**Time Series Forecasting vs. Regression**

[[Danayt Aman](https://medium.com/@danaytaman?source=post_page-----cf89d0d0f3bd--------------------------------)](https://medium.com/@danaytaman?source=post_page-----cf89d0d0f3bd--------------------------------)

[Danayt Aman](https://medium.com/@danaytaman?source=post_page-----cf89d0d0f3bd--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F63198ad43c62&operation=register&redirect=https%3A%2F%2Fmedium.com%2F%40danaytaman%2Ftime-series-forecasting-vs-regression-cf89d0d0f3bd&user=Danayt+Aman&userId=63198ad43c62&source=post_page-63198ad43c62----cf89d0d0f3bd---------------------post_header-----------)

6 min read

·

Nov 25, 2023

54

During a conversation with

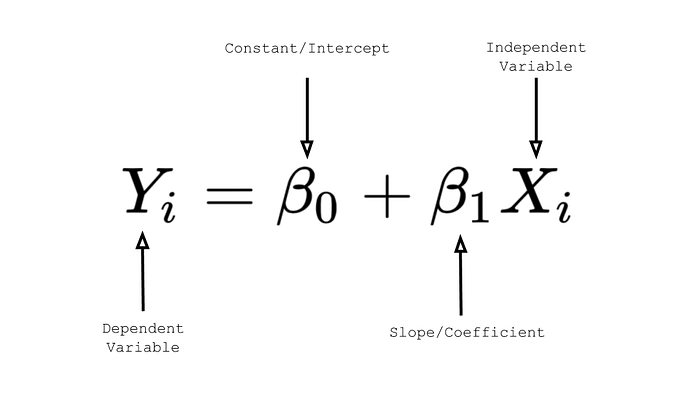
[Abel Mehari](https://medium.com/u/a71cc91fe7e1?source=post_page-----cf89d0d0f3bd--------------------------------)

, I discovered that the energy market operates similarly to a stock exchange market. This was fascinating and sparked my interest in time series analysis, which I thought would be a great topic for my capstone project. Unfortunately, the bootcamp from Flatiron school I’m currently enrolled in doesn’t offer a time series course. I took it upon myself to conduct some outside research and decided to share what I’ve learned about time series predictive modeling.

**Regression Models for Time Series Forecasting:**

Time series forecasting is an important tool used in various industries to predict future values based on historical data. Accurate forecasting is essential for businesses to make informed decisions, optimize resources, and plan for the future. In this case we’ll be forecasting the future consumption of energy for a county in the US. There’s time series models such as LSTM, ARIMA, SARIMAX, and Prophet to mention some that are used for time series but another popular approach to time series forecasting is the use of linear regression models. Not to mention that I found linear regression models to take much less computational time compared to the time series models mentioned above. In this blog, we will explore the similarities and differences between time series forecasting and linear regression models, as well as key considerations when using linear regression for time series forecasting.

Linear regression is a statistical technique used to model the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the variables and aims to find the best-fit line that minimizes the sum of squared errors (residuals). Here’s an equation of a simple linear model with one independent variable. The model explains how a change in X will affect Y, in a more direct interpretation, the coefficient indicates the change in the dependent variable for a one-unit change in the independent variable.



**Creating lag Variables:**

In time series forecasting, linear regression can be applied by treating time as an independent variable and using historical data to predict future values. Both time series forecasting and linear regression models involve predicting future values based on historical data. They both utilize mathematical relationships to establish patterns and trends in the data. However, there are some key differences between the two approaches. One major difference is the time dependency in time series forecasting. Time series data is inherently sequential, with each observation depending on the previous ones. Linear regression models, on the other hand, do not consider the time dependency and assume that the observations are independent.

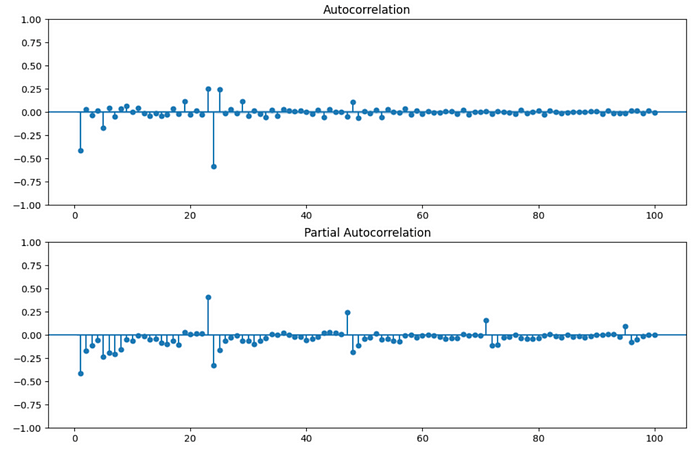
So how can we help our regression models understand time dependencies to analyze each observation based on the previous one?There’s several ways to do this but we’ll explore creating lagged variables in this blog. Lagged variables or lagged values, are past values of a variable that are included as predictors in regression models, they represent the values of the target variable at previous time points. Lag variables are created by shifting the values of the variable by a certain number of time periods. By including lagged variables, the regression model can capture patterns and dependencies of our data. For example in my energy consumption modeling process I shifted the values by 24 hours. This means that for each hour, I’m including the energy consumption value from the previous day at the same hour, helping the model pick on daily trends. Since there’s no previous values for the first 24 hours our shifted data would have null values for the first 24 entries and we can drop those for modeling purposes.

# Creating the lag variables  
for i in range(24):  
df\_lag['lag'+str(i+1)] = df\_lag['SDGE'].shift(i+1)  
# Since the first 24 values won't have any 24th lag, they will be null values, therefore dropping them  
df\_lag = df\_lag.dropna()

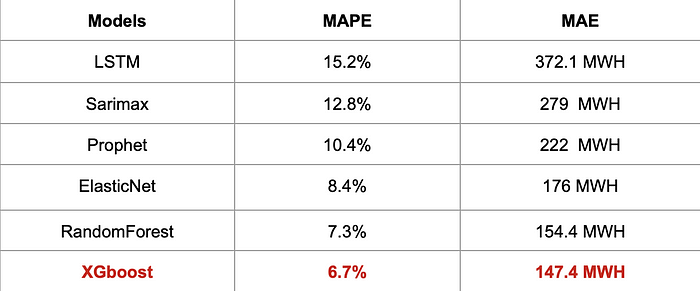
**Autocorrelation and Partial autocorrelation:**

Picking the right lagged variable is a challenge but we can use Auto and partial autocorrelation plots to capture the correlation of the current observation and its lagged value. and ofcourse domain knowledge is key here. Autocorrelation refers to the correlation between a variable and its lagged values. Time series forecasting models need to account for autocorrelation to capture the persistence of past values in predicting future values. Linear regression models as mentioned earlier, assume that the observations are independent and do not consider autocorrelation unless explicitly included as lag variables. We scan for significant spikes as they indicate potential lagged variables to include in our regression model.

# Differencing the data this time to remove the trend and seasonality  
dfacf = []  
dfacf = df\_hr['SDGE']  
dfacf = dfacf.diff().dropna() #Previous hour diff  
dfacf = dfacf.diff(24).dropna() # Daily diff our value 24h  
dfacf = dfacf.diff(24\*365).dropna()#Yearly diff  
lags = 100  
acf\_pacf\_plots(dfacf, lags = lags, figsize = (12,8))



We can in our data have high seasonality even after differencing we’re seeing some seasonality but the significant peak and drop in our graphs around 24 lags are one way to try determining the lagged variable to include. We’ll go ahead and see how our regression models perform using this lagged variable compared to some time series models. For the purpose of this blog we’ll focus on Mean absolute error tells us how far off, on average, our predictions are from the actual values, and mean Absolute Percent error which is the average percentage difference of the actual and the predicted to evaluate model performance.



**Fourier Transformation for TS Models:**

All time series models used fourier transformation to pick on seasonality as we saw in the auto and partial correlation. Our data has high seasonality and we need a signal to help our time series models pick on those daily, weekly and yearly fluctuations. For the regression models we used a 24 hours shifted lagged variable with the first 24 hours dropped using the code provided previously. The fourier transformation used for the time series models mentioned above, LSTM, Prophet and Samimax is provided below:

def add\_fourier\_terms(df, year\_k, week\_k, day\_k):  
"""df: dataframe to add the fourier terms to  
year\_k: the number of Fourier terms the year period should have.  
Thus the model will be fit on 2\*year\_k terms (1 term for sine and 1 for cosine)  
week\_k: same as year\_k but for weekly periods  
day\_k:same as year\_k but for daily periods """  
for k in range(1, year\_k+1): # year has a period of 365.25 including the leap year  
df['year\_sin'+str(k)] = np.sin(2 \*k\* np.pi \* df.index.dayofyear/365.25)  
df['year\_cos'+str(k)] = np.cos(2 \*k\* np.pi \* df.index.dayofyear/365.25)  
for k in range(1, week\_k+1): # week has a period of 7  
df['week\_sin'+str(k)] = np.sin(2 \*k\* np.pi \* df.index.dayofweek/7)  
df['week\_cos'+str(k)] = np.cos(2 \*k\* np.pi \* df.index.dayofweek/7)  
for k in range(1, day\_k+1): # day has period of 24  
df['hour\_sin'+str(k)] = np.sin(2 \*k\* np.pi \* df.index.hour/24)  
df['hour\_cos'+str(k)] = np.cos(2 \*k\* np.pi \* df.index.hour/24)  
add\_fourier\_terms(df, year\_k= 5, week\_k=5, day\_k=5)

**Considerations:**

When using linear regression models for time series forecasting, there are several key considerations to keep in mind. First, the stationarity of the time series data needs to be assessed. Stationarity refers to the stability of the statistical properties of the data over time. If the data is non-stationary, transformations or differencing may be necessary to make it stationary before using regression models.

Time series forecasting and linear regression models share similarities in predicting future values based on historical data. However, they differ in terms of time dependency, handling of autocorrelation, incorporation of seasonality and trend components, and treatment of exogenous variables. When using linear regression models for time series forecasting, it is important to consider the stationarity of the data, select appropriate lag variables, handle outliers and missing values, and evaluate model performance using appropriate metrics. Understanding the unique aspects of time series data and utilizing appropriate modeling techniques are important for accurate forecasting and informed decision-making processes in various industries.

Regression might not always work for time series and it takes trial and tests to find the right algorithm as well as the right lag variable, but hopefully the methods discussed above come handy to improve models predictive capability.