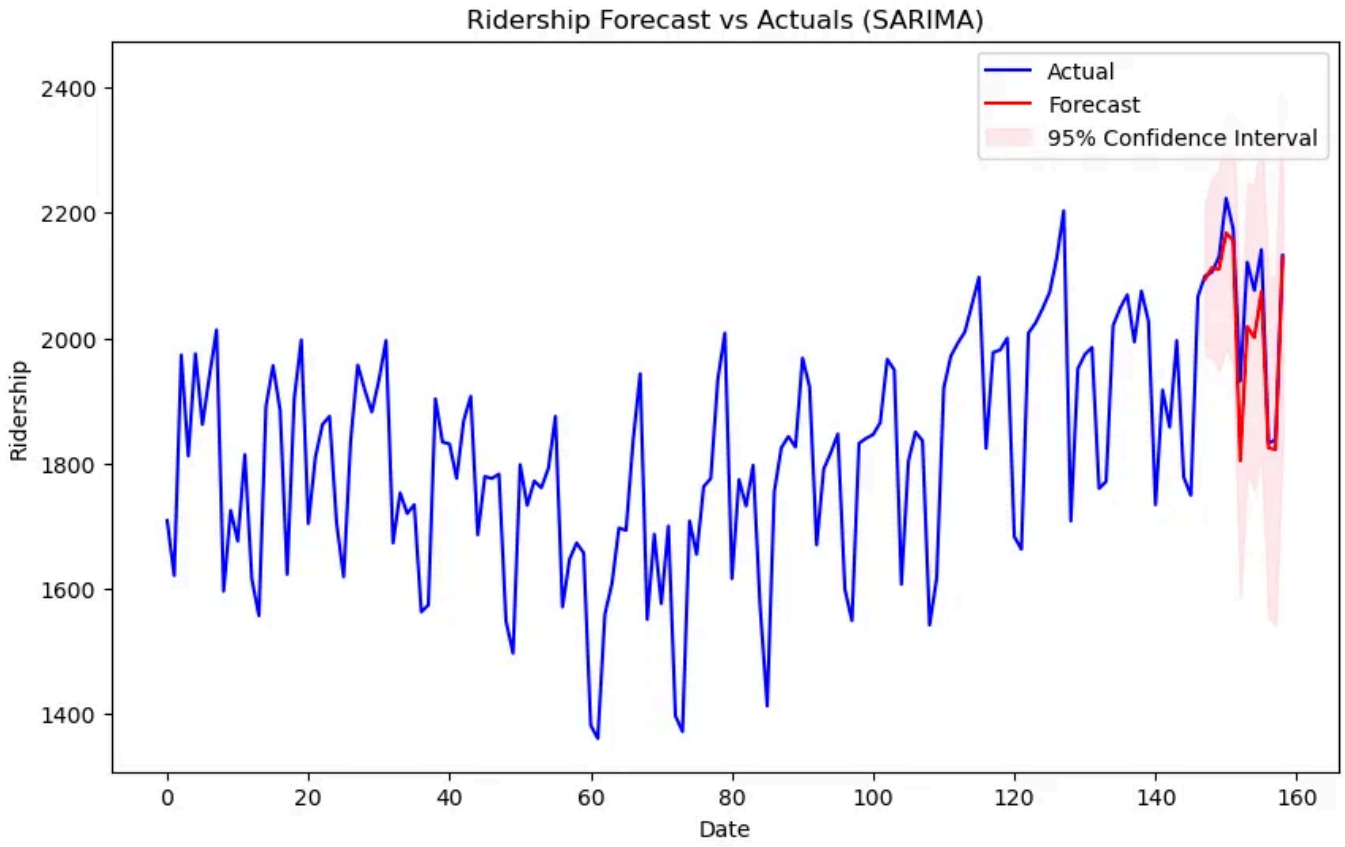
✓ 9m 35.1s

```
Best SARIMA parameters: (0, 2, 2)
Best Seasonal parameters: (0, 2, 2, 12)
Best AIC: 1070.9745724402078
```

Same with the ARIMA model, obtained the best parameters using AIC.

Using a for loop to iterate over all combinations of p, d, and q parameters, I identified the best combination with the lowest AIC:

> $(p,d,q)\ (P,D,Q)m = (0,2,2)\ (0,2,2)_{12}$



Oh wow. The previous ARIMA model performed okay, but the results from the SARIMA model are noticeably better. The forecasted values (represented by the red line) align more closely with the actual values (represented by the blue line).

I evaluated both models quantitatively by calculating the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) to determine which model performs better:

ARIMA model

- Mean Absolute Error: 122.77
- Root Mean Squared Error: 150.22

SARIMA model

- Mean Absolute Error: 58.58
- Root Mean Squared Error: 73.99

ARIMA model's MAE and RMSE are much greater than those of SARIMA model. Therefore, the SARIMA model, which accounts for both seasonality and non-seasonality components, seems to have a better predictive accuracy for your data than the ARIMA model, which does not account for seasonality.

Conclusion

To wrap up, our journey into time series forecasting has vividly showcased the superior performance of the Seasonal ARIMA (SARIMA) model over the standard ARIMA model, especially for data with seasonal trends. Through applying both models to the Amtrak ridership dataset, it's evident that SARIMA excels in accounting for both seasonal fluctuations and non-seasonal dynamics, yielding more precise forecasts. This project vividly underscores the importance of selecting a model that aligns closely with the specific characteristics of the data, particularly crucial when navigating the complexities of seasonal time series forecasting.

✢ The code is uploaded HERE on my GitHub repository in .ipynb format.

✢ Also my LinkedIN profile: https://www.linkedin.com/in/jihyun-kim423/

Arima | Sarima | Time Series Forecasting | Forecasting | Seasonality