Time Series for Web Traffic Forecasting

Benjamin Santana Velazquez

***Abstract*—In recent years, the growth of online platforms has driven the need for accurate and reliable web traffic forecasting techniques. This paper compares different naive models against the application of the Auto-Regressive Integrated Moving Average ($ARIMA$) model to forecast web traffic for chess.com.**

We first dive into a background study on how $ARMA(p, q)$ works and the core ideas behind it. We present how to choose the correct order of p and q using *Akaike Information Criterion* (AIC). Then we find the right model for our specific case.

We then showcase how $ARIMA(0, 1, 5)$ outperforms other naive models as well as $ARIMA(3, 1, 1)$. Implying that the traffic of chess.com can be best modeled by a $MA(q)$ process.

Overall, this paper showcases the application of the $ARIMA(p, d, q)$ model to forecast web traffic for chess.com. It contributes to the existing literature on web traffic forecasting and demonstrates the potential of time series analysis techniques in capturing the dynamic nature of online user behavior.

***Index Terms*—Forecasting Time Series ARIMA**

I. Introduction

Very few events are unaffected by time, this offers a compelling rationale behind the significance of *time series analysis* within the realm of data science. In today's data-driven world companies hoard large amounts of data over time, which contain important information about their business patterns. Consequently, time series analysis has permeated various fields, from the traditional ones like finance and economy — where someone can forecast sales or the EPS of a public company — to lesser explored territories like marketing, where time series analysis allows to track business metrics, such as user engagement and create forecasts based on that. In this piece of work we will delve into the realm of web traffic.

It is common for web applications to keep a register of their traffic at all times. Meaning, if you are the owner of a website it is likely that you have access to the data sent and received by your visitors. What is more, you probably have access to the Click-stream Data of your users, which is the record of an individual's clicks through their journey on the Internet (in this case, your website).

Taking advantage of this information, and the power of forecasting with statistical models in time series, we can forecast the traffic of a website.

Why would we want to do this? *Resource Allocation* and *Performance Optimization*. By forecasting website traffic, we can estimate the expected number of visitors and allocate appropriate resources to handle the load. This can either save costs in the infrastructure of the website like server capacity and bandwidth — when a low number of visitors is present — or it can save the website's users from seeing an overload error when the server's capacity is at peak.

Recently, a popular website by the name of chess.com had this issue. "Chess Is Booming! And Our Servers Are Struggling." [1] was the title of an article posted after days of having their servers at max capacity, not allowing users to play games and losing revenue in the process.

This begs the question: Can we accurately forecast the traffic of the website chess.com to make proactive resource allocation? This would ensure that, at any given point, we have enough resources asigned to handle the traffic load. I will do my best to try and answer this in this short paper.

We will use one of the most popular traditional statistical methods used in time series forecasting, $SARIMA(p, d, q)(P, D, Q)_m$. We will find out if this model is powerful enough to create meaningful and accurate forecasts or if we should opt for naive methods to get better results. [1]

Unfortunately, the traffic of a website is not public information. Meaning we cannot get the actual time series from chess.com itself. We need to rely on other sources to get the actual information. After reviewing the possible options we deiced to go with Semrush, a "all-in-one tool suite for improving online visibility and discovering marketing insights." [3]. One of Semrush tools provides web traffic for different websites. Of course we need to take into account that we are taking the data (the time series itself) from a third party, so there is some level of uncertainty associated with its accuracy. Semrush provided us with the historic information month by month of traffic in chess.com from January 2012 to March 2023. How it looks will be discussed later in the paper. We will try to forecast 3 months into the future given these data points.

In summary, our objective is to assess the effectiveness of the $SARIMA(p, d, q)(P, D, Q)_m$ model in forecasting three months of web traffic on chess.com. We will be conducting an analysis by comparing its performance against naive forecasting methods. Such forecasts play a crucial role in facilitating both resource allocation and performance optimization strategies for the website. The conclusion an results are shown in Section IV and Section V.

II. Background Study

In this section we will define what a time series is and how it is composed. We are also going to discuss the $MA(q)$ model and the $AR(p)$ model, which serve as a base for $SARIMA(p, d, q)(P, D, Q)_m$. Understanding these two parts of the model — I believe — is the keystone to understanding the model as a whole. We are also going to review how $AR(p)$ and $MA(q)$ come together in $ARMA(p, q)$ to model complex time series, and how we can find the orders p and q.

[1] We could also opt to get into the realm of neural network for time series, but this is not covered in the paper, if you are interested please refer to [2]

What is a time series? We can define a time series as a set of points ordered in time, usually equally spaced in time. [4] We can decompose any time series into 3 components:

- *trend*, the slow-moving change in the time series.
- *seasonal component*, a cycle that occurs over a fixed period of time.
- *residuals*, what can not be explained by the trend of seasonal components, this is usually white noise. A model wont be able to forecast any of this part, since it is completely random

To demonstrate how this looks in a time series we will use a classical time series presented in *Box, G. E. P., Time Series Analysis, Forecasting and Control.* [5]. The classic Box & Jenkins airline data presents monthly totals of international airline passengers, 1949 to 1960. Table I, showcase the first five entries of the time series.

TABLE I
FIRST FIVE ROWS OF THE CLASSIC BOX & JENKINS AIRLINE DATA

Month	Passengers
1949-01	112
1949-02	118
1949-03	132
1949-04	129
1949-05	121

If we plot the series, with the time in the x axis and the passengers in the y axis, we get something like Figure 1. This is what we call observed data.

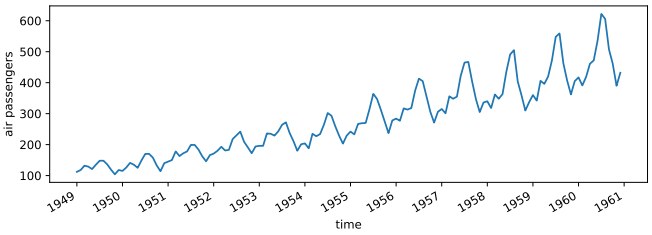

Fig. 1. The classic Box & Jenkins airline data, it presents monthly totals of international airline passengers, 1949 to 1960.

Let us decompose it into the three components, to see how it would each would look in a plot. Figure 2 showcases how this looks.

We can see that we have decomposed the time series into 4 plots. The first one is the observed data, just like we saw in Figure 1. Then we can see one plot for each of the concepts explained above. We see the trend as a slow-change moving upwards. We can see the seasonal component repeating each year, and finally the white noise or residuals of the time series.

Hopefully after seeing this example this concepts have become clearer to the reader.

By now you might be wondering what are the differences between time series forecasting and a other regression tasks. It basically comes down to two things.

- *Time series have an order*, a time series is indexed by time, order must be kept. Say for example, in a regression tasks when you want to predict revenue based on ad spend, it does not matter when certain amount was spent.

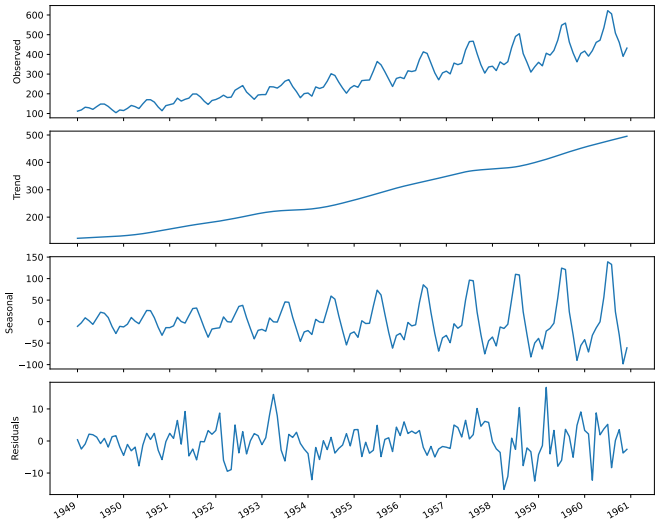

Fig. 2. Decomposition of The classic Box & Jenkins airline data, it presents monthly totals of international airline passengers, 1949 to 1960.

- *Time series do not have features*, it is common to only have two columns, time and the data itself. It usually does not have categories.

I will explain more concepts before diving into $MA(q)$ and $AR(p)$ starting with *baseline models*. A baseline model is a trivial solution to our problem, it uses heuristics more than deep statistics knowledge. It might be clearer if we use an example. Say we have a time series on the sales of XYZ, after getting the data points we calculate the mean and say "My forecast for next month is that our sales will be the mean of all the time series". We got this forecast using a simple statistics, the mean. It is likely that we can get a better forecast than this using more complex statistics. Why then should we bother making this naive models? They allow us to compare our complex model with something. A common example of a baseline model is just simply using the last known value. For example, if we have five years worth of data points, collected each month, and we want to forecast next three months, we can simply copy and paste the values from last three months and be good to go. In some time series, our statistical model may perform worse than this naive methods, then we would be wasting resources every time we make a forecast running our complex statistical models. Because we could just copy and paste the last values, or use the mean, or some other baseline model and get beeter results. In Section IV we will compare our $SARIMA(p,d,q)(P,D,Q)_m$ model with some baseline models and conclude if it is worth using it against simpler models.

One last thing on the baseline methods. How can we compare the models against each other? We will calculate an error metric in order to evaluate the performance of our forecasts. We will use $MAPE$ (mean absolute percentage error), which will measure prediction accuracy, independent of the scale of our data

The next concept is *stationarity*, a stationary time series is one whose statistical properties do not change over time It

has constant mean, variance, and autocorrelation, and these properties are independent of time [4]. Many models assume stationarity but we rarely see stationarity in time series. Lucky for us we can transform the time series to become stationary, there are many ways to transform it but the simplest (and the one we are going to use) is *differencing*. This consists on calculate the changes from one time step to another ($y'_t = y_t - y_{t-1}$). This transformation helps stabilize the mean. Which reduces the trend and seasonality effects. This means that we also need to make sure we do an inverse transform of the data after finishing the model.

We can test for stationarity with the *Augmented Dickey-Fuller* (ADF) test [6], The key idea in the ADF test is to estimate an autoregressive model of the time series and examine the significance of the coefficient on the lagged first difference term. If the coefficient is significantly different from zero, it suggests evidence against the presence of a unit root and supports stationarity. [2]

Basically the null hypothesis states that a unit root is present, meaning that our time series is not stationary. We can reject the null hypothesis if after doing the test, we get a p value less than 0.05.

I think we have a good enough grasp on time series concepts to move into the moving average process ($MA(q)$) and the autoregressive process ($AR(p)$).

A. Moving Average Process

In a moving average (MA) process, the current value depends linearly on the mean of the series, the current error term, and past error terms. [4]. It is usually denoted as $MA(q)$ where q is the order, meaning the number of past error terms that affect the present value).

The moving average of order q is calculated by equation 1.

$$y_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_q \epsilon_{t-q} \quad (1)$$

Where, μ is the mean, ϵ_t is the error term, ϵ_{t-k} is the past error term and θ is the magnitude of impact of the past errors.

Needless to say, the larger q is, the more past error terms affect the present value. This means that we need to choose this hyperparemeter carefully in order to train the data correctly. In simple time series we can use simply use the AutoCorrelation Function (ACF) plot in order to get q. An example of such plot is shown in Figure 3.

In this plot we can see a relevant correlation up to lag 2, meaning we can use $MA(2)$ to model this stationary time series process.

B. Autoregressive Process

An autoregressive process ($AR(p)$) establishes that the output variable depends linearly on its own previous states. An autoregressive process is a regression of a variable against itself. In a time series, this means that the present value is linearly dependent on its past values. [4]

[2]if you want more information on how this work I highly recommend reading the paper where it was proposed [6]

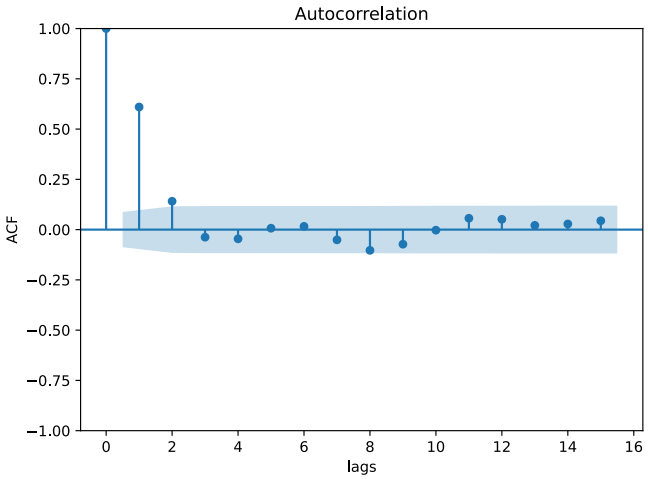

Fig. 3. Example ACF plot, where we see a relevant correlation up to lag 2

It is defined by $AR(p)$ where p defines the number of past values that affect the present value. Equation 2 show an autoregressive process up to lag p.

$$y_t = C + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon_t \quad (2)$$

Where ϕ is the autoregressive coefficients that measure the strength of the relationship between the current value and past values, ϵ_t is the error term and ϵ_{t-k} the past error term.

Like in subsection II-A, we also have an important hyperparemeter here: p. Unfortunately we can not get it from the ACF plot, because if we did an the plot on autoregressive stationary process it would exhibit a pattern of exponential decay [4] like seen in Figure 4.

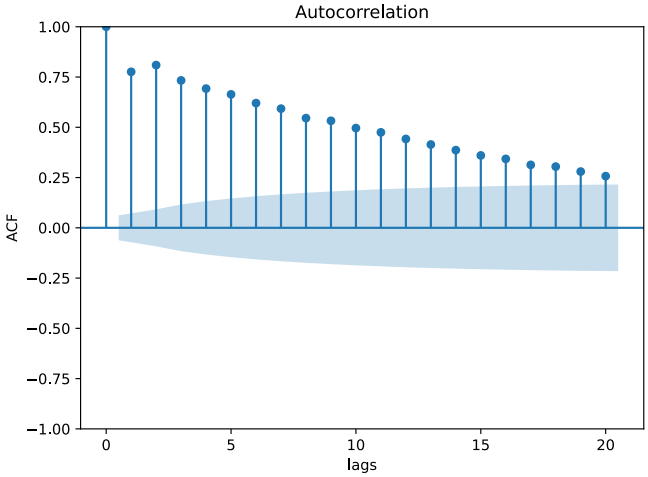

Fig. 4. Example ACF plot on an autoregressive process

We then must turn our attention to the Partial AutoCorrelation Function (PACF) plot to get the order of the autoregression. This measures the correlation between lagged values in a time series when we remove the influence of correlated lagged values in between. Figure 5 showcase how this would look.

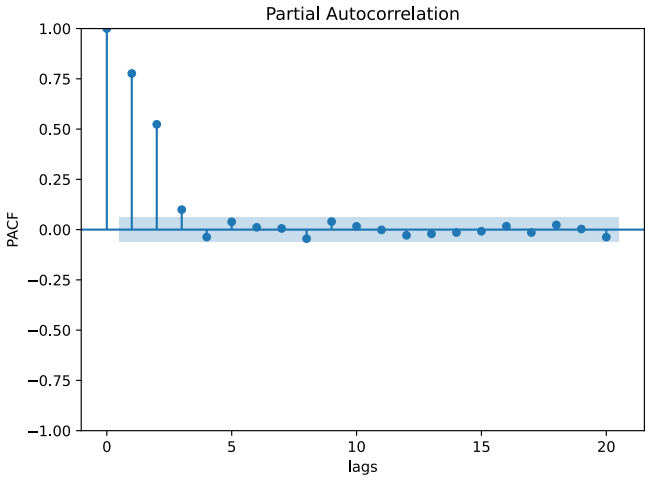

Fig. 5. Example PACF plot, we can see that we have no significant coefficients after lag 3, meaning $AR(p)$ where $p = 3$, we have an autoregressive process of order 3

We can see in this example plot that we have no significant coefficients after lag 3, therefore we can say that this process is $AR(3)$, meaning we have an autoregressive process of order 3.

C. Complex Time Series

We have seen $MA(q)$ and $AR(p)$, but the curious reader (or one with a background on time series analysis), will see that something is missing. What happens when the ACF plot shows a slowly decaying pattern (like seen in Figure 4) or a sinusoidal pattern, and, when plotting PACF you see too, a slowly decaying pattern or a sinusoidal pattern. Meaning we cannot infer an order from the ACF plot or from the PACF plot. Then, we may be in the presence of a complex process, where we will need to combine $AR(p)$ and $MA(q)$ to model it, this is where the $ARMA(p, q)$ model comes into play.

The $ARMA(p, q)$ is simply a combination of the models we saw in subsection II-A and subsection II-B. If we see how we can model it, you will see that we basically have the two equations plugged together. See equation 3 where:

$$\begin{aligned} y_t = C + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon_t \\ + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_q \epsilon_{t-q} \end{aligned} \tag{3}$$

- p determines the number of past *values* that affect the present value.
- q determines the number of past *errors* that affect the present value.

It is worth mentioning that $ARMA(0, q)$ process is equivalent to an $MA(q)$ process, since the order $p = 0$ cancels the $AR(p)$ portion. An $ARMA(p, 0)$ process is equivalent to an $AR(p)$ process, since the order $q = 0$ cancels the $MA(q)$ portion.

This still does not solve the issue I stated above, even if we combine both models, we wont be able to get the order (p and q) using the ACF and PACF plot. Luckily for us we can use the *Akaike Information Criterion* (AIC).

D. Akaike Information Criterion

The AIC estimates the quality of a model relative to other models. Given that there will be some information lost when a model is fitted to the data, the AIC quantifies the information lost. The less information lost, the lower the AIC value and the better the model.

Equation 4 shows how to calculate AIC.

$$AIC = 2k - ln(\hat{L}) \tag{4}$$

Where, k is the number of estimated parameters and $\hat{L}$ is the maximum value of the like-hood function model.

Using AIC to select our model, allows us to keep a balance between the complexity of the model and its goodness on fitting the data. This is because k is directly affected by the order of p and q in $ARMA(p, q)$. Let us see an example; say $p = 2$ and $q = 2$ then $k = p + q = 4$, since we are subtracting the likelihood function model to times two this number, the higher the order gets, the AIC increases, penalizing more complex models.

$\hat{L}$ (the likelihood function) measures the goodness of fit a model has. We can compare it with the distribution function to make it easier to understand becuase it can be seen as the opposite of the distribution function. The distribution function, given a model with fixed parameters, measures the probability of observing a data point. The likelihood function, given a set of data, tells us how likely it is that different model parameters generate the data.

Let us see an example from the book Time Series Forecasting by Marco Peixeiro [4], that helped me understand the concept better:

Say we have a six-sided die.

- The distribution function tells us that there is a $\frac{1}{6}$ probability that we'll observe $1, 2, 3, 4, 5, 6$
- Suppose that we roll the die 10 times an you obtain $[1, 5, 3, 4, 6, 2, 4, 3, 2, 1]$ The likelihood function will determine how likely is that the die has six sides.

So the question in our context is, *How likely is that my observed data is coming from an $ARMA(n, m)$ model?*, where n, m represent two natural numbers. If a model fits the data really well, the maximum value of likelihood will be high [4]. Since AIC function subtracts the $ln(\hat{L})$ from $2k$, it balance between under-fitting and over-fitting.

We need to take into account that AIC quantifies the quality of a model in relation to other models only. Meaning it is a relative measure of quality. What we end up doing in practice is iterating through some combinations of p and q, to calculate the AIC values and selecting the one with the lowest score, finding this way the order of $ARMA(p, q)$. We can extend this process to even more complex models like $SARIMA(p, d, q)(P, D, Q)_m$, AIC will help us find this hyperparameters.

Coming back to our initial problem, after finding a combination of p and q, we still need to check one more thing.

1) Calculating Residuals: The residuals of a model are simply the difference between the predicted values and the actual values [4]. If the process found by the $ARMA(p, q)$ model is the same as the original process then the residuals

would be just ϵ_t (the error terms), meaning white noise, we cannot predict these values based on past values since they are complete random. In summary, we want our residuals to behave like white noise (completely random).

We will need to check for two things, *Q-Q plot* and the *Ljung-Box test*.

The quantile-quantile *Q-Q* plot is a graphical technique for determining if two data sets come from populations with a common distribution. This is a plot of the quantiles of the first data set against the quantiles of the second data set. By a quantile, we mean the fraction (or percent) of points below the given value. [7]

Basically we are going to plot the distribution of our residuals in one axis, and a normal distribution in the other axis. If $y = x$ this means that our residuals behave exactly like a normal distribution, meaning they are white noise/completely random, so the order we chose (for p and q) is good.

Luckily for us, the python module *statsmodels* [8], allows us to do diagnostic plots for our residuals in a really easy way.

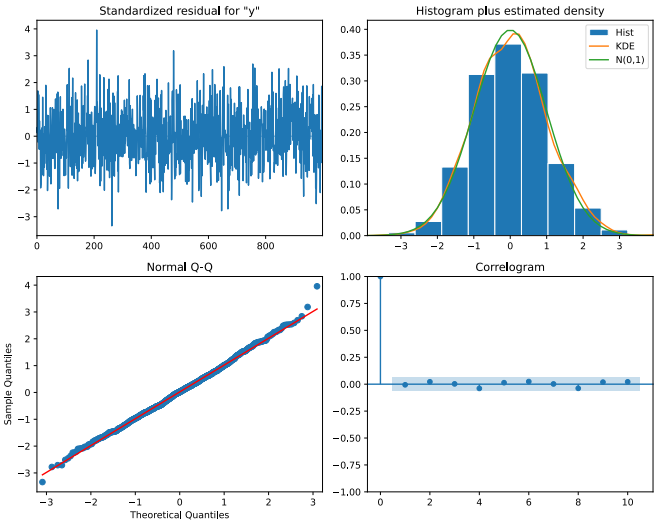

Fig. 6. Example Diagnostic Plots for the residuals of a $ARMA(1,1)$ model

Please refer to Figure 6. In the top left we find a plot of the residuals, we can see here that they are stationary, and they do not have a trend. This is good, because that is the behavior of a normal distribution. In the top right, we can see a histogram of the residuals, it looks like a normal distribution. In the bottom right, we can see the autocorrelation function of the residuals, as we can see there is no significant lag after lag 0, therefore there is no apparent autocorrelation.

Lastly on the bottom left, we found the Q-Q plot, as stated above, it is comparing the distribution of our residuals with a normal distribution. We can see that in almost all instances $y = x$ meaning our residuals behave like a normal distribution, therefore our residuals most be white noise (completely random) which is what we expected.

After having a good Q-Q plot we go to Ljung-Box test to demonstrate that the residuals are uncorrelated.

The Ljung-Box test is a statistical test that determines whether the autocorrelation of a group of data is significantly different from 0. The null hypothesis states that the data is independently distributed, meaning that there is no autocorrelation. If the p-value is larger than 0.05, we cannot reject the null hypothesis meaning that the residuals are independently distributed. [4]

Basically if in the Q-Q plot $y = x$ and when doing the Ljung-Box test we get on every lag $p > 0.05$ we can start using the model for forecasting.

E. Brief Summary

In this section we saw basic concepts of time series, like decomposition and stationarity. We learned about the $MA(q)$ and $AR(p)$ models, and how they work together to form $ARMA(p,q)$ and model complex time series. We saw how to use AIC to find the order of p and q. We also saw how to analyze the residuals to diagnose the model and see if we can use it in forecasting. It goes without saying that this of course does not cover every single concept related to $SARIMA(p,d,q)(P,D,Q)_m$. The idea of this section is to give an understanding of the core concepts behind $SARIMA(p,d,q)(P,D,Q)_m$, to help the reader follow the process we take in Section III. I encourage the reader to refer to the references we used throughout this section for a complete understanding of $SARIMA(p,d,q)(P,D,Q)_m$.

III. Experimental Framework

This section serves as the foundation for evaluating the performance of the $SARIMA(p,d,q)(P,D,Q)_m$ model with chess.com web traffic. The section outlines the model configuration, meaning how we chose the hyperparameters and how good the proposed approach works.

As stated in Section I, our objective is to assess the effectiveness of the $SARIMA(p,d,q)(P,D,Q)_m$ model in forecasting three months of web traffic on chess.com. We will be conducting an analysis by comparing its performance against naive forecasting methods. To be more specific we will compare its performance, against two *naive methods* (also known as baseline models), *Last Known Value* and *Window Average*.

Since we are trying to forecast 3 months into the future, for our first naive method (*Last Known Value*), we will take the values of the last 3 months and use them as a forecast. For our second baseline model (*Window Average*) we will take a window of time, calculate the average and use that as a forecast. In this case, that window of time will be the the last 6 months. We will compare the models using the $MAPE$ function discussed in Section II.

In Section I we stated the traffic of a website is not public information. We got the information from a third-party Semrush. Semrush tools provides web traffic for different websites. We need to take into account that we are taking the data from a third party, so there is a some level of uncertainty associated with its accuracy. Semrush provided us with the historic information month by month of traffic in chess.com from January 2012 to March 2023. Since we are planning to forecast three months into the future, we will split our data, in sets, one for training and one for testing. We will take

the information from January 2012 up to December 2022 and make that our training set. While the first three months of 2023 will serve as testing data. Meaning we will compare our forecast with the actual information of these 3 months.

Let us start by outlining in general terms the process we will follow.

1) Gather Data Points
2) Decompose Time series
3) Check for Stationarity
4) Fit every combination of $SARIMA(p,d,q)(P,D,Q)_m$
5) Select model with lowest AIC [3]
6) Diagnose residuals
7) Make forecast

A. Gather Data Points

We downloaded the time series as a csv file, meaning it was easy to manipulate using python. The first entries of the time series, look something like Table IV

TABLE II
TRAFFIC DATA

Index	Date	Traffic
0	2012-01	526,370
1	2012-02	489,447
2	2012-03	482,083
3	2012-04	497,484
4	2012-05	531,260

We can see that the entries are collected monthly and on further inspection we have no missing values. If we plot the date against the traffic we get something like Figure 7

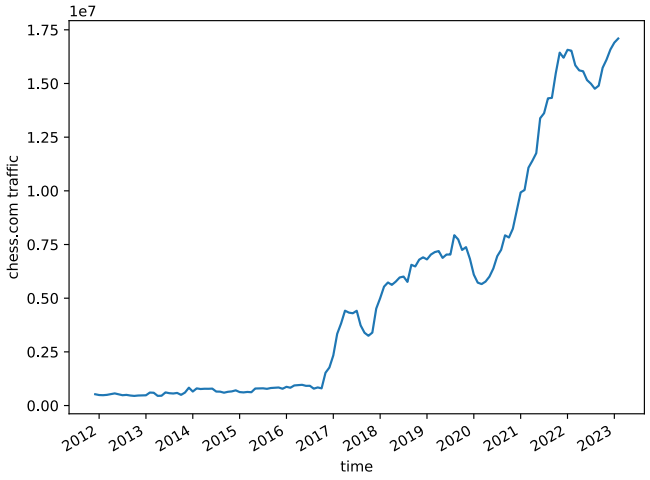

Fig. 7. Observed chess.com traffic from January 2012 to March 2023

We can see a big spike on 2017, and an almost constant trend upward. For this reason we will drop the values from before December 2016, since taking these items into account will make our model forecast worse. We are left with 76 data points, which should be enough for our forecast.

As mentioned above, we will split the dataset, into training an testing dataset. We need to take the last 3 months off from our dataset to get the actual training data points. In Figure 8 we can see the dataset once we have done this two changes.

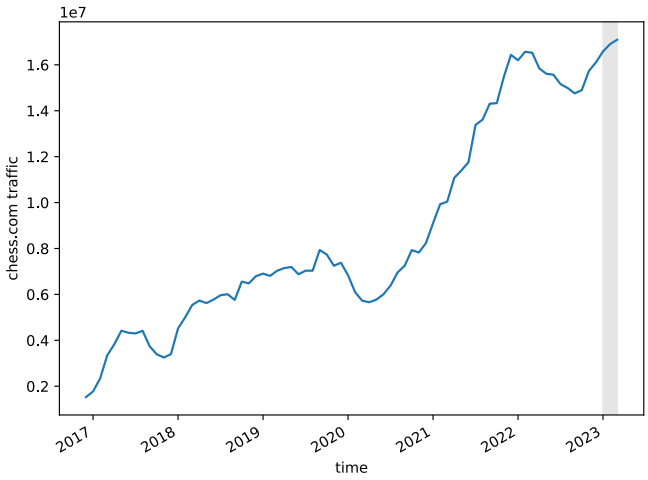

Fig. 8. Observed chess.com traffic from December 2016 to March 2023, with testing points grayed out

B. Decompose Time series

Now that we have our time series ready let us analyze the decomposition of the time series. [4] Please refer to Figure 9

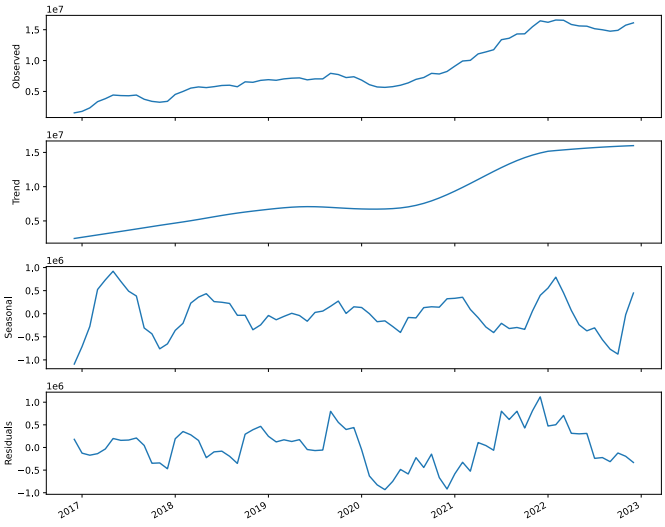

Fig. 9. Decomposed chess.com traffic from December 2016 to December 2023

In the first plot, top to bottom, we can see the observed data, which is the same as in Figure 8. Next plot we can see that the data has mostly a slow moving change upwards, meaning it has a positive trend so most likely we are not dealing with a stationary process. Moving one plot down we will see the seasonality of the data, it does not seem to have any, there are no apparent cycle that occurs over a fixed period of time. If this is hard to see, compare Figure 2 with Figure 9. In Figure 2

[3] Refer to section II-D if you do not know what AIC is.

[4] If you do not know what decomposing the time series means please refer to Section II or Figure 2

we can see a clear cycle that occurs every year, in Figure 9 we see nothing similar. This means that it is likely our process does not has seasonality. Meaning all the part of Seasonality in $SARIMA(p,d,q)(P,D,Q)_m$ can be discarded, we would end up with $ARIMA(p,d,q)$. This reduces the combinations of hyperparameters we need to calculate.[5].

C. Check for Stationarity

We now know that it is likely that our time series is not stationary, nevertheless let us run our *Augmented Dickey-Fuller* (ADF) test. [6]. The ADF statistic gave us -0.34789 and $p = 0.91844$ this means $p > 0.05$, therefore we cannot reject the null hypothesis, and the series is not stationary. We use differencing to make it stationary. After differencing it once we get a new ADF statistic of -3.18885 and $p = 0.02063$ this means $p < 0.05$, therefore we can reject the null hypothesis and say that the series is stationary after calculate the first discrete difference. This means we have our first hyperparemeter for $ARIMA(p,d,q)$, $d = 1$ [7].

D. Fit every combination of $SARIMA(p,d,q)(P,D,Q)_m$

We have stated above that our model is $SARIMA(p,d,q)(P,D,Q)_m$ but with the seasonality part set to zeros. Therefore we only need to find the combination of the p and q orders. This becuase $P = 0$, $D = 0$, $Q = 0$ since the time series has no seasonality. We will try all the combinations generated by the Cartesian product of $A \times A$ where A is a set defined as $A = \{0,1,2,3,4,5,6,7,8,9\}$. This means we will try $ARIMA(1,1,1)$, then $ARIMA(1,1,2)$ up to $ARIMA(1,1,9)$, to then go $ARIMA(2,1,1)$ and try all combinations. We will test each model with the AIC go to Section II-D for more information on how it works. After doing this, we record the entries into a table, of which the first five entries are shown in Table III.

TABLE III
AIC VALUES

Index	(p,q)	AIC
0	(3, 1)	2078.407462
1	(0, 5)	2079.774906
2	(4, 1)	2080.024761
3	(3, 2)	2080.353652
4	(2, 3)	2080.360632
5	(0, 6)	2081.280308
6	(1, 5)	2081.292037

E. Select Model with Lowest AIC

We can see that our top choice is for $p = 3$ and $q = 1$. The second option, is quite interesting it says $p = 0$ and $q = 5$, meaning a $MA(5)$ process might model the time series if we make it stationary. This becuase if $p = 0$ we can take all the $AR(p)$ part of the process. It is worth exploring the top two combinations and compare then against each other, see which one performs better.

F. Diagnose Residuals

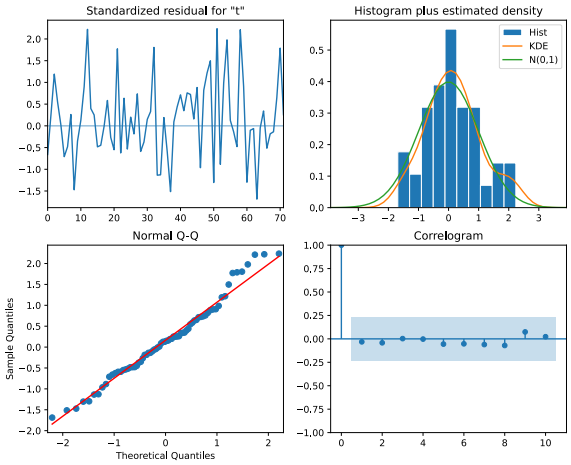

Fig. 10. Diagnostics plots for $ARIMA(3,1,1)$ residuals

1) $ARIMA(3,1,1)$: Please refer to Figure 10 while reading this subsection. In the top left plot we can see that when plotted, the residuals seem to be stationary, this is a good sign, becuase we do not want our residuals to have any trend. On the top right plot we can see that the residuals behave similar to a normal distribution. In the bottom right plot (Q-Q plot) we see that in almost every point y is similar to x, meaning the distribution of our residuals and a normal distribution are similar. Finally in the bottom right plot, we see that the residuals do no seem to have any correlation. So everything points that our residuals behave like white noise, which is the result we want.

When doing the Ljung-Box test, discussed in Section II-D1, we saw that at any given lag $p > 0.05$ meaning we cannot reject the null hypothesis, therefore we can say that no lag is correlated.

2) $ARIMA(0,1,5)$: Please refer to Figure 11 while reading this subsection.

The analysis for Figure 10 is pretty much the same as the one presented in Subsection III-F1 the only difference is on the Q-Q plot, but it is not significant so I wont spend time discussing it.

Similar to $ARIMA(3,1,1)$ when doing the Ljung-Box test, discussed in Section II-D1, we saw that at any given lag $p > 0.05$ meaning we cannot reject the null hypothesis, therefore we can say that no lag is correlated.

G. Forecast

We have defined the models we will use to make the forecast $ARIMA(3,1,1)$ and $ARIMA(0,1,5)$. Now we will insert the forecasts in a table side by side with the testing set (the

[5]Please refer to subsection II-D to learn more about how we find the hyperparameters ($ARIMA(p,d,q)$ order)

[6]To learn more about the *ADF* test please go back to Section II or to [6]

[7]d is directly related to the number of times we have to apply a differentiation to make the series stationary, if after applying the first differentiation $p > 0.05$ we would have to differentiate again and d would become 2

TABLE IV
TRAFFIC DATA

Index	Date	Traffic	Last Known Value	Window Average	$ARIMA(3,1,1)$	$ARIMA(0,1,5)$
73	2023-01	16,583,965	14,898,733	15,274,165	16,412,187	16,381,638
74	2023-02	16,902,916	15,730,408	15,274,165	16,707,268	16,770,711
75	2023-03	17,094,718	16,110,785	15,274,165	16,804,804	16,971,533

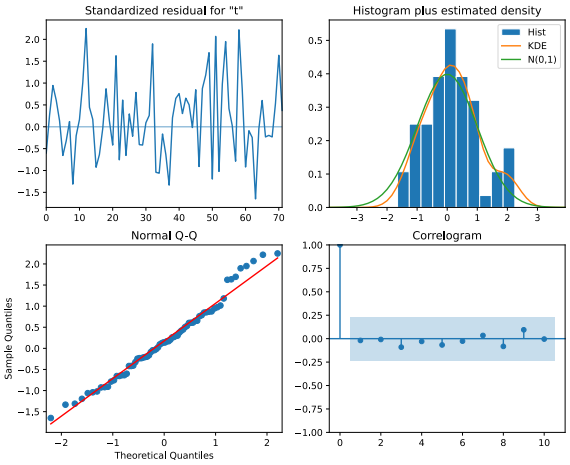

Fig. 11. Diagnostics plots for $ARIMA(0,1,5)$ residuals

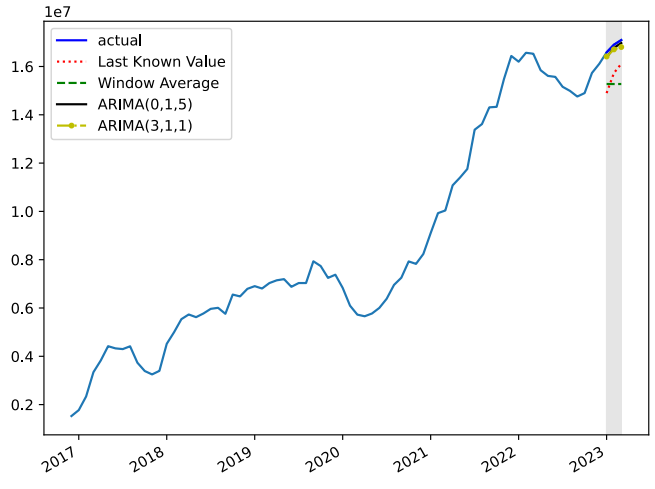

Fig. 12. chess.com traffic forecast for the first three months of 2023. Forecast window is grayed out. Whilst the lines represent the forecast. A close up of this same figure is shown in Figure 13

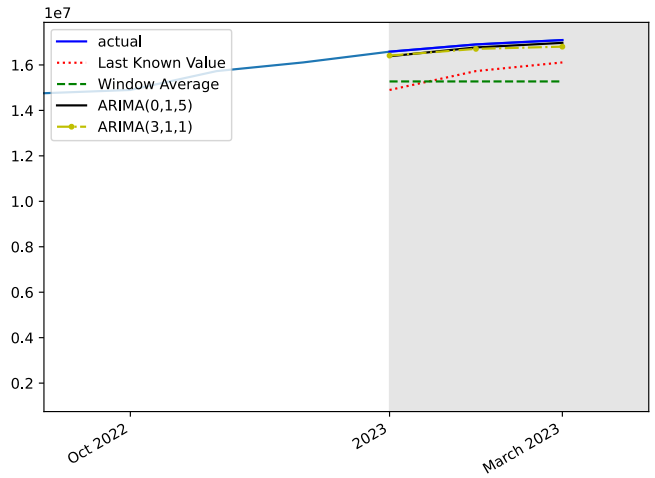

Fig. 13. chess.com traffic forecast for the first three months of 2023. Forecast window is grayed out. Whilst the lines represent the forecast.

actual values). Please refer to Table IV to see this results. It is hard to visualize which model did better by just looking at the values on the table, in the next section we will use plots, as well as the $MAPE$ to see which model performed better.

IV. ANALYSIS OF RESULTS

In this section we will compare the forecasts the different methods produced. The models we are going to compare are the following

- *Last Known Value*, this is a naive model, we simply took last three months and forecast the same values for the next three months
- *Window Average*, this is a naive model, we took the last six months, calculate the average and forecast that value for the next 3 months
- $ARIMA(3,1,1)$, this is an $ARIMA(p,d,q)$ model with order $p=3$ and $q=1$, while the difference order $d=1$
- $ARIMA(0,1,5)$, this is an $ARIMA(p,d,q)$ model with order $p=0$ and $q=5$, while the difference order $d=1$. $p=0$ means that it wont use the autoregression part of $ARIMA(p,d,q)$ instead it will use just a moving average model of order five.

Table IV shows the actual traffic value, compared with these 4 models, but to get a better grasp of how they perform we can direct ourselves to Figure 12. Figure 13 represents the same information but from a closer point of view.

It seems like the worst performing model is *Window Average*, this is not a big surprise since it is a static naive model. It will forecast the same every month and wont take into account trends. We also see that $ARIMA(3,1,1)$ and $ARIMA(0,1,5)$ seem to perform pretty similar. Nevertheless before jumping into conclusions let us calculate the $MAPE$ (mean absolute percentage error) of each model. It will basically tell us how off we were against the actual values on average. Please refer to Figure 14.

We can now get a better analysis of the results. The worst performing model is *Window Average*, followed by the *Last Known Value* which on average was off by 7.62%. This means that at any point we forecast we might be, 7.62% above or below the actual value, making this a bad model relative to

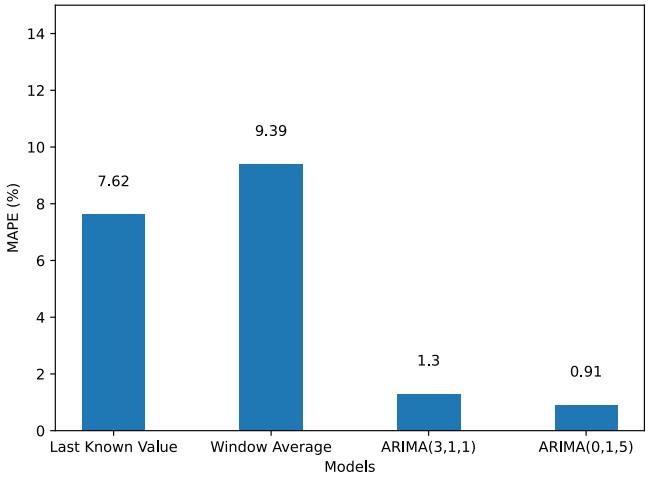

Fig. 14. Mean Average Percentage Error (MAPE) of the 4 models we used to make a forecast

the next two.

We then can take a look at $ARIMA(3,1,1)$ and $ARIMA(0,1,5)$, needless to say, these two models perform better than our baselines, and among them $ARIMA(0,1,5)$ performs better by 0.4. Since we are talking in millions here, 0.4 will have a lot of significance. Therefore the model we should opt to use is $ARIMA(0,1,5)$.

We now can start a rolling forecasting, meaning every months it passes we can run again the model with the new data point, creating more accurate results for next periods of time.

V. Conclusion

In conclusion, this study aimed to forecast web traffic using $SARIMA(p,d,q)(P,D,Q)_m$. After seeing that the time series did not had seasonality in Section III-B, we opted to use $ARIMA(p,d,q)$.

The findings indicate that the $ARIMA(0,5,1)$ model performed well in forecasting web traffic, as evidenced by the low Mean Absolute Percentage Error (MAPE) compared against the baseline models and $ARIMA(3,1,1)$ The model effectively captured the underlying patterns and trends in the web data, enabling accurate predictions.

Future research could explore using *Recurrent Neural Networks* and other deep learning models to explore more advances forecasting techniques to imporce accuracy of web traffic forecasting.

In summary, $ARIMA(0,1,5)$ demonstrated its potential as an effective tool for web traffic forecasting. Its ability to capture and predict patterns in web data makes it a valuable resource chess.com for both resource allocation and performance optimization strategies for the website. However, further research and refinement are warranted to enhance its performance and account for complex factors influencing web traffic dynamics.

Acknowledgment

I would like to express my sincere appreciation and gratitude to Marco Peixeiro for his exceptional book, "Time Series Forecasting." This book has been an invaluable resource throughout my research and study in the field of time series analysis.

Marco Peixeiro's expertise and comprehensive coverage of time series forecasting techniques have provided me with deep insights and practical knowledge. The clear explanations and numerous examples presented in the book have greatly enhanced my understanding of this complex subject.

References

[1] Chess.com. (2023, June) Chess is booming! and our servers are struggling. [Online]. Available: https://www.chess.com/blog/CHESScom/chess-is-booming-and-our-servers-are-struggling

[2] R. Casado-Vara, A. Martin del Rey, D. Pérez-Palau, L. de-la Fuente-Valentín, and J. M. Corchado, "Web traffic time series forecasting using lstm neural networks with distributed asynchronous training," *Mathematics*, vol. 9, no. 4, 2021. [Online]. Available: https://www.mdpi.com/2227-7390/9/4/421

[3] Semrush. (2023, June) Where does semrush data come from? [Online]. Available: https://www.semrush.com/kb/998-where-does-semrush-data-come-from

[4] M. Peixeiro, *Time Series Forecasting*. Shelter Island, NY 11964: Manning Publications Co., 2022.

[5] G. Box and G. Jenkins, *Time Series Analysis: Forecasting and Control*, revised edition ed. San Francisco: Holden Day, 1976.

[6] S. E. Said and D. A. Dickey, "Testing for unit roots in autoregressive-moving average models of unknown order," *Biometrika*, vol. 71, no. 3, pp. 599–607, 1984. [Online]. Available: http://www.jstor.org/stable/2336570

[7] I. T. Laboratory. (2023, June) Quantile-quantile plot. [Online]. Available: https://www.itl.nist.gov/div898/handbook/eda/section3/qqplot.htm

[8] S. Seabold and J. Perktold, "statsmodels: Econometric and statistical modeling with python," in *9th Python in Science Conference*, 2010.