Time Series for Web Traffic Forecasting

Benjamin Santana Velazquez

***Abstract*—Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum dignissim felis in nisi lobortis, nec tincidunt nulla interdum. Nulla facilisi. Proin et augue risus. Nullam non ipsum in nisi accumsan bibendum. Curabitur sed elit at est aliquet convallis. Nullam dictum ullamcorper mauris id hendrerit. Mauris a dui tincidunt, feugiat magna vel, interdum sapien.**

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

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 imporant 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 make 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 Clickstream 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 infrastrucutre of the website like server capacity, bandwith — 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." [**?**] 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 enable proactive resource allocation and ensure sufficient resources are allocated in advance? 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, $SARIMAX(p,d,q)(P,D,Q)_m$. We will find out if this model is powerful enough to create meaningful and accurate forecasts or if opting for naive methods to get better results. we should opt into the world of neural networks [1]

Unfortunately, the taffic 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 decied to go with semrush, a "all-in-one tool suite for improving online visibility and discovering marketing insights." [**?**]. 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 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. How it looks will be discussed later in the paper. What I wanted to point here is that 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 $SARIMAX(p,d,q)(P,D,Q)_m$ 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 conclution 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 $SARIMAX(p,d,q)(P,D,Q)_m$. 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 the steps we are going to follow to tackle the problem at hand. So let us start with a simple question...

What is a time series? We can define a time series as a set of points ordered in time, usually equally spaced in time. [**?**] 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 whtie noise. Any

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

model wont be able to forecast any of this part, since it is completely random

Now that we have defined these concepts of a time series, let us see how they llok in action. To demostrate how this looks in a time series we will use classical time series presented in *Box, G. E. P., Time Series Analysis, Forecasting and Control.* [**?**]. The classic Box & Jenkins airline data presents monthly totals of international airline passengers, 1949 to 1960. Table I, showcase the first 5 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.

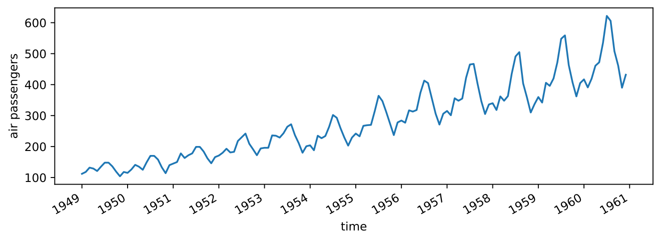

Fig. 1. The classic Box & Jenkins airline data, it presents monthly totals of international airline passengers, 1949 to 1960.

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

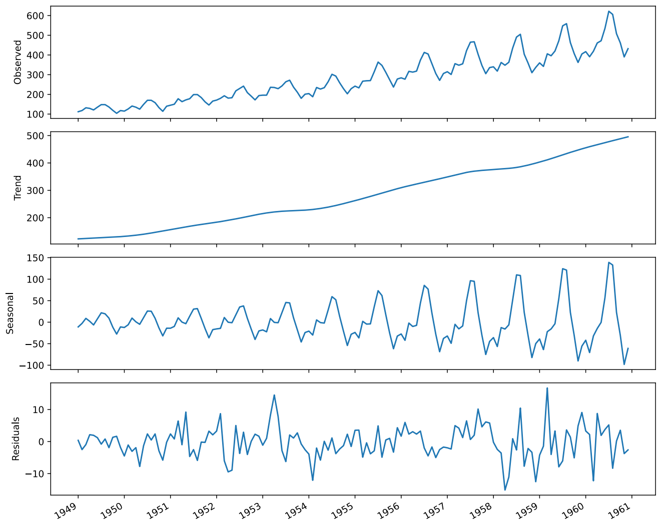

Fig. 2. Decomposition of The classic Box & Jenkins airline data, it presents monthly totals of international airline passengers, 1949 to 1960.

We can see that now we have 4 plots displaying. 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 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 has an order*, — a time series is indexed by time, order must be kept, in regression tasks when you want to predict revenue based on ad spend, it does not matter when certain amount was spent on ads. — and *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.

Before jumping into $MA(q)$ and $AR(p)$ I want to explain some concepts, the first one is *baseline models*. A baseline model is a trivial solution to our problem, it uses heuristics more than deep statsticts knowledge or anything else. 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 "I forecast next month our sales will be the mean of all the time series". We got this forecast using a simple statistics concept like it is the mean. It is likely that we can get a better forecast than this using more complex stastics. 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 value known. 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 maybe performs worse than this naive methods, then we would be wasting resources every forecast running our complex statistical models. Because we could just copy and past the last values, or use the mean, or some other baseline model. In Section IV we will compare our $SARIMAX(p,d,q)(P,D,Q)_m$ model with a baseline model and conclude if it is worth it to use it against some simpler methods.

One last thing on the baseline methods. How do 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 [**?**]. 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 calcluate 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 transfrom of the data after finishing the model.

We can test for stationarity with the *Augmented Dickey-Fuller* (ADF) test [**?**], 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 to go 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 depens linearly on the mean of the series, the current error term, and past error terms. [**?**]. 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.

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

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 large 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.

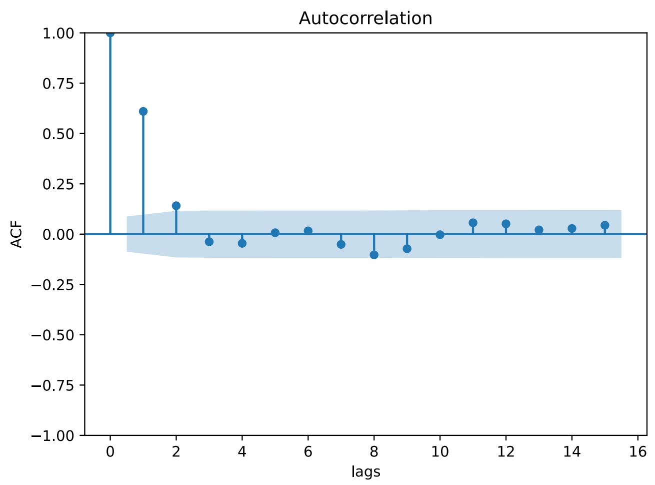

Fig. 3. Example ACF plot, where we see a relevant correlation up to lag 2

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

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. [**?**]

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

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

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 [**?**] like see in Figure 4.

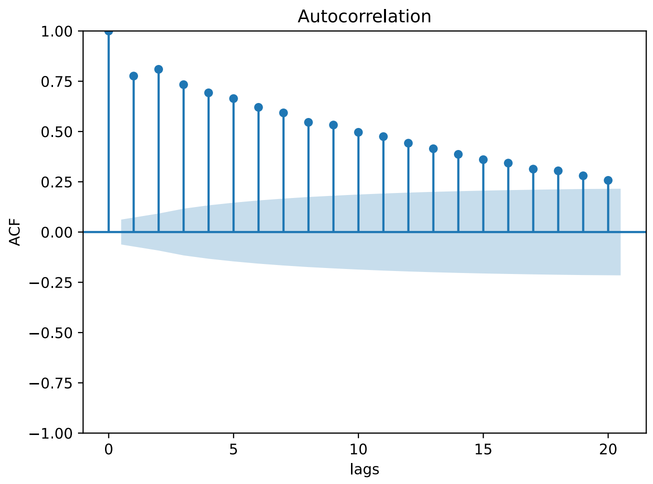

Fig. 4. Example ACF plot on an autoregressive process

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.

We can see in this example plot that we have no significant coeficients 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

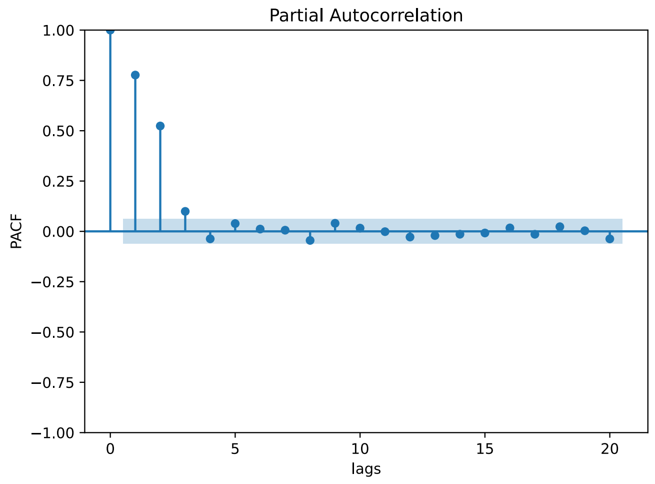

Fig. 5. Example PACF plot, We can see that we have no significant coeficients after lag 3, meaning $AR(p)$ where $p = 3$, we have an autoregressive process of order 3

that something is missing. What happens when the ACF plot shows a slowly decaying pattern 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 precnese 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 combien 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, $\hat{L}$ maximum value of the likehood function model.

Using AIC to select our model, allows us to keep a balance between the complexity of the model and its goodness of fitting the data. This is becase 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 substracting the likelihood function model to times two this number, the higher the order gets, the AIC increases, penalizng more complex models.

$\hat{L}$ (the likelihood function) measures the goodness of fit a model has. Comparing it with the distribution function will make it easier to understand. It can be seen as the opposite of the distribution function. In the distribution function, given a model with fixed parameters, it measuers the probability of obvserving a data point. In the likelihood function, given a set of data, it tells us how likely it is that different model parametrs generate the data.

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

- 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* $ARMA(1, 1)$ If a model fits the data really well, the maximum value of likelihood will be high [**?**]. Since AIC function subtracts the $ln(\hat{L})$ from $2k$, it balance between underfitting and overfitting.

Taking into account that AIC quantifies the quality of a model in relation to other models only. so it is a relative measure of quality. What we end upj doing in pracitce is iterating through some combinations of p and q, to calculate the AIC values and selecting the one with the lowest score. Findingin 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 (except from d and D, but more on that later).

Coming back to our inital problem (finding the order of $ARMA(p, q)$), 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 [**?**]. 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 preddict 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. [**?**]

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 residuls 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* [**?**], allows us to do diagnostic plots for our residuals in a really easy way.

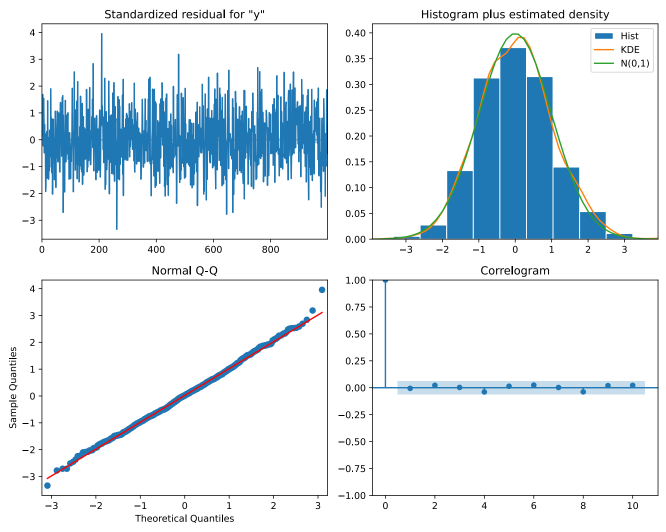

Fig. 6. Example Diagnositc Plots for the residuals of a $ARMA(1,1)$ model

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 behaviout 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 distribtution. 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.

III. Eperimental framework

IV. Analysis of Results

V. Conclusion

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.