JACOBS UNIVERSITY

Application of Machine Learning Methods from Time series Forecasting to Predict Load and Price for the France Energy Market

by

Didier Iyamuremye

Advanced Project II in Data Engineering

Prof. Stefan Kettemann
Advanced Project Supervisor

Date of Submission: February 7, 2020

—Jacobs University Bremen —

Abstract

The aim of this project is to apply Machine Learning algorithms and experiment with a real world dataset where data is being used to make predictions of the future price and load by the help of past recent observations,I have used some forecasting methods like Vector Auto Regression Model and Auto Regression Integrated Moving Average Model to predict the future price and load with the goal of maintaining the price stability.

Contents

1 Introduction

In this project, I have used the same open source like for the previous advanced project 1 titled as ”France Enery Market”, the platform designed to provide data to use for analysis, I have managed to download all the specific data for France energy market from the year 2016 to 2018 and then selected only 5 columns from the entire data in the wind and solar energy area which are: The actual total load, day ahead price, day-ahead total load forecast, Generation-solar day ahead and Generation-wind onshore day ahead to use according to the main purpose of the project of applying Machine Learning models to the data in order to do forecasting of the future load and the price.[1]

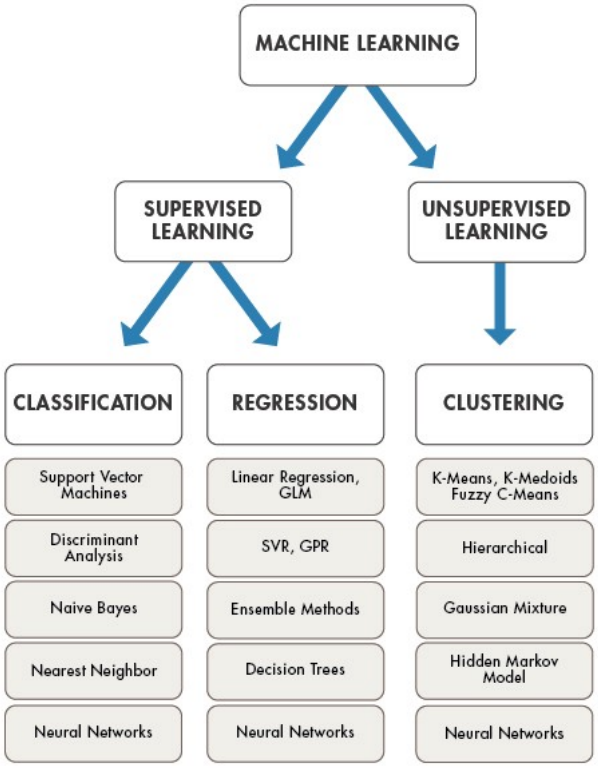


Figure 1: Machine Learning Algorithm

[2]
Machine Learning defined as a type of data analysis that uses algorithms that learn from data. And also taken as a study of algorithms and statistical models that computer systems use to perform a specific task without using any explicit instructions, but relying on patterns and interested in inference or prediction instead. Machine Learning is a subset and a blanch of artificial intelligence based on the idea that systems can learn from data, identify patterns and also make decisions with less human intervention and is typically categorized as supervised or unsupervised learning,for supervised learning technique you have to train the machine using data which is well labeled where the output data patterns are known to the system,for the supervised learning algorithms we have different models like linear regression,k nearest neighbour,Random Forest etc but in contrast unsupervised learning technique is another machine learning technique where in this case you do not need to supervise the model it helps us to find all kind of unknown pat-

terns in data, it uses different methods as well to do clustering like Convolutional neural network,auto vector regression,k-means clustering algorithms,is known to be a complex method compared to supervised learning.[2]

2 Data Description

During the data description stage,I have managed to get the dataset with the total size of 26308 where the data attributes are generally the solar generation forecast,actual total load,solar generation actual,day ahead price,load day ahead,wind onshore generation forecast and wind onshore generation actual,actual total load before doing selection of the ones to use for models, Then I have only selected 5 columns which are :The actual total load,day ahead price,day-ahead total load forecast,Generation-solar day ahead and Generation-wind onshore day ahead.Though there was different options to download data using any data formats, I have chosen csv format for all datasets.And I have only taken the time range of 2016 to 2018. Table 1 explains how the data were organized.

Table 1: Data Description table.

Data Attributes	Data Description	Data types	Null values
Day-ahead Total Load Forecast	total load forecast in MH/h	float64	20
Actual Total Load	actual total load in MW/h	float64	4
cet-timestamp	time format :DD.MM.YY HH:MM	object	0
Day-ahead-Price	day ahead price in EUR	float64	4
Generation-wind onshore Day ahead	wind onshore day ahead in MW/h	float64	4
Generation-solar Day ahead	solar day ahead in MW/h	float64	20

3 Preprocessing Data

Data processing is a data mining technique that involves transforming raw data into an understandable format. By the fact that real-world data is often incomplete and inconsistent or lacking in certain trends or behaviors and is likely to contain many errors. so from this stage of preprocessing the data,after downloading the data sets via an open source platform[3],generally the data was not consistent either by its time series formats or even missing values,smooth noisy data it was required to remove some outliers and then resolve inconsistencies before starting processing data in order to try machine learning methods. I did data cleaning by filling the missing values,identity and remove the outliers,even because the format of Time series was string it was not easy to manage them,in addition Machine Learning has steps for Data Proprocessing.
Step 1: Import Libraries
Step 2: Importing the Dataset
Step 3: Taking care of the missing data in dataset
Step 4: Encording categorial data
Step 5: Splitting the dataset into training and test set
Step 6: Feature extraction/scaling

Then from there apply the Machine learning models to the data. Additionally, I have first changed its format to timestamp as shown in the table below.

Table 2: Original Data Format table.

MTU(CET)	Generation Solar[MW]	Generation Wind onshore[MW]	Data types
01.01.2016 00:00-01.01.2016 01:00	132	937	csv
01.01.2016 01:00-01.01.2016 02:00	809	1045	csv
01.01.2016 02:00-01.01.2016 03:00	1622	1055	csv
01.01.2016 03:00-01.01.2016 04:00	2154	1065	csv
01.01.2016 04:00-01.01.2016 05:00	2318	1076	csv

I have shown only 2 columns among all columns I have in the dataset ,the time shown in table 2 has the string format, and there was much time redundancy or duplicates which was not necessary which also caused some difficulties on the stage of doing plots for predictions,as a typical example here, I have shown only 2 different categories of renewable energies among the one I have downloaded,5 first range time with corresponding amounts of solar power generated in megawatts,generated wind onshore with also 5 amounts of data in megawatts during the year 2016.

Table 3: New Data Format Table.

MTU(CET)	Generation Solar[MW]	Generation Wind onshore[MW]	Data types
01.01.2016 01:00	132	937	csv
01.01.2016 02:00	809	1045	csv
01.01.2016 03:00	1622	1055	csv
01.01.2016 04:00	2154	1065	csv
01.01.2016 05:00	2318	1076	csv

This table 3 described the new data I have used after changing the time format from string to timestamp, I have shown only two different amounts which are generation solar in mega watts and generation wind onshore in mega watts with the csv data formats during the year of 2016.

4 Data Acquisition

The Data collection used to analyse my case study of applying Machine Learning methods of France energy was found from entsoe Transparency platform [3]. In brief entsoe Transparency platform is an open source platform defined as a central collection and publication of electricity generation,consumption and transportation data specifically only limited for the pan-European countries. However it requires to create an account in order to be able to export the data you want within different formats like csv,xlsx,xml,such data are for day ahead,total load,price for wind and solar generated within the years from 2016 to 2018 I was able to export in different ways depend on which data you want by Load side total load,total load forecast-week ahead,month ahead or year ahead,Generation i got Generation Forecasts for wind and solar Transmission data view I used Day-ahead prices.

5 Data Exploration

Briefly, data exploration is an approach that is similar to initial data analysis,whereby the main idea is to understand what is the dataset and the characteristics of the data.Here I have tried different machine learning models to check how it performs on the dataset.First of all before applying a model you have to check if the data is stationary,time series is said to be stationary when the mean,variance and covariance remains constant over time, if time series of the data is non-stationary, then you make it stationary by doing differencing and applying log transformations.

5.1 Forecasting with ARIMA Model

ARIMA is one of the machine learning model which is abbreviated as 'Auto Regressive Integrated Moving Average' which is actually a class of models that explains a given time series based on its own past values,roughly meant that model uses lags and the lagged forecast errors so that the equation can be used to forecast future values.
Mathematically,an ARIMA model can be created using the statistical model library as follows:
1.Define the model by calling ARIMA() function and then passing in the p, d, and q parameters.
2.The ARIMA model is prepared on the training data by calling the fit() function.
3.Predictions can be made by calling the predict() function and specifying the time to be predicted or the index of time.
An ARIMA model is characterized by 3 terms: p, d, q where:
p is the order of the AutoRegressive term
q is the order of the Moving Average term
d is the number of differencing required to make the time series stationary
Then,If a time series, has seasonal patterns, then you need to add seasonal terms and it becomes SARIMA, short for 'Seasonal ARIMA'. More on that once we finish ARIMA. The mathematical formula for the AutoRegressive and Moving Average models,first of all a pure Auto Regressive model is one where Yt depends only on its own lags,means that Yt is a function of the lags of Yt.
[4]

Y(t)=$\alpha + \beta 1Y(t-1) + \beta 2Y(t-2) + .. + \beta pY(t-p) + \theta 1\epsilon(t-1) + \theta 2\epsilon(t-2) + .. + \theta q\epsilon(t-q)$ (1)

where, $Y(t-1)$ is the lag1 of the series, $\beta 1$ is the coefficient of lag1 that the model estimates and α is the intercept term, also estimated by the model.

Likewise,again for a pure Moving Average (MA only) model is one where Yt depends only on the lagged forecast errors. The formula above is the final equation of ARIMA model by combining AR and MA models. ARIMA model in full detail,Predicted Y(t) = Constant + Linear combination Lags of Y (up to p lags) + Linear Combination of Lagged forecast errors (up to q lags).The objective, therefore, is to identify the values of p, d and q.As I said before d stands for differenciation, so if the time series is not stationary you have to apply differencing in order to make it stationary, my actual entire data was stationary(DF test results find it on page 12) but because ARIMA model need to split the data into small portion of

recent observations and after doing that, the new observation data becomes non-stationary,I have done the stationarity process to make it stationary otherwise it would have affected the model parameters. From the time series where is stationary, the value of d=0, but once you split the data into small portion of recent observations and data becomes non-stationary,if you make the time series stationary again,you get the value of p,d,q and the value of d will also change,from there you get everything needed to find the ARIMA model where you only have to pass the ARIMA() function the order which has the values of p,d,q and gives you the observations of the Akaike Information Critera (AIC) which is a widely used as a measure of a statistical model. Also it basically quantifies the goodness of fit and the simplicity of the model into a single statistic,Bayesian Information Criteria(BIC).[5]
Steps for the methods used to check if the time series of the data is stationary:
By checking the series if is stationary we have to make sure that variance,mean and covariance are constant with time if they are changing with time that means that series is not stationary. Then you might follow these steps to make the series stationary. Mathematically, we can use statistical tests properties to check of a given series are not constant with time, which is the condition for stationary time series. Suppose we have a time series:

$$\textbf{Y(t)} = a * Y(t-1) + \epsilon(t)$$

(2)

where $Y(t)$ is the value at the time instant t and $\epsilon(t)$ is the error term. In order to calculate $Y(t)$ we need the value of $Y(t-1)$ which is :

$$\textbf{Y(t-1)} = a * Y(t-2) + \epsilon(t-1)$$

(3)
Then, if we do that for all observations, the value of $Y(t)$ will come out to be :

$$\textbf{Y(t)} = a^n * Y(t-n) + \sum_{t=1}^{n} (\epsilon(t-i) * a^i)$$

(4)

If the value of a is 1 which is a unit in the above equation, then the predictions will be equal to the $Y(t-n)$ and sum of all erros from t-n to t, which means that the variance will increase with time. This is known as unit root in a time series. We know that for a stationary time series, the variance must not be a function of time. The unit root tests check the presence of unit root in the series by just checking if value of a=1. Below are the two of the most commonly used unit root stationary tests:[6] **ADF(Augmented Dickey Fuller)Test** The Augmented Dickey Fuller test is one of the most popular statistical tests. Which can be used to determine the presence of unit root in the series, and hence help us undestand if the series is stationary or not. The null and alternate hypothesis of this test are follows: Null Hypothesis: The series has a unit root with the value of a=1

Alternative Hypothesis: The series has no unit root
Addition to that, if we fail to reject the null hypothesis, we can immediately say that the series is non-stationary, this tells us that the series can be linear or difference stationary.
Test for stationary: If the test statistic is less than the critical value, we can reject the null hypothesis means that the series is stationary. When the test statistic is greater than the critical value, we fail to reject the null hypthoses which means that the series is not stationary.
Generally, we have different types of stationarity,which are follows:
1.Strict Stationary:this strict stationary series satisfies the mathematical definition of a stationary process. Then for a strict stationary series the mean,covariance and variance are not the function of time. The main aim is to convert a non-stationary series into a strict stationary series for making predictions.
2.Trend Stationary:It's a series that has no unit root but displays a trend is referred to as a trend stationary series. Once the trend is removed, the resulting series will be strict stationary.
3.Difference Stationary: This type of stationary it's a time series that can be made strict stationary by differencing falls under difference stationary. ADF test is also known as a difference stationarity test according to what i have explained early.
Differencing as one way used to make series stationary, mathematically it can be written as:

$$\textbf{Y(t)} = Y(t) - Y(t-1)$$

(5)
Where $Y(t)$ is the value at a time t

But for seasonal Differencing, instead of calculating the difference between consecutive values, we calculate the difference between an observation and a previous observation from the same season, mathematically it can be written as:

$$\textbf{Y(t)} = Y(t) - Y(t-n)$$

(6)

where $Y(t)$ is the value at a time t for previous observation, and $Y(t-n)$ is the value at a time t and n(number of observations) for an observation.

For figure[3], I have used the C[-1,1] as the correlation function with the intervals between -1 and 1 on the y axis and the day ahead price[EUR/Mwh] data but here shows the size of the laf between the elements of the time series on the x axis,AutoCorrelation Function(ACF) is the residual used to determine ARIMA model selection.

In order to select the ARIMA Model order,to get the measure of Accuracy I have used the Mean Squared error(MSE),Mean Absolute Error(MAE),Mean Percentage Error(MAPE) and the Root Mean Squared Error(RMSE), these are their mathematical formula:

Mean squared error MSE = $\frac{1}{n}\sum_{t=1}^{n} e_t^2$

Figure 2: Differencing Recent Observation for Day-ahead Price [EUR/MWh].

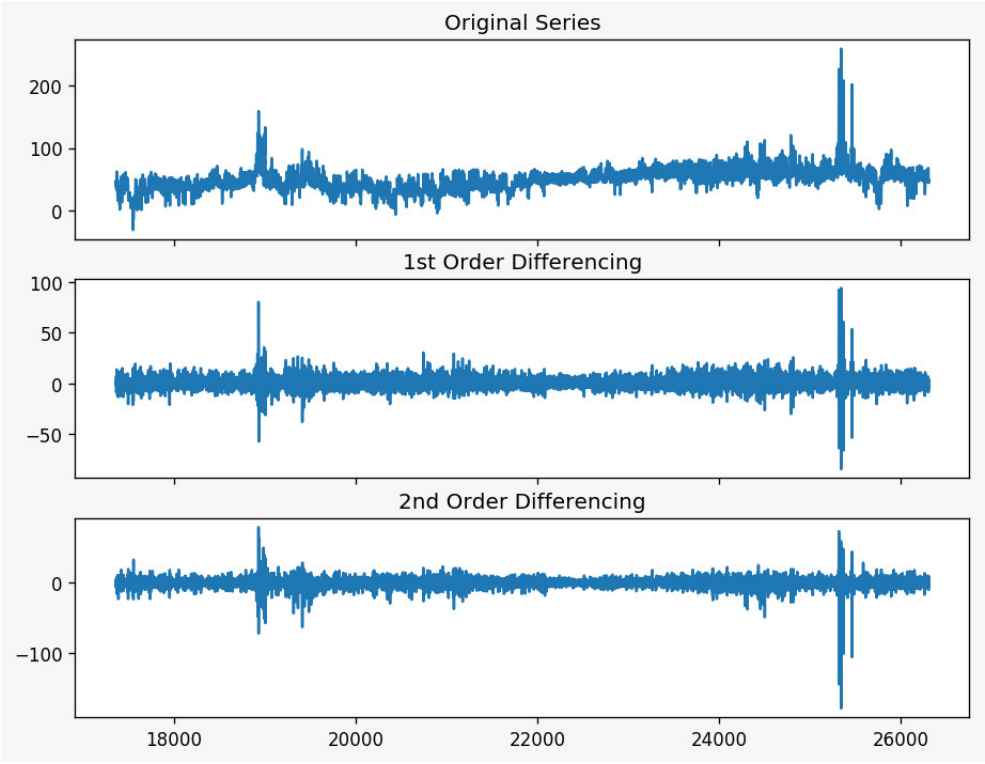
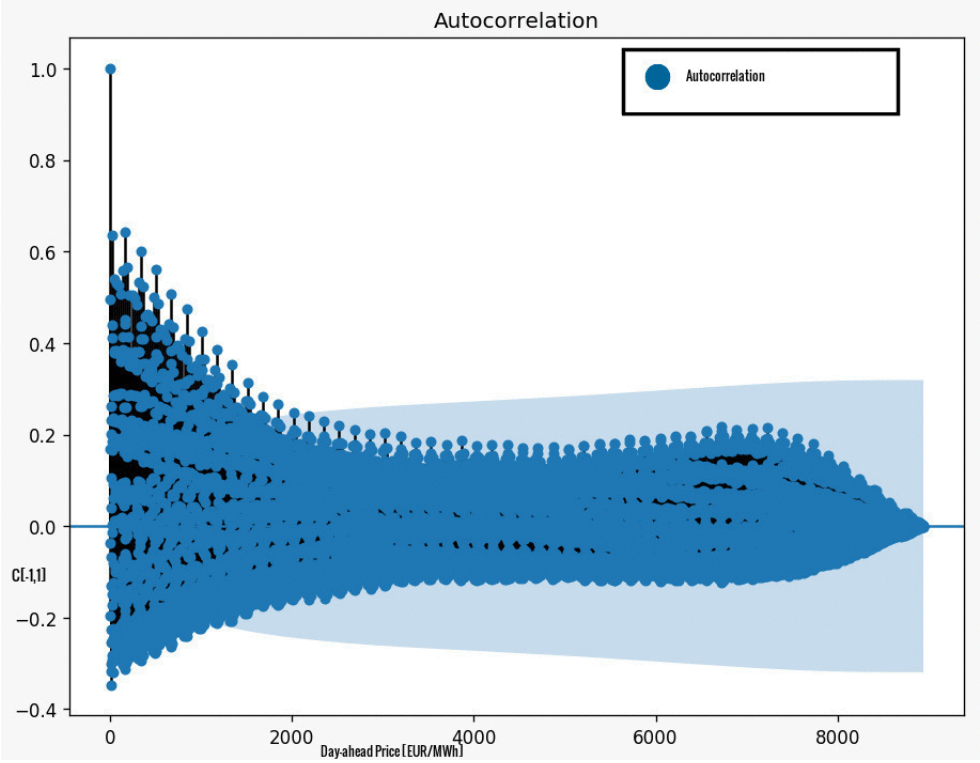


Figure 3: Autocorrelation for Day-ahead Price [EUR/MWh].



Root mean squared error RMSE = $\sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$

Mean absolute error MAE = $\frac{1}{n}\sum_{t=1}^{n} |e_t|$

Mean absolute percentage error MAPE = $\frac{100\%}{n}\sum_{t=1}^{n} \left|\frac{e_t}{y_t}\right|$

Where MSE=Mean Squared Error,n is the number of samples, t is the time,e is the estimator MAPE=Mean Absolute Percentage Error is a measure of prediction accuracy of a forecasting method in statistics like in trend estimation and also used as a loss function for regression problems in machine learning. MAE=Mean Absolute Error, refers to the results of measuring the difference between two continuous variables but in machine learning is a model evaluation metric often used with regression models. RMSE= Root Mean Squared Error is just a squared root of the Mean squared error. Due to the fact that ARIMA model does not performing well on the large amount of data sets, after dividing the time series data into small portion, I have done Cross Validation(CV)in order to test the performance of ARIMA model after using only few selected recent observations. Cross validation uses this mathematical formula:

$$\textbf{CV} = 1/T\sum_{t=1}^{n}[et/(1-ht)]^2$$

(7)
Where et is the residual obtained from fitting the model to all T observations. Meanwhile when computing the cross validation statistic, it is not needed to actually fit T separate models.[7]

5.2 Forecast Price and Load using VAR Model

5.2.1 VAR Model Overview

Vector Autoregression(VAR) is one of the machine learning methods used in a multivariate forecasting algorithm that can take place when two or more time series influence each other. It is called Autoregressive because each variable here I mean time series is modeled as a function of the past values, that is the predictors are nothing but the lags(time delayed value) of the series. In addition to that, VAR model is bi-directional not uni-directional as I said from above the reason is that the variables influence each other.
VAR model formula, each variable is modeled as a linear combination of past values of itself and the past values of other variables in the system. Since you have multiple time series that influence each other, it is modeled as a system of equations with one equation per variable(Time series).
For example, if you have 4 time series that influence each other, we will also have a system of 5 equations. Generally, the equation is framed like this:
Let's suppose that you have two variables(Time series) Y1 and Y2, from there you need to forecast the values of these variables at time(t).

To calculate Y1(t), Vector Autoregression model will use the past values of both Y1 as well as Y2. In the same manner, to computer Y2(t), the past values of both Y1 and Y2 be used.
The system of equations for a VAR(1) model with two time series(variables Y1 and Y2) is as follows:

$$\textbf{Y(1,t)} = \alpha 1 + \beta 11, 1Y(1, t-1) + \beta 12, 1Y(2, t-1) + \epsilon(1, t)$$

(8)

$$\textbf{Y(2,t)} = \alpha 2 + \beta 21, 1Y(1, t-1) + \beta 22, 1Y(2, t-1) + \epsilon(2, t)$$

(9) Where, $Y(1, t-1)$ and $Y(2, t-1)$ are the first lag of time series Y1 and Y2 respectively. The above equation is referred to as a VAR(1) model,because each equation is of order 1, and it contains up to one lag of each of the predictors(Y1 and Y2). So what I can add to that is that, since the Y terms in the equations are interrelated, the Y's are considered as endogenous or produced variables, rather than as exogenous predictors. Likewise, the second order VAR(2) model for two variables would include up to 2 lags for each variable (Y1 and Y2).

$$\textbf{Y(1,t)} = \alpha 1 + \beta 11, 1Y(1, t-1) + \beta 12, 1Y2, (t-1) + \beta 11, 2Y(2, t-2) + \beta 12, 2Y(2, t-2) + \epsilon(1, t)$$

(10)

$$\textbf{Y(2,t)} = \alpha 2 + \beta 21, 1Y(1, t-1) + \beta 22, 1Y(2, t-1) + \beta 21, 2Y1(1, t-2) + \beta 12, 2Y(2, t-2) + \epsilon(2, t)$$

(11)
Now, you can imagine what a second order VAR(2) model with 3 variables(Y1,Y2,Y3) would look like with 3 different equations and so on till n order VAR(n) model.[8]

5.2.2 VAR Model Algorithm

Formally,the procedure to build a VAR model involves the following steps:

1.Analyze the time series characteristics.
2.Test for causation amongst the time series.
3.Test for stationarity.
The observations in a stationary time series are not dependent on time,so time series are stationary if they do not have trend or seasonal effects. You might check if the data time series is stationary by reviewing the summary statistics. Where Summary statistics calculated on the time series are consistent over time, like the mean or the variance of the observations. Mean that when a time series is stationary, it can be easier to model. Statistical modeling methods assume or require the time series to be stationary to be effective. then if not stationary apply the step 4.
4.Transform the series to make it stationary, if the data is statistically not stationary.
5.Find optimal order (p). generally,keeping variables that are not statistically significant can really reduce the model's precision, so it's standard practice to sue

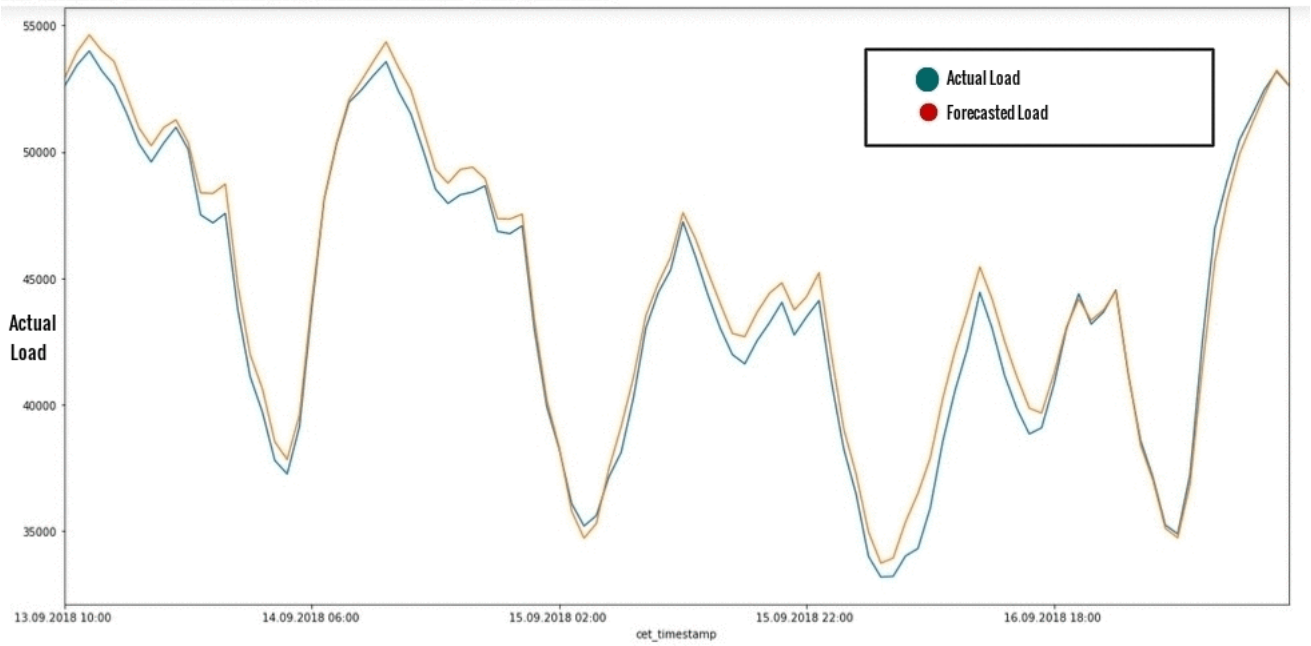
the coefficient p-values to decide whether to include variables in the final model.
6.Prepare training and test datasets.
7.Train the model.
8.Roll back the transformations by doing differenciantions,if the time series is not stationary, and we do that to get the real forecast.
9.Evaluate the model using test set
10.Finally,Forecast to future

5.3 Results for ARIMA and VAR Models

5.3.1 VAR Model Performance Results

In VAR model,each variable is a linear function of past lags of itself and past lags of the other variables,In order to forecast the total load, I have split the training data into 90% with 23676 of data and 10% of testing data with 2631 data,chosen lag order(17) and the number of observations(100), to check how model fits I have selected 300 observations. While I tried to evaluate the quality of forecast using MSE,MAPE and RMSE and the best result where MSE=41.9300, MAPE=35.049 and RMSE=6.4360 which gives the better overview of how the model forecast good for future total load. Then forecasts the future daily load with recent 100

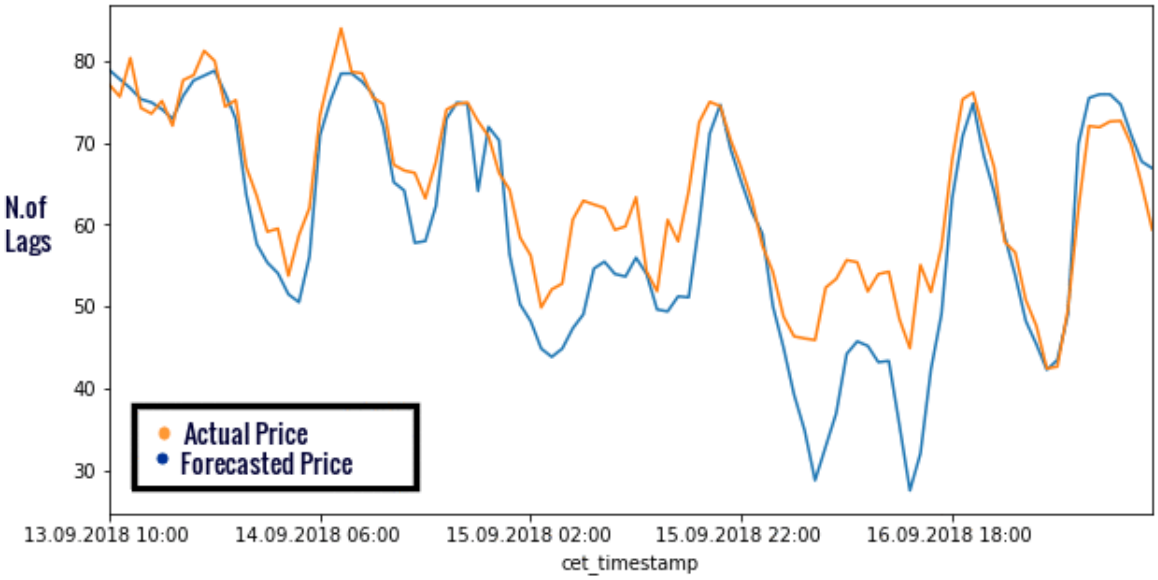
Figure 4: VAR Model-Forecasted load.



observations(nobs=100) while the model fitted at 600 data, VAR model does better for correlation between the actual load and the forecasted load according to the plot below, where it has a better correlation during the Autumn season(september) time series,from the plot x axis has the actual load data and y axis has the daily 3 hours ahead time series of the month of september in the year 2018. From the experience VAR Model does better for forecasting the load than predicting the price according to the result I have gotten,in addition to that though generally they say that the larger the number of the data sets, the higher the better the model performed, but in my scenario, once I take some few recent

observations like typical example of 100 observations, VAR model perform better in forecasting the load rather than taking the entire data sets.

Figure 5: VAR Model-Forecasted Price.



For forecasting price using VAR Model, for the first trial I have used same past recent observations of 90% for training with 23676 data and 10% for testing with 2631 data but does not perform better like for forecasting the total load, then I used cross validation to assess the predictive performance of the model and to be able to judge how it performs outside the sample to a new data set and the motivation to use cross validation techniques was that when we fit a mdoel, we are fitting it to a training dataset. In order to forecast 3 hours ahead for the energy price I have trained the data with size 8769 with lag order(26) for the first time and trained again the data with size 17538 with same lag order of 26 and the evaluation gave me this performance for MSE,MAPE,RMSE and the last line in the table is for average loss errors.

Table 4: VAR Evaluation Performance results.

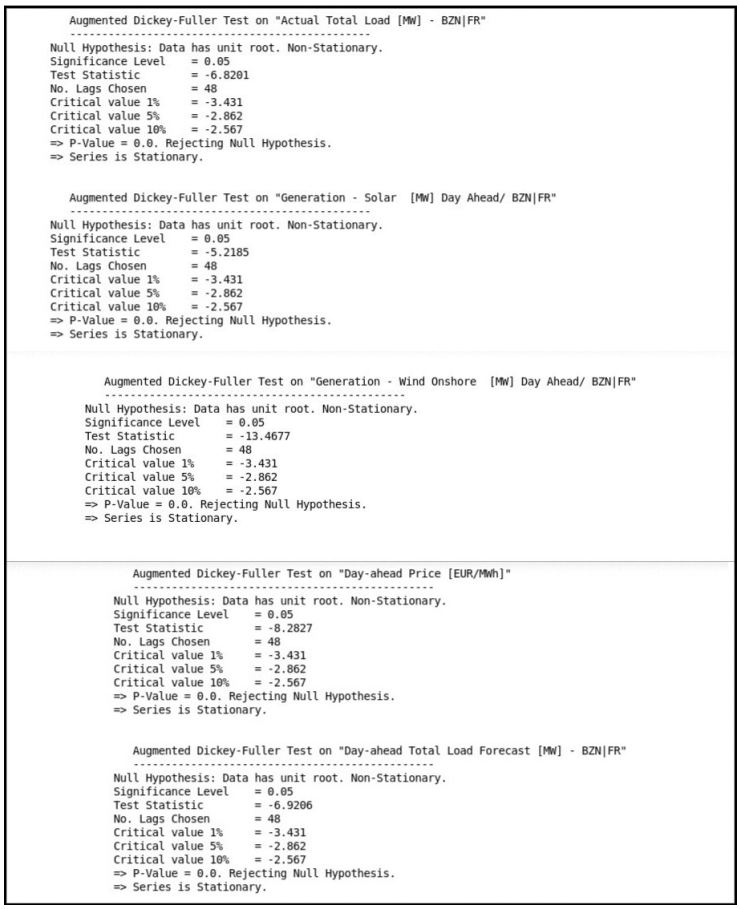
MSE	MAPE	RMSE
51.10267	5.4937	7.1486
32.75744	64.60561	5.72341
41.93	35.04969	6.436012

This is the summary of the VAR Order Selection result obtained and highlights the minimum order score of a model to be selected.

VAR Order Selection (* highlights the minimums)

	AIC	BIC	FPE	HQIC
0	52.21	52.21	4.732e+22	52.21
1	40.28	40.28	3.106e+17	40.28
2	37.34	37.35	1.651e+16	37.35
3	37.09*	37.11*	1.284e+16*	37.10*

By the time I have started to use VAR model, I have calculated Augmented Dickey-Full(ADF) test in order to be informed about the degree to which a null hypothesis can be rejected or fail to be rejected. The result must be interpreted for a given problem to be meaningful. Nevertheless, they can give a quick check and confirmatory evidence that the time series is stationary or non-stationary. As I said from above, ADF test is a type of statistical test called a unit root test. The intuition behind a unit root test is that it determines how strongly a time series is defined by a trend. I have interpreted the result using the p-value from the test. A p-value below a threshold such as 1,5 or 10%, time series is stationary for all data, and i have shown all the values obtained after doing ADF test.



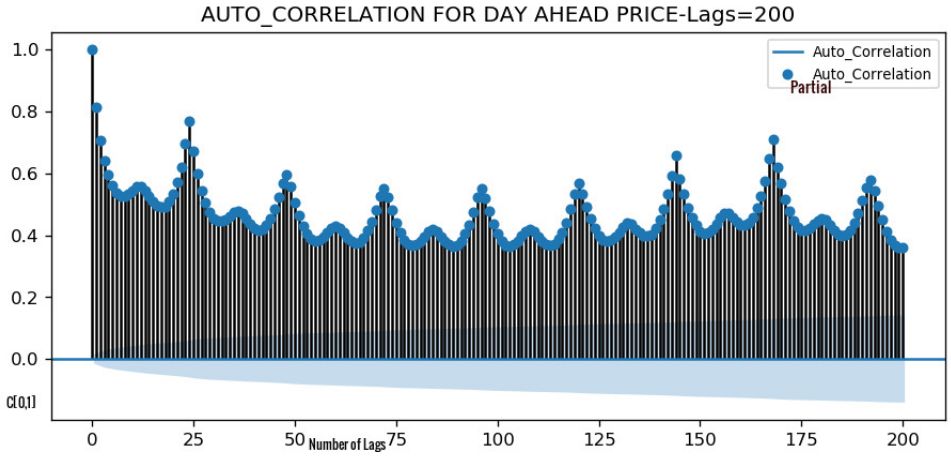
5.3.2 ARIMA Model

During the stage of finding the forecasted price of the recent observations while the step of fitting the model was over, I have found the Mean Squared Error(MSE) in order to select the best ARIMA order, Mean Squared Error is the average squared difference between the actual target variables and predicted variables. I have done the ARIMA model selection by considering the best(minimun) Mean

Table 5: Select The best ARIMA order.

ARIMA Order	MSE
ARIMA(2,0,3)	9.355 :Best MSE
ARIMA(1,0,1)	10.064
ARIMA(1,0,2)	9.819
ARIMA(1,0,3)	9.872
ARIMA(1,1,1)	10.253
ARIMA(1,1,3)	11.149
ARIMA(2,0,1)	9.739
ARIMA(2,0,2)	10.061
ARIMA(2,0,3)	9.355
ARIMA(2,2,3)	9.459

Squared root(MSE), which is ARIMA(2,2,3) with the MSE=9.355. For the first prediction of the price,I have selected only 100 recent observations from 2018 data by taking the range between [10000-10100] after spliting the data into training 90% and testing 10% of the dataset, by the fact that I have seen that ARIMA model does better for short term time series at least 100 preferably observations beyond that, it reduces the chance of the model performance. After doing ARIMA model selection I got Mean Squared Error(MSE) equal to 9.355 as shown in the table above, we knew that the ARIMA order(2,0,3) means that p=2,d=0,q=3 is the suitable model rather than other ARIMA order result. why not d=0? for some values, since I have said that my data time series was stationary, and during time series is stationary d=0, here the reason is that once you divide the data into portion observations, immediately the time series changes and becomes non-stationary, and to continue for further steps of performing the ARIMA model, you have to make the time series stationary and the value of d can change and sometime not equal to 0.

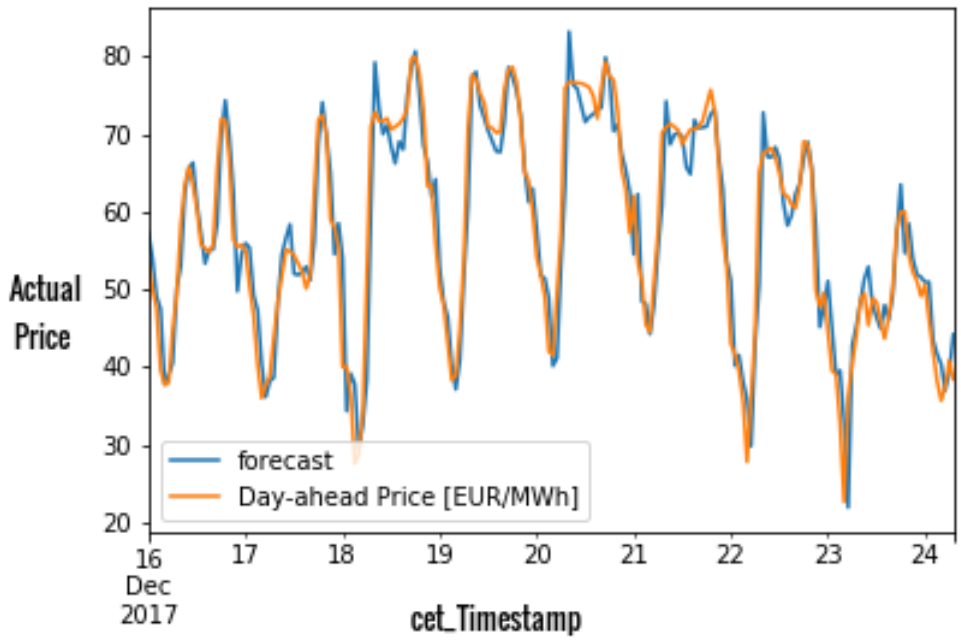


A Partial Auto Correlation Function(PACF) is a summary of the relationship be-

tween an observation in a time series with observations at prior time steps with the relationships of intervening observations removed in the data. Auto Correlation Function(ACF) is a auto-correlation function which give us values of auto-correlation of any series with its lagged values, ACF and PACF are used to determine ARIMA model by applying model-selection acf-pacf function,PACF helps to check and select the best significant lags to use for our ARIMA model, Here from above plot, you can observe that the ACF lag 1 is quite significant since is well above the significance line, even few other lag values below lag 1 are also quite significant as well, slightly managing to cross the significance limit(blue region), I have used the significant lag values to decide for the p-value.

Here I am showing the auto-correlation and partial correlation plot combined while I was going to decide the ARIMA model selection,I have selected 200 lags as shown on the plot on the x axis with the correlation interval between 0 and 1 on the y axis.[8]

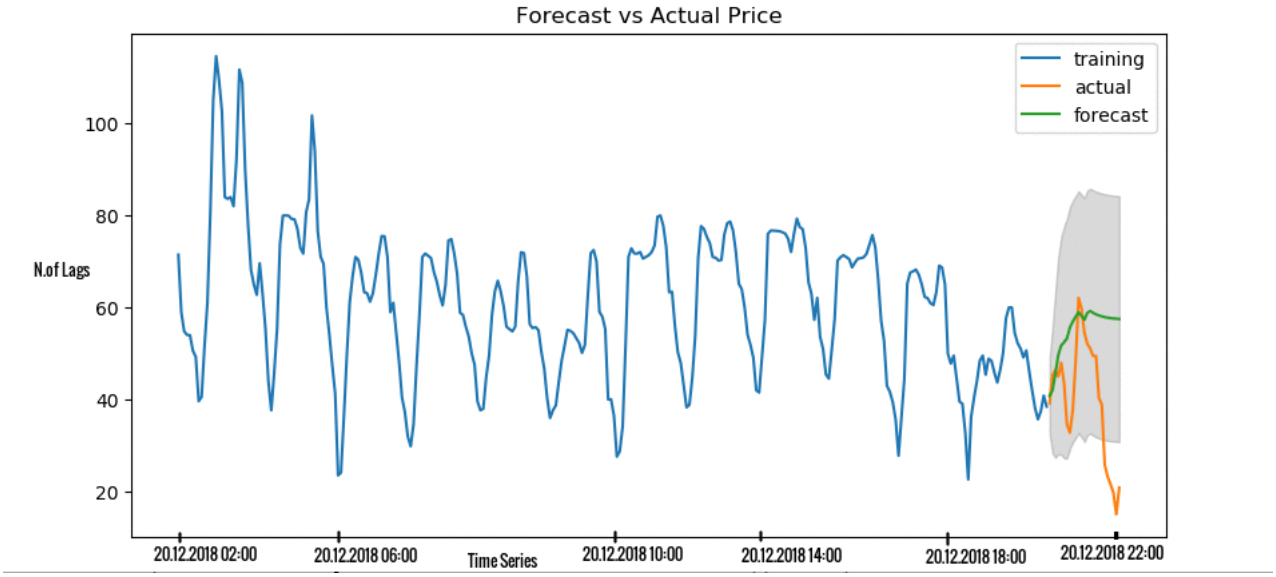
Figure 6: ARIMA Model Fitting for price Result



Model fitting is a measure of how well a machine learning model generalizes to similar data to that on which it was trained. From the plot above I am showing how the model fitted with the selected actual few recent observations(100) in the range of [10000-10100] of the dataset, after determining the ARIMA model selection order which is ARIMA(2,0,3) values of p,d,q, I have everything needed to fit the ARIMA Model. We know that a model that is well-fitted produces more accurate outcomes, as you compare the outcomes to real by observed values of the target variable to determine their accuracy. From the plot actual 100 recent price observations is on y axis while few days of december 2017 are on x axis.

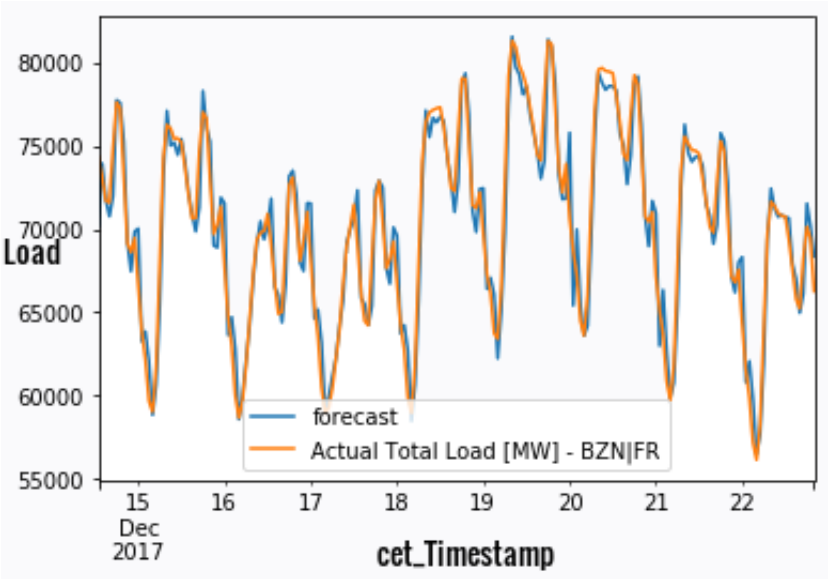
After performing the ARIMA model with the ARIMA(2,0,3) best selection order,

Figure 7: ARIMA Model Forecasted Price result.



from the chart, ARIMA model seems to give a directionally correct forecast for the last 3 hours price forecast from 20.12.2018 18:00 to 20.12.2018 22:00. And also the actual recent observed values lie within the 95% confidence band. But some of the predicted forecasts are consistently below the actuals,which means that there is a chance to get more closer actuals and forecasts while continue playing around with splitting actual recent price observations and getting different best model selection order that depends on small MSE while AIC also reduced, and the accuracy will certainly improve. So, there is definitely scope for improvement.

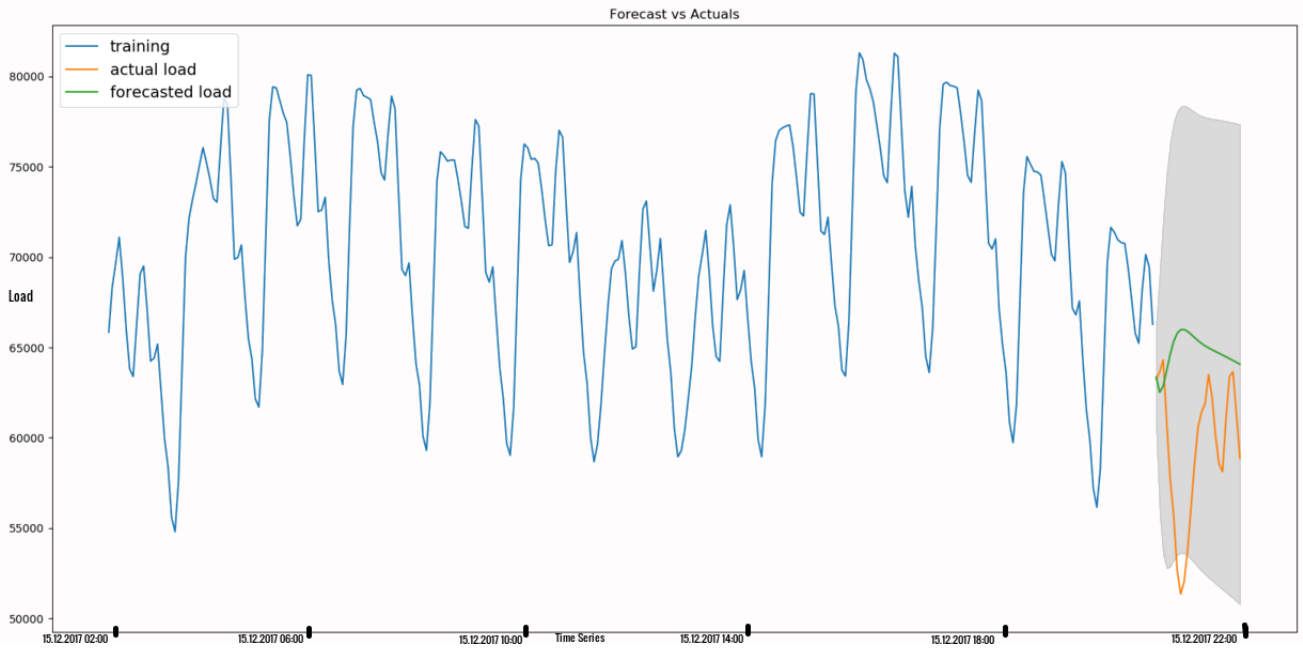
Figure 8: ARIMA Model Fitting for load Result



ARIMA model fitting result for the load I have used the same size of data used

for fitting the price, from the plot I am showing how the ARIMA model fitted with the selected actual few recent observations(100) ub the range of [10000-10100] of the dataset which belongs in the dates of december of 2017 but for trainig dataset I selected 300 recent observations to be shown on the plot, after determining the ARIMA model selection order which is ARIMA(2,2,3) values of p,d,q. ARIMA model seems to give a directionally correct forecast for the last 3 hours load forecast from 20.12.2018 18:00 to 20.12.2018 22:00. And also the actual recent observed values lie within the 95% confidence band. According to how the result for forecasting load looks, it shows that it follows the patterns of actual load.

Figure 9: ARIMA Model Forecasted Load result.



To forecast the load using ARIMA model, I have used same proceduce like forecasting the price except for the best ARIMA model selection order which is ARIMA(2,2,3) model values of p,d,q, which means that we fit the model where this sets the lag value to 2 for autoregression, uses a difference order of 2 to make the time series stationary and also uses a moving average model of 3, give us the best MSE value of 19.266 for the predictions.

5.3.3 Compare the performance of VAR and ARIMA models

we know that to use statistical models such as VAR model and ARIMA model, we should take care of patterns before the data can be fit into the model. For the VAR model we need stationary condition to be performed as well as for ARIMA model by putting enough structure into the model to identity some of the parameters in a model. VAR model performs well in forecasting the load where the forecasted data follows the patterns of the actual load as it is mostly used for long term load forecasting , but VAR model does not perform well in forecasting the price by contrary ARIMA model performs well in forecasting the price and the load. The results of these models are compared with their MSE,MAPE but obtaining accurate forecast results depend on various factors. the MAPE gives relative eros in percentage, which does not depend on the scale of forecasted variables, for VAR model we obtain MSE=41.93 and MAPE=35.04 where for ARIMA model MSE=9.355 and MAPE=0.017. From my point of view I can conclude that the performance of the model depends on the data you are forecasting.

5.3.4 ARIMA Model Challenge Discussion

ARIMA Model is a short term forecast method which is estensively also used to forecast the price,it has to always split time series into small portion of observation(recent observations at least of nbs=100),Generally in machine learning, It is known that the larger the amount of data sets the better the model performance but for ARIMA model it does not work like that, it works in other way around where having few recent number of observations the ARIMA model gets a good accurate forecasted data. But also in another hand it is logical to use recent observation as past variables in order to forecast the future variables especially for our scenario of price,but the challenge is that once you have time series of the actual entire data which is stationary, and select the few recent number of observations its time series becomes non-stationary and force you to make the time series stationary in order to select the best model order.

6 Conclusion

As conclusion,the objective of this report project is to try machine learning techniques that can give a forecast value of the price and load in order to maintain the price stability, we know that a common problem in time series analysis is to make a forecast for the time series at hand, forecasting the price and load is a good idea which helps to control the price, where price monitoring is conducted to achieve good development of energy trade. ARIMA and VAR techniques are used to do forecasts, I have used them to forecast the price and load of France energy market. VAR model performs better for load forecast by considering the recent number of observations while ARIMA model used for short term forecast and it performs good for forecasting the price of 100 recent number of observations in the month of december 2018 and even for the load forecast. But I can conclude that the performance of the model depends on the data you are forecasting. I really enjoyed to work on this project of applying machine learning techniques in time series forecasting specifically by applying ARIMA and VAR models, I will use them easily in future case study especially when it comes to business forecasting or other academic research.

7 Appendix

```
#ARIMA MODEL AND VAR MODEL CODES
#import necessary libraries
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.dates as dates
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
from statsmodels.tsa.api import VAR
from statsmodels.tsa.stattools import adfuller
from statsmodels.tools.eval_measures import rmse, aic
from sklearn.model_selection import train_test_split
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import matplotlib.pyplot as plt
import warnings
from statsmodels.tsa.arima_model import ARIMA
import itertools
from pandas import datetime
from statsmodels.tsa.arima_model import ARIMA
from sklearn.metrics import mean_squared_error
#import the data set from local directory(ARIMA MODEL)
df = pd.read_csv("/root/Adv_Projekt2/combined.csv")
price = "Day-ahead Price [EUR/MWh]"
# apply interpolation to create continuity in dataset
df = df.interpolate(method="cubic")
df = df.set_index("cet_timestamp") # set time as index
df_size = len(df[-17538:]) * 0.95 # take 95 of the data starting from 17538
```

```
df = df.set_index("cet_timestamp") # set time as index
df_size = len(df[-17538:]) * 0.95 # take 95 of the data starting from 17538
# splitting data to train and test
df_train, df_test = df[:int(df_size)], df[int(df_size):]
print(df_train.shape)
print(df_test.shape)
print(df.columns)
#Adfuller Test for Stationary
plt.rcParams.update({'figure.figsize':(9,7), 'figure.dpi':120})

fig, axes = plt.subplots(3, 2, sharex=True)
axes[0, 0].plot(df_test[price]); axes[0, 0].set_title('Original Series')
plot_acf(df_test[price])

# 1st Differencing
axes[1, 0].plot(df_test[price].diff()); axes[1, 0].set_title('1st Order Differencing')
plot_acf(df_test[price].diff().dropna())

# 2nd Differencing
axes[2, 0].plot(df_test[price].diff().diff()); axes[2, 0].set_title('2nd Order Differencing')
plot_acf(df_test[price].diff().diff().dropna())
plt.show()
#creating the ARIMA Model
p=d=q = range(0,6)
pdq = list(itertools.product(p,d,q))
for param in pdq:
    try:
        model = ARIMA(df_train[-100:][price], order=(2, 0, 3))
        model_fit = model.fit()
```

```
        model = ARIMA(df_train[-100:][price], order=(2, 0, 3))
        model_fit = model.fit()
        print(param, model_fit.aic)
    except:
        continue

model = ARIMA(df_train[-200:][price], order=(2,0,3))
model_fit = model.fit()
model_fit.plot_predict(dynamic=False)
plt.show()
# evaluate an ARIMA model for a given order (p,d,q)
def evaluate_arima_model(X, arima_order):
    # prepare training dataset
    train_size = int(len(X) * 0.95)
    train, test = X[0:train_size], X[train_size:]
    print(len(train))
    print(len(test))
    history = [x for x in train]
    # make predictions
    predictions = list()
    for t in range(len(test)):
        model = ARIMA(history, order=arima_order)
        model_fit = model.fit(disp=0)
        yhat = model_fit.forecast()[0]
        predictions.append(yhat)
        history.append(test[t])
    # calculate out of sample error
    error = mean_squared_error(test, predictions)
    return error
```

```
    # calculate out of sample error
    error = mean_squared_error(test, predictions)
    return error

# evaluate combinations of p, d and q values for an ARIMA model
def evaluate_models(dataset, p_values, d_values, q_values):
    dataset = dataset.astype('float32')
    best_score, best_cfg = float("inf"), None
    for p in p_values:
        for d in d_values:
            for q in q_values:
                order = (p,d,q)
                try:
                    mse = evaluate_arima_model(dataset, order)
                    if mse < best_score:
                        best_score, best_cfg = mse, order
                    print('ARIMA%s MSE=%.3f' % (order,mse))
                except:
                    continue
    print('Best ARIMA%s MSE=%.3f' % (best_cfg, best_score))

# evaluate parameters
p_values = [1, 2]
d_values = range(0, 3)
q_values = range(1, 4)
warnings.filterwarnings("ignore")
evaluate_models(df[price][10000:10100], p_values, d_values, q_values)
#Evaluate ARIMA MODEL
def evaluate_arima_model(X, arima_order):
```

```
#Evaluate ARIMA MODEL
def evaluate_arima_model(X, arima_order):
    # prepare training dataset
    train_size = -25
    train, test = X[0:train_size], X[train_size:]
    history = [x for x in train]
    print(len(test))
    # make predictions
    predictions = list()
    for t in range(len(test)):
        print(t)
        model = ARIMA(history, order=arima_order)
        model_fit = model.fit(disp=0)
        yhat = model_fit.forecast()[0]
        predictions.append(yhat)
        history.append(test[t])
    # calculate out of sample error
    error = mean_squared_error(test, predictions)
    #mape = np.mean(np.abs(predictions - test)/np.abs(test))  # MAPE

    print(type(test))
    fc_series = pd.Series(predictions, index=test.index)
    mape = np.mean(np.abs(fc_series - test)/np.abs(test))  # MAPE

    print(mape)
    plt.figure(figsize=(40,20), dpi=100)
    plt.plot(test,label="Test ")
    #plt.plot(train,label="Train")
    plt.plot(fc_series,label="Forecast")
```

```
    #plt.plot(train,label="Train")
    plt.plot(fc_series,label="Forecast")
    plt.legend(loc='upper left', fontsize=8)
    plt.title("Forecast vs Train")
    plt.show()

    return error
#evaluate the arima model using the best selection order(2,0,3)
error = evaluate_arima_model(df[price],(2,0,3))
print(error)

fc, se, conf = model_fit.forecast(25, alpha=0.05)  # 95% conf

# Make as pandas series
fc_series = pd.Series(fc, index=df_test[:25][price].index)
lower_series = pd.Series(conf[:, 0], index=df_test[:25][price].index)
upper_series = pd.Series(conf[:, 1], index=df_test[:25][price].index)

# Plot
plt.figure(figsize=(12,5), dpi=100)
plt.plot(df_train[price][-300:], label='training')
plt.plot(df_test[price][:25], label='actual')
plt.plot(fc_series[:25], label='forecast')
plt.fill_between(lower_series[:25].index, lower_series[:25], upper_series[:25],
                 color='k', alpha=.15)
plt.title('Forecast vs Actuals')
plt.legend(loc='upper right', fontsize=10)
plt.xticks(range(len(index)), index)
plt.show()
```

```python
#VAR MODEL
#cross validation
# Checking if the time series is stationary or not
def adfuller_test(series, signif=0.05, name='', verbose=False):
    """Perform ADFuller to test for Stationarity of given series and print report"""
    r = adfuller(series, autolag='AIC')
    output = {'test_statistic':round(r[0], 4), 'pvalue':round(r[1], 4), 'n_lags':round(r[2], 4), 'n_obs':r[3]}
    p_value = output['pvalue']
    def adjust(val, length= 6): return str(val).ljust(length)

    # Print Summary
    print(f'    Augmented Dickey-Fuller Test on "{name}"', "\n   ", '-'*47)
    print(f' Null Hypothesis: Data has unit root. Non-Stationary.')
    print(f' Significance Level    = {signif}')
    print(f' Test Statistic        = {output["test_statistic"]}')
    print(f' No. Lags Chosen       = {output["n_lags"]}')

    for key,val in r[4].items():
        print(f' Critical value {adjust(key)} = {round(val, 3)}')

    if p_value <= signif:
        print(f" => P-Value = {p_value}. Rejecting Null Hypothesis.")
        print(f" => Series is Stationary.")
    else:
        print(f" => P-Value = {p_value}. Weak evidence to reject the Null Hypothesis.")
        print(f" => Series is Non-Stationary.")
# Select the Order (P) of VAR model
model = VAR(df_train)
for i in [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17]:
```

```python
model = VAR(df_train)
for i in [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17]:
    result = model.fit(i)
    print('Lag Order =', i)
    print('AIC : ', result.aic)
    print('BIC : ', result.bic)
    print('FPE : ', result.fpe)
    print('HQIC: ', result.hqic, '\n')
#Summary of model selection order
x = model.select_order(maxlags=3)
x.summary()
#Forecast
lag_order = model_fitted.k_ar
print(lag_order)  #> 4
forecast_input = df_train.values
fc = model_fitted.forecast(y=model_fitted.y, steps=100)
df_forecast = pd.DataFrame(fc, index=df_test.index[:nobs], columns=df.columns)
print(len(df_forecast))

def forecast_accuracy(forecast, actual):

    mse = mean_squared_error(actual,forecast) # MSE
    mape = np.mean(np.abs(forecast - actual)/np.abs(actual)) * 100  # MAPE
    rmse = np.mean((forecast - actual)**2)**.5  # RMSE

    return({'mse': mse, 'mape': mape, 'rmse': rmse}, [mse,mape,rmse])

def cross_validation(train,test,order):
```

```python
def cross_validation(train,test,order):

    predictions = list()
    for t in range(len(test)):
        model = VAR(train)
        model_fit = model.fit(order)

        yhat = model_fit.forecast(y = model_fit.y, steps = 1)

        predictions.append(yhat[0])

        # add real value to train
        train.loc[len(train)] = test.iloc[t]
        # mse for day_ahead_price
    forecast = pd.DataFrame(predictions, index=test.index, columns=df.columns)
    error = forecast_accuracy(forecast["Day-ahead Price [EUR/MWh]"],test["Day-ahead Price [EUR/MWh]"])
    return error
err_list = list()
cv_train_size = 2923
order = 200
i = 0

while cv_train_size <= len(df):

    new_df = df[:cv_train_size]
    train = new_df[:cv_train_size-25]
    test = new_df[cv_train_size-25:cv_train_size]

    print("\nIteration -> ", i)
```

```python
    print("Test size -> ",len(test))

    # store mse error results
    err = var_model(train,test,order)
    print("MSE -> ", err, end = "\n")
    err_list.append(err)

    cv_train_size += 2923
    i += 1
print("Model Performance\nAverage MSE -> ",sum(err_list)/len(err_list))
forecast = pd.DataFrame(predictions, index=test.index, columns=df.columns)
plt.rcParams["figure.figsize"] = (20,10)
print(forecast_accuracy(forecast["Day-ahead Price [EUR/MWh]"],test["Day-ahead Price [EUR/MWh]
plt.figure(figsize=(12,5), dpi=100)
plt.plot(test["Day-ahead Price [EUR/MWh]"])
plt.plot(forecast["Day-ahead Price [EUR/MWh]"])

def model_selection(train):

    max_lags = 25
    model = VAR(train)

    order = model.select_order(maxlags=max_lags)

    while order.bic == max_lags:

        max_lags += 5
        order = model.select_order(maxlags=max_lags)
```

```python
    while order.bic == max_lags:

        max_lags += 5
        order = model.select_order(maxlags=max_lags)

    orders = [order.aic,order.bic,order.fpe,order.hqic]
    return np.unique(orders)
# selecting the models
all_models = model_selection(df_train)
print(all_models)
# Testing Models Performance
err_list = list()
for m in all_models:
# take 95% of the data
    df_size = len(df) * 0.95
# splitting data to train and test
    df_train, df_test = df[:int(df_size)], df[int(df_size):]
    err,errL = cross_validation(df_train,df_test,m)
    err_list.append(errL)
    print("Train size -> ",len(df_train))
    print("Test size -> ",len(df_test))
    print("Lags (Order) -> ", m)
    print("Loss errors -> ", err, end = "\n")
```

References

[1] Almas Heshmati Dr. Ellen Ueberschär, Jagoda Munic. A review of renewable energy supply and energy efficiency technologies. *Forschungsinstitut zur Zukunft der Arbeit*, 04 2014.

[2] Machine learning algorithm. https://in.mathworks.com/help/stats/machine-learning-in-matlab.html?w.mathworks.com.

[3] entose transparency platform. https://transparency.entsoe.eu/.

[4] Saptarsi Goswami Prapanna Mondal, Labani Shit. Study of effectiveness of timeseries modeling(arima) in forecasting stock prices. *International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol.4, No.2*, April 2014.

[5] Helmut Luetkepohl. Vector autoregressive models. *EUI Working Paper ECO2011/30*, 2011.

[6] He X Luo L, Zhang X. Hospital daily outpatient visits forecasting using a combinatorial model based on arima and ses models. *BMC Health Serv Res 17*, July 2017.

[7] Forecasting:principles and practice. https://otexts.com/fpp2.

[8] Correlation analysis of time series. https://nwfsc-timeseries.github.io/atsa-labs/sec-tslab-correlation-within-and-among-time-series.html.